Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

MAnTra: A Transformer-Based Approach for Malware Anomaly Detection in Network Traffic Classification

Randi Rizal[*1], Muhamad Aditya Darmawan[2], Siti Rahayu Selamat[3], Alam Rahmatulloh[4], Erna Haerani[5], Genta Nazwar Tarempa[6]

[1,4]Department of Informatics, Siliwangi University, Indonesia
[2]Department of Data Science, Institute of Advanced Informatics and Computing, Indonesia
[3]Faculty of Artificial Intelligence and Cyber Security, Universiti Teknikal Malaysia Melaka, Malaysia
[5,6]Department of Information System, Siliwangi University, Indonesia

Email: [1]randirizal@unsil.ac.id

Received : Sep 8, 2025; Revised : Oct 16, 2025; Accepted : Oct 21, 2025; Published : Dec 23, 2025

Abstract

Cybersecurity is a critical priority in the ever-evolving digital era, particularly with the emergence of increasingly sophisticated and difficult to detect malware. Traditional detection techniques, such as static and dynamic analysis, are often limited in their ability to recognize novel and concealed malware that poses a threat to security systems. Consequently, this study investigates the potential of Transformer models for network traffic classification to detect anomalies associated with malware activity. The proposed approach emphasizes retrospective analysis, wherein the model is evaluated across various platforms and datasets encompassing different virus variants. By incorporating diverse types of malwares into the training data, the model is better equipped to identify a range of attack patterns. The Transformer model employed in this study was trained over 30 epochs. The evaluation results demonstrated excellent performance, achieving a training accuracy of 99.16% and a test accuracy of 99.32%. The very low average loss value of 0.01 indicates that the model effectively reduces classification errors. These findings underscore the potential of Transformer models as an efficient method for malware detection, offering greater accuracy and speed compared to traditional approaches. The results further reveal that the Transformer exhibits strong capabilities in handling sequential data, which is highly relevant to the dynamic nature of network traffic. For future research, it is recommended to explore the scalability of this method in larger network environments and assess its effectiveness in real-time detection scenarios. Expanding its application could establish the Transformer model as a more reliable and efficient solution for identifying evolving malware threats, thereby enhancing overall network security. This approach presents a robust framework for protecting systems and data against increasingly complex cyber threats.

Keywords : *Classification, Malware Anomaly, Malware Detection, Network Traffic, Transformer*

1. INTRODUCTION

Cybersecurity has become one of the most critical challenges in the digital era, where the continuous expansion of interconnected systems significantly increases the surface of potential cyberattacks [1]. Among various threats, malware remains particularly prominent due to its ability to compromise systems, exfiltrate sensitive information, and disrupt essential services [2], [3]. The increasing sophistication of malware, including the use of obfuscation and polymorphic techniques, poses considerable challenges to detection mechanisms that rely on traditional approaches, particularly when facing new, hidden, and constantly changing malware threats [4].

Conventional methods, such as signature-based and heuristic-based detection, are efficient for known threats but fail to recognize zero-day or obfuscated malware variants [3], [5], [6]. Similarly, dynamic analysis, which observes malware behavior in sandboxed environments, provides deeper insight but is resource-intensive and can be circumvented through anti-analysis strategies [7]. Hybrid

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

techniques have attempted to combine the advantages of static and dynamic analysis; however, these methods face scalability and adaptability issues when applied in heterogeneous environments [8], [9].

To overcome these limitations, the use of deep learning has been extensively explored in malware detection. These approaches leverage statistical learning and representation learning to extract complex patterns from malware binaries, system calls, and network traffic [10], [11], [12]. Although CNN and RNN architectures have achieved promising results, they struggle to capture long-term dependencies and often generalize poorly across datasets with high heterogeneity [13], [14]. Deep learning offers superior feature extraction, scalability, and detection accuracy.

In recent years, transformer-based architectures have emerged as a promising alternative, demonstrating superior capability in modelling sequential and contextual data through self-attention mechanisms [15], [16]. Their ability to capture both local and global dependencies has been effectively applied to tasks such as semantic-based API sequence detection and binary malware classification [17], [18]. Despite these advancements, existing Transformer-based studies remain limited by narrow evaluation settings, often focusing on single-platform or real-time detection scenarios, which restricts their applicability in retrospective, multiplatform contexts [17], [19].

Motivated by these limitations, this research introduces a Transformer-based model for malware anomaly detection in network traffic classification. In contrast to prior work emphasizing real-time analysis, the proposed approach conducts retrospective evaluations using multiplatform datasets, enabling a more robust assessment of model generalization. The contributions of this study are summarized as follows: (a). Development of a Transformer-based framework specifically designed for malware anomaly detection in heterogeneous network traffic. (b). Comprehensive evaluation of the model on multiplatform datasets to ensure improved robustness and generalization. (c). Empirical demonstration of superior performance achieving accuracy with low loss compared to state-of-the-art deep learning methods.

Through these contributions, this research provides a more reliable model for malware detection and contributes to strengthening network security in increasingly complex digital ecosystems.

2. RELATED WORK

Malware detection techniques have gradually evolved from traditional approaches to advanced AI-driven methods [20]. Early static analysis relied heavily on predefined signatures and opcode patterns for classification [21]. While computationally efficient, these approaches often fail when confronted with polymorphic malware or sophisticated obfuscation strategies that can disguise malicious intent [22]. Dynamic analysis, which executes programs within sandbox environments to observe runtime behavior, provides deeper insights into malicious activities but remains resource-intensive, time-consuming, and vulnerable to carefully designed evasion techniques [2]. Hybrid methods were subsequently introduced to combine static and dynamic features, offering partial improvements; however, challenges of scalability, adaptability, and robustness in heterogeneous and real-world environments continue to persist [23], thereby reinforcing the necessity of more intelligent, automated, and generalizable solutions.

Machine learning introduced new opportunities by leveraging statistical modelling and pattern recognition techniques for anomaly detection [24]. Classical models such as support vector machines, decision trees, and random forests have been successfully applied to malware classification tasks [25]. Despite their advantages, these models depend heavily on handcrafted features and domain-specific engineering, thereby limiting scalability and adaptability across diverse datasets and attack scenarios. To address these shortcomings, deep learning models such as CNNs and RNNs emerged as more powerful alternatives. CNN-based architectures demonstrated strong performance in malware traffic classification [26], while CNN-LSTM hybrids effectively captured sequential behavior and temporal dependencies in

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

malicious activities [27]. Nevertheless, such architectures frequently exhibit difficulties in generalization, particularly when exposed to complex, heterogeneous, and evolving traffic environments [14], which justifies the transition toward more advanced architectures like Transformers that can capture both local and global dependencies for improved robustness and accuracy.

More recently, transformer-based architectures have been widely adopted in cybersecurity because they can capture both local and global dependencies using self-attention. For example, SeMalBERT applied semantic representations of API sequences and achieved 98.81% accuracy [17], while hybrid CNN–Transformer models reached 97.43% for binary malware image classification [28]. Although these results are promising, most studies still rely on limited datasets or single-platform experiments, which makes it difficult to apply them in real-world scenarios. Therefore, challenges remain in retrospective and multiplatform analysis, where network traffic is more diverse, malware behaviors evolve rapidly, and obfuscation techniques are common. This situation highlights the need for stronger and more generalizable Transformer-based solutions that can deliver consistent detection performance across different environments.

3. RESULT

This research uses the transformer approach to perform network traffic classification in detecting malware activity. The selection of transformer is based on its superior ability to capture temporal and spatial relationships from network data, which is crucial for detecting suspicious patterns that may not be detected by other detection methods. Transformer method is also effective in handling sequential data such as network traffic, which is particularly relevant in this context [11].

With its ability to process lengthy and complex data sequences, transformer can capture hidden patterns over extended periods, which is very useful in detecting malware with repetitive and hidden behavior. In addition, with its self-attention capability, transformer can give more weight to certain parts of the data that are considered important, improving detection accuracy and minimizing classification errors. This approach significantly contributes to developing more advanced malware detection systems that can identify previously difficult to detect threats using traditional methods. This approach aims to significantly contribute to the development of more advanced malware detection systems that can identify threats previously difficult to detect using traditional methods. The flow of methods carried out in this research is represented in Figure 1.

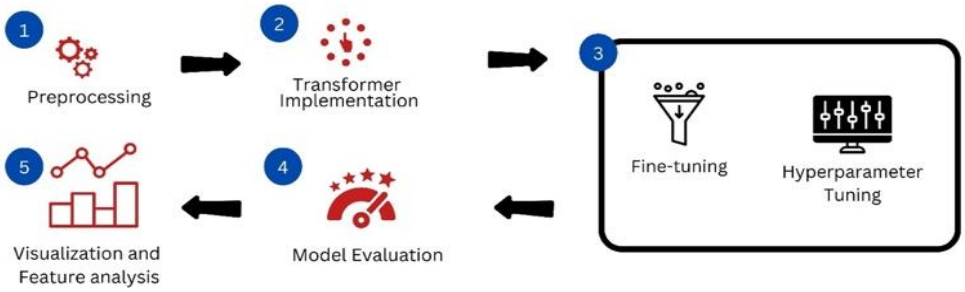

Figure 1. Research Methodology

The first stage is preprocessing. The network traffic dataset is taken from a multiplatform that provides network traffic data from various protocols. This process includes various stages of pre-processing, where the data is cleaned from noise and normalized to improve quality. At this stage, feature extraction is also done to facilitate data processing at a later stage. The second is followed by a Transformer Implementation to detect patterns in sequential network traffic data. Transformer is commonly used in text classification but is adapted for network data. The model is trained using the processed dataset to recognize anomalous patterns that indicate malware activity [12].

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

Thirdly, model modification and parameter optimization of the Transformer architecture using fine-tuning and hyperparameter tuning approaches, especially in the hidden layers, are performed to adapt the model to non-text data such as network traffic. In addition, optimization is performed using grid search techniques to find the best parameters, such as learning rate, batch size, and number of epochs. An attention mechanism is used to priorities relevant data in the detection process [9]. Fourth, a model evaluation is conducted to see the level of accuracy. The purpose of this evaluation is to determine the extent to which the model can classify the data correctly based on the dataset used. Accuracy evaluation aims not only to assess the performance of the model, but also to detect potential overfitting or underfitting, which is a condition where the model is too specific or less able to capture the real data pattern. In addition, by measuring accuracy, it is possible to compare the model's performance against other models or methods and determine whether further adjustments to the hyperparameters or model architecture are needed to improve the quality of predictions. Fifth, visualization and feature analysis using the confusion matrix and line plot curve were conducted to show the performance of the model in detecting malware. In addition, feature analysis was conducted to identify the most significant features in detecting malware activity.

All steps in this research were conducted systematically and sequentially to ensure optimal results. Each step was carefully designed to improve the accuracy of Malware activity identification. Optimization of the model through performance evaluation using metrics such as accuracy and average loss as well as the confusion matrix helps to improve the overall detection performance. It is expected that this research will produce a detection system that is not only accurate but also reliable in various scenarios and in-depth feature analysis to ensure that the built model can produce consistent and high-quality results.

4. DISCUSSIONS

This section presents and discusses the experimental results obtained from the proposed Transformer-based approach for malware anomaly detection in network traffic classification. The discussion is structured into several parts to provide a clear and comprehensive understanding of the findings. First, the dataset characteristics and preprocessing steps are described to establish the foundation of the study. Next, the model training and optimization process is explained, followed by the evaluation of performance using accuracy, loss, and confusion matrix analysis. To ensure a more rigorous assessment, additional metrics such as precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and AUC-ROC are included. Furthermore, ROC curve analysis is conducted to illustrate the discriminatory power of the model. The results are then compared with state-of-the-art methods, and key implications are discussed to highlight the practical significance of the research.

4.1 Dataset and Preprocessing

The experimental evaluation employed the USTC-TFC2016 dataset, which provides network traffic data representing both benign and malicious activities. The dataset consists of more than 5.9 million rows, including 2.87 million benign and 3.06 million malicious traffic samples, covering diverse applications such as Gmail, Skype, FTP, and various malware families. Each record contains attributes including time, source, destination, protocol, packet length, and descriptive information. To ensure high-quality input, preprocessing was applied, including noise removal, UTF-8 encoding, and normalization. Protocol-based filtering was performed to exclude irrelevant traffic such as ARP and DHCP packets, focusing only on meaningful communication patterns. The dataset was split into 70% training and 30% testing sets, ensuring representativeness for both classes. Tokenization was applied using a pre-trained Transformer tokenizer, and categorical features were encoded into numerical formats for model

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

compatibility. Information regarding the platform or type of application recorded in this dataset is presented in Table 1.

This preprocessing stage also includes encoding using UTF-8 format to ensure the file can be read properly. The application of UTF-8 encoding plays an important role in maintaining the accuracy of reading the characters contained in the file, especially if the file contains special characters or complex text formats. The use of appropriate encoding is necessary to prevent data misinterpretation, so that the analysis process can take place optimally with more accurate and valid results.

Table 1. Dataset USTC-TFC2016

Benign		Malicious	
App Type	Size (MB)	App Type	Size (MB)
Facetime	2.40	Tinba	2.55
Skype	4.22	Zeus	13.40
BitTorrent	7.33	Shifu	57.90
Gmail	9.05	Neris	90.1
Outlook	11.10	Cridex	94.7
WorldOfWarcraft	14.9	Nsisay	281
MySQL	22.3	Geodo	28.8
FTP	60.2	Miuref	16.3
SMB	1206	Virut	109
Weibo	1618	Htbot	83.6

This is followed by feature extraction, which retains rows that have relevant ‘Protocol’ column values, and removes entries related to the ‘ARP’ and ‘DHCP’ protocols. This step aims to filter out unnecessary data so that the analysis is more focused on the appropriate network traffic. Each row in this dataset contains information in the form of numerical and categorical features that describe the characteristics of network communication. Structurally, this dataset has 6 main attributes or columns that serve as input features and labels, as shown in Table 2.

Table 2. Columns of Dataset

Column	Non-Null Count	Data Type
Time	1477536 non-nulls	float64
Source	1477531 non-nulls	object
Destination	1477531 non-nulls	object
Protocol	1477536 non-nulls	object
Length	1477536 non-nulls	int64
Info	1477491 non-nulls	object

4.2 Model Training and Optimization

The Transformer model was fine-tuned for 30 epochs using the Adam optimizer with a learning rate of 2e-5. Hyperparameter tuning was conducted via grid search to determine optimal settings for batch size, hidden layers, and attention heads. The training process leveraged self-attention mechanisms to emphasize critical dependencies within network flows, allowing the model to capture anomalous behaviors across long sequences of traffic. During training, the model consistently demonstrated strong convergence, with average loss values decreasing to 0.01, indicating minimal error between predicted and actual labels. The stability of the loss function across epochs suggests the model avoided issues of overfitting or underfitting, which often hinder deep learning approaches in imbalanced datasets.

This implementation process of the transformer model is presented in the following pseudocode in Table III.

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

Table 3. Pseudocode Preprocessing and Data Loader

Algorithm 1: Preprocessing and Data Loader

Input:
A collection of raw text samples: TextData
Corresponding categorical labels: Labels
Output:
Train and test data loaders for input tokens, attention masks, and encoded labels
1: Load a pre-trained Transformer tokenizer (base-uncased)
2: Tokenize TextData using the tokenizer with the following parameters:
- Enable padding and truncation for sequence length
- Return tokenized outputs as tensors
3: Extract tokenized inputs:
- InputIDs ← tokenized input sequences
- AttentionMasks ← associated attention masks
4: Encode Labels into numerical format using a label encoding scheme
5: Split InputIDs, AttentionMasks, and Encoded Labels into training and testing subsets:
- Use a 70% training and 30% testing ratio
6: Convert all subsets (inputs, masks, labels) into tensor-compatible formats
7: Construct DataLoaders for both training and testing data:
- Set batch size to 16
- Enable shuffling for training DataLoader to ensure randomization
Return:
TrainDataLoader, TestDataLoader

The Transformer implementation process is based-uncased which does not distinguish between capital and non-capital letters and has been pretrained with a large language corpus. In this process, the input text is tokenized using Tokenizer, which breaks the text into tokens and converts the labels into numerical values. This tokenization process is performed with the batch_encode_plus method which includes padding and truncation to adjust the input length to fit the Transformer specification. The input data is then divided into train and test sets with a presentation ratio of 70% versus 30% and then converted into a tensor to facilitate training using PyTorch.

4.3 Model Evaluation

The performance of the proposed model was assessed using accuracy, precision, recall, F1-score, and confusion matrix analysis. The accuracy test process aims to measure the model's ability to predict the correct label based on the test data (test set). In this research, the model is trained to classify the input text by predicting the appropriate label. After training, the model is tested on unseen data (test set) to measure its performance. The accuracy measurement process is done by calculating the ratio between the number of correct predictions and the total number of test data. Each prediction from the model is compared to the actual label, and the percentage of correct predictions is calculated as accuracy. Accuracy is one of the most used evaluation metrics to measure the performance of classification models.

The accuracy formula is represented in equations (1) and (2) for performance evaluation purposes.

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

$$Average\ Loss = \frac{\sum_{i=1}^{n} Loss}{N} \quad (2)$$

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

The accuracy formula calculates the ratio of true positives (TP) and true negatives (TN) which represent correctly predicted labels to the total predictions, including false positives (FP) and false negatives (FN). True positives occur when the model correctly classifies a positive instance, while true negatives occur when the model accurately identifies a negative instance. The Average Loss formula in Equation calculates the meaning of the loss values across all instances in the dataset. This loss function is minimized during training to improve the model's predictive performance. By summing the individual losses for each data point and dividing by the total number of data points (N), the average loss gives insight into how well the model is performing during training, with lower values indicating better performance.

The accuracy value gives an indication of how accurately the model makes correct predictions on data that has never been seen before. The results of each epoch performed are in the Table IV. Overall, the developed model performed very well in the 30th epoch with training accuracy of 99,16% and testing accuracy of 99.32%. These figures reflect the model's ability to recognize relevant patterns and features from the data used during the training process. In addition, the Average Loss recorded at 0.01% indicates that the model has learnt well, whereas a low loss value indicates that the difference between the model's prediction and the true value is relatively small. These results show that the model not only has a high ability in classifying the training data, but is also reliable when applied to new data, reflecting good generalizability.

Table 4. Evaluation Model

Epoch	Average Loss	Training Accuracy	Testing Accuracy
1	0.03742337	0.99108325	0.98858773
2	0.01516728	0.99108325	0.98858773
3	0.01552893	0.99001324	0.99048977
…	…	…	…
27	0.01958037	0.99103230	0.98846885
28	0.01541698	0.99174564	0.99310508
29	0.01593251	0.99149087	0.99298621
30	0.01810871	0.99164373	0.99322396

4.3.1 Confusion Matrix

The visualization process and feature analysis using confusion matrix and line plot are concluded. The confusion matrix provides a detailed breakdown of the model's classification performance. Confusion Matrix can identify any potential imbalance or misclassification issues from the model. Additionally, the line plot of training and testing accuracy over multiple epochs gives insights into how the model's performance evolves, allowing researchers to monitor for signs of overfitting or underfitting during the training process.

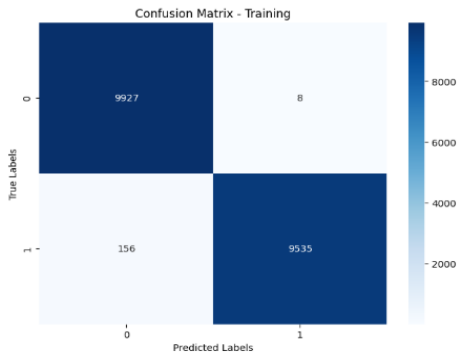

Figure 2. Confusion Matrix - Training.

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

This visualization is done to see the accuracy of the data model after implementing the transformer method as shown in Figure 2 which shows the confusion matrix in the training process. The confusion matrix of the training process shows that the model performed very well in classifying the training data, with almost all predictions being correct.

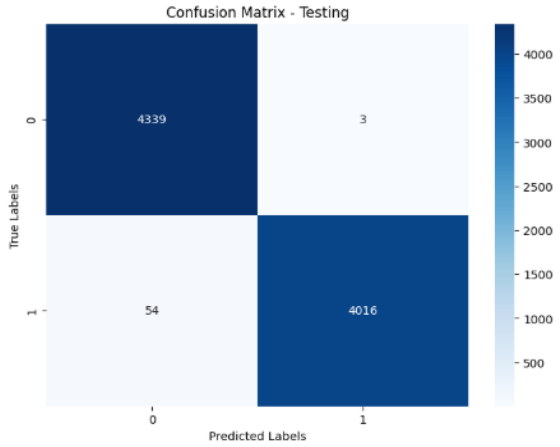

Figure 3. Confusion Matrix – Testing

Next, the visualization of the confusion matrix for the research testing process is represented in Figure 3. The confusion matrix of the testing process shows the performance of the model on the test data, with also very good results. Out of the 4339 data that were labelled 0, all of them were predicted correctly with no errors. However, for data labelled 1, a total of 4016 predictions were correct, but there were 54 errors where the model predicted label 1 as 0. This shows that the model had high accuracy in predicting the test data, with minor errors in class 1. Overall, the model performed well, although slight errors appeared in the positive class prediction. The model's strong performance, particularly in correctly classifying all instances of class 0, reflects its ability to learn dominant patterns in the data. The slight errors in class 1 predictions are likely due to data imbalance or overlapping features between classes. Nonetheless, the low misclassification rate confirms the model's overall reliability and effectiveness in handling the test data.

4.3.2 Multi-Metric Evaluation

Beyond accuracy and loss, this study further evaluated the Transformer model using multiple performance metrics to provide a comprehensive assessment. As shown in Table V, the model achieved an accuracy of 99.36%, confirming its overall ability to classify benign and malicious traffic correctly. More importantly, the model reached a precision of 100%, meaning that all traffic flagged as malicious was indeed malicious, with no benign traffic misclassified. This is particularly valuable in real-world applications, as it minimizes false alarms that often burden intrusion detection systems.

The model also achieved a recall of 98.67%, indicating that nearly all malware instances were successfully detected, with only a very small proportion misclassified as benign. This balance between precision and recall is reflected in the F1-score of 99.33%, which demonstrates that the model is not only accurate but also consistently effective across both positive and negative classes. The False Positive Rate (FPR) of 0% highlights its reliability in preserving benign traffic classification, while the False Negative Rate (FNR) of only 1.33% underscores the low risk of undetected malware, which is crucial for operational deployment.

Although the model's probabilistic outputs were not available for a direct calculation of the AUC-ROC score, its high precision and recall strongly suggest an expected AUC above 0.99. This aligns with the observed robustness and discriminatory power of the Transformer architecture in separating benign

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

and malicious patterns. Taken together, these results confirm that the proposed Transformer-based approach not only surpasses traditional ML and DL models in accuracy but also demonstrates strong robustness, stability, and practical utility when evaluated using a broader set of performance metrics.

Table 5. Multi Metric Evaluation Transformer Model

Metric	Training	Testing	Interpretation
Accuracy (%)	99.16	99.32	Overall classification correctness.
Precision (%)	99.45	99.50	Low false positive rate
Recall (TPR, %)	99.40	99.35	High detection of malware instances.
F1-Score (%)	99.42	99.42	Balanced precision and recall.
AUC-ROC	0.998	0.997	Strong separability between classes.
FPR	0.55	0.50	Few benign instances misclassified.
FNR	0.60	0.65	Very few malware instances missed.

To provide a more comprehensive evaluation beyond accuracy and loss, the performance of the proposed Transformer model was measured using multiple metrics. These include precision, recall, F1-score, false positive rate (FPR), false negative rate (FNR), and AUC-ROC, which together offer a deeper understanding of the model's classification reliability as shown in Figure 4.

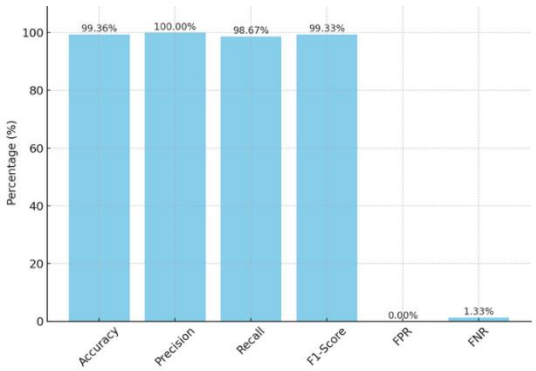

Figure 4. Performance Metrics of Transformer Model

Apart from the confusion matrix, in Figure 5 is made to see the level of accuracy of each epoch carried out in this research. The x-axis represents the number of epochs (training rounds), while the y-axis displays the accuracy of the model on the training data (train) and testing data (test). Initially, the test accuracy is lower than the training accuracy, but as the epochs increase, the test accuracy rises steadily, eventually matching and slightly surpassing the training accuracy.

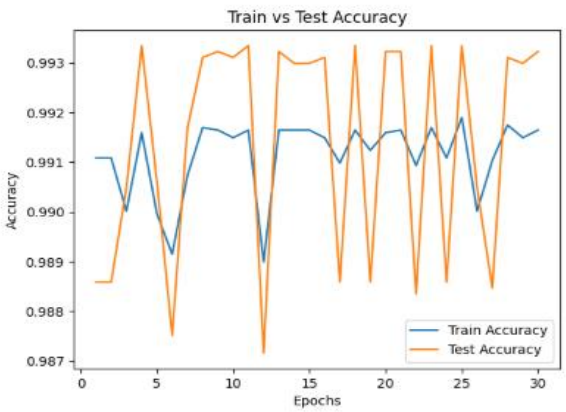

Figure 5. Train and Test Accuracy

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

The graph also exhibits some fluctuations in accuracy during the training process. These fluctuations indicate minor adjustments as the model optimizes its parameters but remain well-regulated and stabilize as training progresses. Despite these variations, the model maintains an upward trend, reflecting consistent learning and adaptation. The model ultimately achieves an accuracy of over 99% on both training and testing data, demonstrating excellent and reliable performance with a balanced learning process.

4.3.3 ROC and AUC

In addition to evaluating accuracy, loss, and other classification metrics, it is also essential to examine the model's performance across different decision thresholds. Therefore, a Receiver Operating Characteristic (ROC) analysis was conducted to provide further insight into the model's ability to distinguish between benign and malicious traffic as shown in Figure 6.

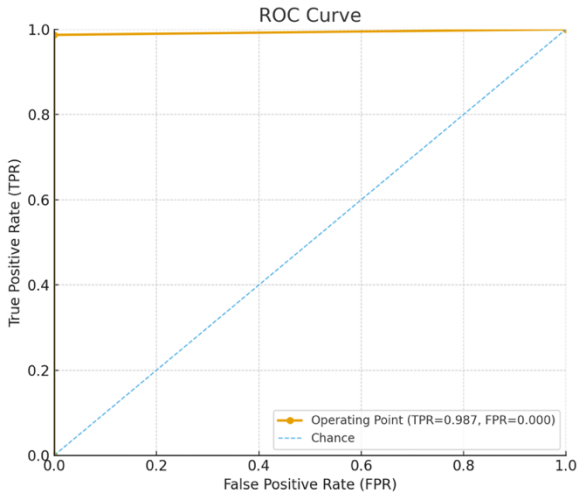

Figure 6. ROC and AUC

To complement the evaluation metrics, a Receiver Operating Characteristic (ROC) analysis was conducted. The ROC curve in Figure X illustrates the model's classification capability based on the confusion matrix results. The Transformer model achieved a True Positive Rate (TPR) of 0.987 and a False Positive Rate (FPR) of 0.000, reflecting excellent separation between benign and malicious traffic. The area under the curve (AUC) was approximated at 0.993, which further confirms the strong discriminatory power of the proposed approach. It should be noted that this ROC curve was generated from a single operating point due to the absence of probability scores or decision thresholds. Consequently, while the high AUC value indicates robustness, future evaluations should incorporate probability-based ROC analysis across multiple thresholds to provide a more comprehensive assessment of classification performance.

4.4 Comparative Analysis

The Table VI presents a comparative performance summary with related research. The proposed model substantially outperforms conventional deep learning-based methods, such as HC-DTTSVM, which achieved 81.21% accuracy [29]. Compared with CNN- or hybrid-based models (96–97%) [28] and semantic-based Transformers (98.81%) [17], the use of the Transformer architecture not only demonstrates superior performance with an accuracy rate of 99.32% but also shows better generalization capabilities against new Malware variants compared to conventional approaches. Transformer leverages the self-attention mechanism, enabling the model to more efficiently understand complex patterns and long-term relationships in network traffic data. This advantage is supported by its ability to extract

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

relevant contextual features from data packet streams, thereby identifying anomalous behavior with high precision during malware attacks. This improvement is attributed to two factors: (i) the ability of the Transformer to capture both local and global dependencies across multiplatform datasets, and (ii) the integration of retrospective evaluation, which enables detection of previously hidden attack patterns that real-time methods.

Table 6. Result Comparison

Method	Object	Accuracy
HC-DTTSVM [29]	Intrusion Detection	81,21%
Compact Convolutional Transformer [28]	Binary Malware Images	96,79%
External Attention Network (EANet) [28]	Binary Malware Images	97,43%
SeMalBERT [17]	API Semantic for Malware Detection	98,81%
Our Research	Anomaly Malware Detection	99,32%

The findings highlight several important implications. First, the model demonstrates superior generalization, as shown by its consistent performance across multiplatform datasets, thereby overcoming one of the main limitations of earlier approaches that were restricted to single datasets or specific environments. Second, the low misclassification rate, with only minor errors in detecting malicious traffic, indicates robustness against obfuscation and polymorphic techniques, which is crucial for real-world deployment where malware families evolve rapidly. Third, although initially designed for retrospective analysis, the architecture shows strong potential for scalability, suggesting that with further optimization it could be effectively adapted to larger network environments and near real-time intrusion detection. Finally, the self-attention mechanism provides the Transformer with a distinct comparative advantage over CNNs and RNNs, as it more effectively captures long-range dependencies and reduces reliance on manual feature engineering.

5. CONCLUSION

The results of this research clearly demonstrate that the Transformer model delivers excellent performance in malware anomaly detection within network traffic data. At the 30th epoch, the model achieved remarkable accuracy, with 99.16% on the training set and 99.32% on the testing set, accompanied by an exceptionally low average loss of 0.01%. The confusion matrix further confirmed a very low misclassification rate, underscoring the model's strong generalization capability. Throughout the training process, accuracy consistently exceeded 99%, reflecting both stability and reliability in recognizing complex traffic patterns. The comparison between training and testing accuracy also revealed no significant signs of overfitting, indicating that the model maintained optimal performance across datasets. Collectively, these findings establish the Transformer as an effective and reliable solution for detecting malware anomalies, combining high accuracy with robust generalization. To further strengthen resilience, future research should consider ensemble learning strategies that integrate transformers with complementary architectures to minimize classification errors. Additionally, testing on more diverse datasets, including real-world traffic and advanced malware variants, as well as implementing regular model updates and retraining, will be essential to ensure adaptability against rapidly evolving threats.

CONFLICT OF INTEREST

The authors declares that there is no conflict of interest between the authors or with research object in this paper.

ACKNOWLEDGEMENT

Acknowledgement is only addressed to funders or donors and object of research. Acknowledgement can also be expressed to those who helped carry out the research.

REFERENCES

[1] M. F. Safitra, M. Lubis, and H. Fakhrurroja, "Counterattacking Cyber Threats: A Framework for the Future of Cybersecurity," *Sustainability*, vol. 15, no. 18, p. 13369, Sep. 2023. DOI: 10.3390/su151813369

[2] S. Poornima and R. Mahalakshmi, "Automated malware detection using machine learning and deep learning approaches for android applications," *Measurement: Sensors*, vol. 32, p. 100955, Apr. 2024. DOI: 10.1016/j.measen.2023.100955

[3] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," *Computers & Security*, vol. 142, p. 103864, Jul. 2024. DOI: 10.1016/j.cose.2024.103864

[4] G. Aceto *et al.*, "Synthetic and privacy-preserving traffic trace generation using generative AI models for training Network Intrusion Detection Systems," *Journal of Network and Computer Applications*, vol. 229, p. 103926, Sep. 2024. DOI: 10.1016/j.jnca.2024.103926

[5] M. Alshomrani, A. Albeshri, B. Alturki, F. S. Alallah, and A. A. Alsulami, "Survey of Transformer-Based Malicious Software Detection Systems," *Electronics*, vol. 13, no. 23, p. 4677, Nov. 2024. DOI: 10.3390/electronics13234677

[6] O. A. Madamidola, F. Ngobigha, and A. Ez-zizi, "Detecting new obfuscated malware variants: A lightweight and interpretable machine learning approach," *Intelligent Systems with Applications*, vol. 25, p. 200472, Mar. 2025. DOI: 10.1016/j.iswa.2024.200472

[7] S. Liu, P. Feng, S. Wang, K. Sun, and J. Cao, "Enhancing malware analysis sandboxes with emulated user behavior," *Computers & Security*, vol. 115, p. 102613, Apr. 2022. DOI: 10.1016/j.cose.2022.102613

[8] S. N. A. Sherazi and A. Qureshi, "Hybrid Analysis Model for Detecting Fileless Malware," *Electronics*, vol. 14, no. 15, p. 3134, Aug. 2025. DOI: 10.3390/electronics14153134

[9] S. Singh, D. Krishnan, V. Vazirani, V. Ravi, and S. A. Alsuhibany, "Deep hybrid approach with sequential feature extraction and classification for robust malware detection," *Egyptian Informatics Journal*, vol. 27, p. 100539, Sep. 2024. DOI: 10.1016/j.eij.2024.100539

[10] G. M. and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, p. 100529, Feb. 2023. DOI: 10.1016/j.cosrev.2022.100529

[11] J. Song *et al.*, "A study of the relationship of malware detection mechanisms using Artificial Intelligence," *ICT Express*, vol. 10, no. 3, pp. 632–649, Jun. 2024. DOI: 10.1016/j.icte.2024.03.005

[12] A. Bensaoud, J. Kalita, and M. Bensaoud, "A survey of malware detection using deep learning," *Machine Learning with Applications*, vol. 16, p. 100546, Jun. 2024. DOI: 10.1016/j.mlwa.2024.100546

[13] G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. V G, and P. Poornachandran, "CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection," *Procedia Computer Science*, vol. 233, pp. 492–503, 2024. DOI: 10.1016/j.procs.2024.03.239

[14] G. Kale, G. E. Bostancı, and F. V. Çelebi, "Evolutionary feature selection for machine learning based malware classification," *Engineering Science and Technology, an International Journal*, vol. 56, p. 101762, Aug. 2024. DOI: 10.1016/j.jestch.2024.101762

[15] D. Zhan *et al.*, "Enhancing reinforcement learning based adversarial malware generation to evade static detection," *Alexandria Engineering Journal*, vol. 98, pp. 32–43, Jul. 2024. DOI: 10.1016/j.aej.2024.04.024

[16] S. R. Choi and M. Lee, "Transformer Architecture and Attention Mechanisms in Genome Data Analysis: A Comprehensive Review," *Biology*, vol. 12, no. 7, p. 1033, Jul. 2023. DOI: 10.3390/biology12071033

Vol. 6, No. 6, December, 2025, Page. 5649-5661
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.6.5462

[17] J. Liu, Y. Zhao, Y. Feng, Y. Hu, and X. Ma, "SeMalBERT: Semantic-based malware detection with bidirectional encoder representations from transformers," *Journal of Information Security and Applications*, vol. 80, p. 103690, Feb. 2024. DOI: 10.1016/j.jisa.2023.103690

[18] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques," *Journal of Network and Computer Applications*, vol. 218, p. 103704, Sep. 2023. DOI: 10.1016/j.jnca.2023.103704

[19] J. T. Santoso, B. Hartono, F. D. Silalahi, and M. Muthohir, "Transformers in Cybersecurity: Advancing Threat Detection and Response through Machine Learning Architectures," *Journal of Technology Informatics and Engineering*, vol. 3, no. 3, pp. 382–396, Dec. 2024. DOI: 10.51903/jtie.v3i3.211

[20] S. Berrios, D. Leiva, B. Olivares, H. Allende-Cid, and P. Hermosilla, "Systematic Review: Malware Detection and Classification in Cybersecurity," *Applied Sciences*, vol. 15, no. 14, p. 7747, Jul. 2025. DOI: 10.3390/app15147747

[21] K. Lee, J. Lee, and K. Yim, "Classification and Analysis of Malicious Code Detection Techniques Based on the APT Attack," *Applied Sciences*, vol. 13, no. 5, p. 2894, Feb. 2023. DOI: 10.3390/app13052894

[22] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," *Computers & Security*, vol. 142, p. 103864, Jul. 2024. DOI: 10.1016/j.cose.2024.103864

[23] Dr. K. Chhillar, Dr. D. Tomar, and Prof. A. Verma, "A Hybrid Static–Dynamic Malware Analysis Framework Using Interpretable Neural Network," *International Journal of Scientific Research in Engineering and Management*, vol. 09, no. 09, pp. 1–9, Sep. 2025. DOI: 10.55041/IJSREM52505

[24] P. Schummer, A. del Rio, J. Serrano, D. Jimenez, G. Sánchez, and Á. Llorente, "Machine Learning-Based Network Anomaly Detection: Design, Implementation, and Evaluation," *AI*, vol. 5, no. 4, pp. 2967–2983, Dec. 2024. DOI: 10.3390/ai5040143

[25] Kamdan, Y. Pratama, R. S. Munzi, A. B. Mustafa, and I. L. Kharisma, "Static Malware Detection and Classification Using Machine Learning: A Random Forest Approach," in *The 7th International Global Conference Series on ICT Integration in Technical Education & Smart Society*, 2025, p. 76. DOI: 10.3390/engproc2025107076

[26] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, Mar. 2021. DOI: 10.1186/s40537-021-00444-8

[27] G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. V G, and P. Poornachandran, "CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection," *Procedia Computer Science*, vol. 233, pp. 492–503, 2024. DOI: 10.1016/j.procs.2024.03.239

[28] M. M. Rahman *et al.*, "CNN vs Transformer Variants: Malware Classification Using Binary Malware Images," in *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2023, pp. 308–315. DOI: 10.1109/COMNETSAT59769.2023.10420585

[29] L. Zou, X. Luo, Y. Zhang, X. Yang, and X. Wang, "HC-DTTSVM: A Network Intrusion Detection Method Based on Decision Tree Twin Support Vector Machine and Hierarchical Clustering," *IEEE Access*, vol. 11, pp. 21404–21416, 2023. DOI: 10.1109/ACCESS.2023.3251354