https://doi.org/10.31436/iiumej.v27i1.4009

Development of a Model for Malware Detection and Classification at the Byte Level Based on Transformer

RULOF BALTWIN TALLANE[1], AKHMAD UNGGUL PRIANTORO[1*], RIZA MUHIDA[2]

[1] *Faculty of Information and Technology, Budi Luhur University, Jakarta, Indonesia*
[2]*Faculty of Engineering, Bandar Lampung University, Lampung, Indonesia*

**Corresponding author: akhmad.unggul@budiluhur.ac.id*

(Received: 24 October 2025; Accepted: 20 December 2025; Published online: 12 January 2026)

ABSTRACT: Malware threats have been a critical concern in cybersecurity, particularly due to the increasing complexity and constantly evolving variants that are difficult to detect using conventional signature-based or static rule-based methods. This research focused on developing a Transformer-based model at the byte level to detect and classify malware effectively and adaptively, thereby streamlining the analytical process without requiring specialized feature tokenization. The primary objective was to design and evaluate a Transformer model that captures universal and adaptive malware patterns directly from raw byte representations, enabling cross-platform applicability. A quantitative experimental approach was employed using three public datasets: Malware Detection PE-Based Analysis, MC-dataset-binary, and Malware.zip. Data processing involved byte embedding, dilated 1D convolution, multi-head self-attention, and attention pooling. Model optimization was conducted using AdamW with a combined scheduler, Stochastic Weight Averaging (SWA), random byte masking augmentation, and Mixup Embedding. Experimental results showed that the byte-level Transformer model achieved high classification accuracy across the three datasets, namely 99%, 92%, and 94%, respectively. These results demonstrate that a byte-level Transformer can effectively capture universal malware patterns in binary data, offering a flexible and highly accurate approach to developing resilient defenses against modern cyber threats.

ABSTRAK: Ancaman perisian perosak (malware) merupakan isu kritikal dalam keselamatan siber, terutama apabila disebabkan oleh peningkatan kerumitan dan variasi yang sentiasa berevolusi, sekaligus menyukarkan pengesanan menggunakan kaedah konvensional berasaskan tandatangan atau peraturan statik. Kajian ini memfokuskan kepada pembangunan model berasaskan Transformasi pada peringkat bait (byte-level) bagi mengesan dan mengklasifikasikan perisian perosak secara berkesan dan adaptif, tanpa memerlukan pengekstrakan atau penandaan ciri khusus. Objektif utama kajian adalah bagi mereka bentuk dan menilai keberkesanan model Transformasi yang mampu menangkap corak perisian perosak bersifat universal dan adaptif secara langsung daripada representasi bait mentah, seterusnya membolehkan kebolehgunaan merentas platform. Pendekatan eksperimen kuantitatif digunakan dengan melibatkan tiga set data awam, iaitu Analisis Berdasarkan Pengesanan Perisian Perosak PE, Set Data Binari MC, dan Malware.zip. Pemprosesan data merangkumi pembenaman bait, konvolusi 1D terlarut (dilated), perhatian kendiri berbilang kepala, serta pengumpulan berasaskan perhatian. Pengoptimuman model dilaksanakan menggunakan AdamW dengan penjadual gabungan, Purata Pemberat Stokastik (SWA), penambahan data melalui penyamaran bait rawak, dan Penggabungan Embedding. Dapatan kajian melalui eksperimen menunjukkan bahawa model Transformasi peringkat bait mencapai ketepatan pengelasan tertinggi, iaitu masing-masing 99%, 92%, dan 94% bagi ketiga-tiga set data. Dapatan ini membuktikan bahawa Transformasi peringkat bait berupaya

menangkap corak perisian perosak universal daripada data binari, sekaligus menawarkan pendekatan fleksibel dan berketepatan tinggi bagi membangunkan pertahanan yang lebih berdaya tahan terhadap ancaman siber moden.

KEYWORDS: *Malware, Byte Level, Transformer, Machine Learning, Cybersecurity*

1. INTRODUCTION

The rapid advancement of information technology has significantly transformed various aspects of modern society. However, this progress has also introduced substantial challenges in the field of cybersecurity. Among the most critical threats is malware, a type of malicious software designed to disrupt, damage, or gain unauthorized access to computer systems. The severity of malware threats lies in their potential to cause serious harm to both individuals and organizations. According to the Indonesia Cybersecurity Landscape Report 2024 published by Badan Siber dan Sandi Negara (BSSN - the Indonesian National Cyber and Crypto Agency), malware remains one of the top five reported incident categories, alongside data breaches (39.1%), traffic anomalies (33.3%), malicious activity (6.3%), and trojans (5.6%). These highlight malware as a persistent and pressing issue within the national cybersecurity landscape, demanding effective mitigation strategies to safeguard digital infrastructures [1].

In recent years, malware detection has shifted from conventional rule-based or signature-based approaches to machine-learning-driven methods. This transformation represents not merely a technological trend but also a strategic response to the increasingly complex and dynamic nature of modern cyber threats. Contemporary malware variants employ sophisticated evasion techniques such as obfuscation, packing, and polymorphism, making them difficult to detect with traditional methods. Static rule-based detection and simple heuristic analysis often fall short in addressing zero-day malware and novel attack vectors. Consequently, machine learning has emerged as a promising approach for developing adaptive, data-driven detection systems.

Machine learning enables models to learn directly from data rather than relying solely on handcrafted rules. By leveraging various classification and representation learning techniques, machine learning models can generalize beyond previously known patterns, thereby detecting new and unseen malware variants. In the context of malware detection, two major approaches exist: static analysis, which extracts structural features such as byte n-grams, opcodes, or file headers without executing the malware [2] and dynamic analysis, which observes the behavior of suspicious files in a controlled environment, such as API calls, registry modifications, or network connections [3]. Both approaches generate feature sets that can subsequently be processed using machine learning algorithms for malware classification.

Despite their effectiveness, conventional machine learning methods often depend on feature engineering and tokenization, which increase analytical complexity and reduce model flexibility. To address these limitations, this research proposes a byte-level Transformer-based approach for malware detection and classification. Transformers, powered by the multi-head self-attention mechanism, are capable of modeling contextual dependencies across data sequences [4]. Operating directly at the byte level eliminates the need for explicit feature tokenization, such as opcodes or API calls, thereby simplifying the analytical pipeline. Moreover, byte-level representation offers better generalization across different file types and operating systems, making the approach inherently more adaptive and platform-independent.

The main objective of this study was to design and evaluate a byte-level Transformer model that effectively detects and classifies malware variants, including zero-day attacks, while

https://doi.org/10.31436/iiumej.v27i1.4009

reducing reliance on domain-specific feature extraction. This approach is expected to enhance detection accuracy, improve model adaptability, and provide practical cross-platform applicability for modern cybersecurity systems.

2. RELATED WORKS

This section presents the theoretical foundations and related works that underpin this research. The discussion covers deep learning, the Transformer architecture, and the fundamentals of malware, including malware classification. Furthermore, it reviews prior studies on malware detection using byte-level and Transformer-based approaches, followed by the study's conceptual framework and hypothesis.

2.1. Deep learning

Deep learning is a subfield of machine learning that utilizes multi-layered artificial neural networks to extract hierarchical representations from raw data automatically [5]. Each layer in the network transforms the input into increasingly abstract features, enabling the model to capture complex and nonlinear relationships that conventional machine learning often fails to address effectively. Unlike traditional approaches that rely heavily on handcrafted features, deep learning learns directly from large volumes of data, thereby enhancing its ability to generalize to unseen inputs.

Deep learning approaches can broadly be classified into four categories. Supervised learning encompasses methods such as Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). CNNs are widely used in computer vision tasks due to their ability to capture spatial patterns. In contrast, RNNs are effective for processing sequential data, such as natural language and time series. Unsupervised learning encompasses Auto-Encoders, Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), and Generative Adversarial Networks (GANs), which are designed to uncover latent structures in unlabeled data. Reinforcement learning is generally divided into value-based methods, such as Deep Q-Learning, and policy-based methods, such as policy gradient algorithms, which have shown success in robotics and autonomous systems. Finally, hybrid models integrate elements from these categories to combine their strengths, enabling more robust and adaptive solutions to real-world challenges [6].

2.2. Transformer Architecture

The Transformer is a deep neural network architecture designed for efficient sequence modeling. It replaces recurrence and convolution with a self-attention mechanism, enabling parallel computation and effective modeling of both short- and long-range dependencies [7]. In our context, this capability is critical because malware binaries often contain non-contiguous and obfuscated byte patterns that require global context to detect reliably [8] .

https://doi.org/10.31436/iiumej.v27i1.4009

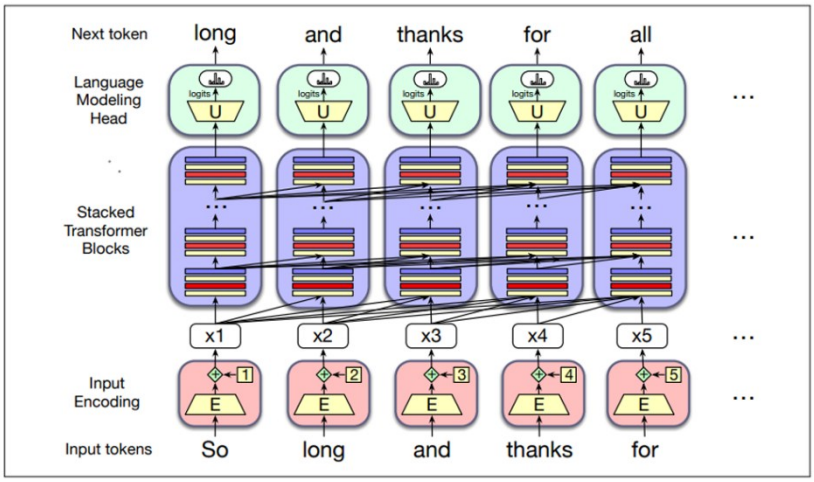

Figure 1. Transformer architecture [7]

As shown in Figure 1, the standard architecture comprises an encoder that maps the input sequence to contextual representations and a task head that uses these representations for prediction. For sequence-to-sequence tasks, the head is a decoder for malware classification at the byte level; it is typically a classification head attached to the encoder outputs. The stack includes input embeddings plus positional encodings, repeated layers of multi-head self-attention and position-wise feedforward networks, with residual connections and layer normalization to ensure stable optimization.

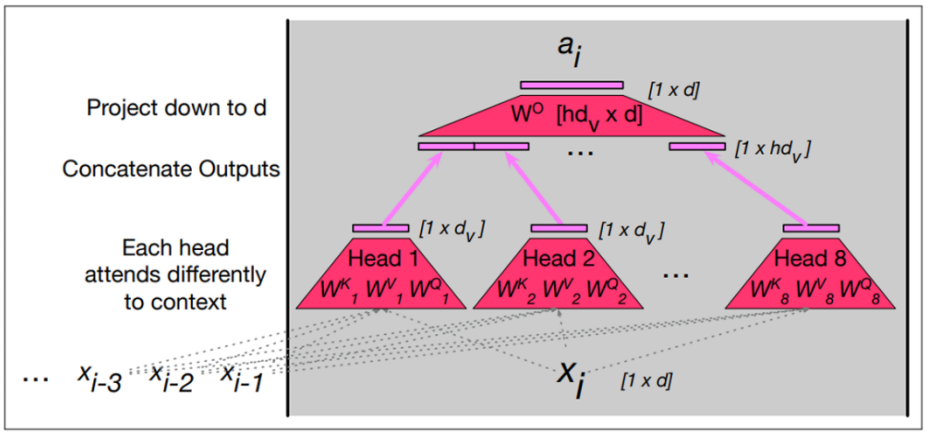

Figure 2. Multi-head attention mechanism [7]

The multi-head attention mechanism, as shown in Figure 2, enables the model to attend to different types of relationships in parallel, including local byte n-grams, long-range dependencies, and structural cues. This process produces richer contextual embeddings than a single attention head. This parallelism is particularly valuable for polymorphic/metamorphic malware, where discriminative evidence is dispersed across distant byte positions. Empirically, recent cybersecurity studies report that Transformer variants can learn directly from raw byte representations and improve detection accuracy and robustness [9-11].

A concise summary of the main components and their roles is provided in Table 1. Together, these components enable the proposed byte-level pipeline to capture global context without handcrafted tokenization, thereby aligning with our goal of building an adaptive, cross-platform detector.

Table 1. Key components of the Transformer architecture.

Component	Function
Input & Positional Embeddings	Map bytes to dense vectors and inject sequence order information.
Multi-Head Attention	Capture diverse contextual relationships across tokens in parallel.
Feedforward Network	Transform attention outputs through non-linear projections.
Residual Connections	Preserve information flow and stabilize training.
Layer Normalization	Improve numerical stability and accelerate convergence.

2.3. Malware

Malware, or malicious software, is among the most persistent and evolving threats in cybersecurity. Its primary objective is to compromise the confidentiality, integrity, and availability of digital systems, leading to severe consequences such as large-scale data breaches, financial losses, and the disruption of critical infrastructures [12]. Early generations of malware were relatively simple, typically relying on signature-based mechanisms such as viruses and worms. In contrast, modern variants have become increasingly sophisticated by adopting advanced evasion strategies. Polymorphism enables malware to alter its structure, thereby rendering signature-based detection ineffective continuously. Metamorphism rewrites malicious code into structurally distinct but functionally equivalent forms, complicating static inspection. In addition, obfuscation techniques deliberately conceal the program's logic flow to hinder both automated detection and manual analysis [13].

The growing complexity of digital ecosystems further exacerbates the challenge. Mobile platforms are frequently targeted due to the prevalence of third-party applications and insufficient vetting processes. Similarly, Internet of Things (IoT) devices, often deployed with limited computational resources and lacking regular security updates, significantly expand the attack surface for large-scale propagation. Cloud infrastructures introduce additional risks by centralizing sensitive data in shared environments, where a single compromise may cascade into systemic vulnerabilities [14]. Beyond these vectors, zero-day malware has emerged as a particularly critical concern. Zero-day attacks exploit previously unknown vulnerabilities, leaving defenders without available patches or signatures. Consequently, it is extremely difficult to detect them using traditional methods, underscoring the necessity of adaptive detection models capable of generalizing across unseen threats [15].

Malware can also be categorized according to its primary intent. Ransomware encrypts files and demands payment for decryption; spyware surreptitiously monitors and exfiltrates sensitive information, whereas botnets aggregate compromised devices into large-scale networks for coordinated attacks such as Distributed Denial-of-Service (DDoS) attacks. These diverse objectives highlight the breadth of modern malware threats and reinforce the importance of robust detection strategies that extend beyond static inspection [16].

3. RESEARCH METHODS

3.1. Research Approach

This study adopted a quantitative experimental approach to develop and evaluate a byte-level Transformer model for malware detection and classification. This approach was selected

because it enables objective measurement of model performance using numerical evaluation metrics. The research begins by identifying limitations in conventional malware detection methods, particularly rule-based and feature-engineering approaches, which struggle to address complex and evolving threats. To address these challenges, this study designs and trains a Transformer model directly on raw byte representations, thereby eliminating the need for handcrafted feature tokenization. The use of public datasets ensures transparency, reproducibility, and comparability with prior research, while rigorous evaluation using accuracy, precision, recall, and F1-score provides a comprehensive assessment of the model's effectiveness.

3.2. Sampling Method and Datasets

A purposive sampling strategy was employed to select datasets that contain both malware and benign files in raw binary format. This is essential for the proposed byte-level approach, which requires unprocessed data. Three benchmark datasets were employed:

- Malware Detection PE-Based Analysis [17]: 9,970 samples, including 8,970 malware and 1,000 benign files.
- MC-dataset-binary [18]: 11,480 samples, equally split between 5,740 malware and 5,740 benign files.
- Malware.zip [19]: 9,550 malware samples categorized into 12 malware families.

These datasets were selected to enable cross-dataset testing, thereby validating the generalization capability of the proposed model across diverse sources and malware categories.

3.3. Proposed Model Architecture

The proposed malware detection system is implemented as a Byte-Level Transformer that processes raw binary files without requiring handcrafted feature extraction. The model pipeline comprises several sequential components, each transforming raw byte sequences into discriminative features for classification. Figure 3 presents a schematic of the proposed model architecture, outlining the processing flow from raw binary input to the final classification output.

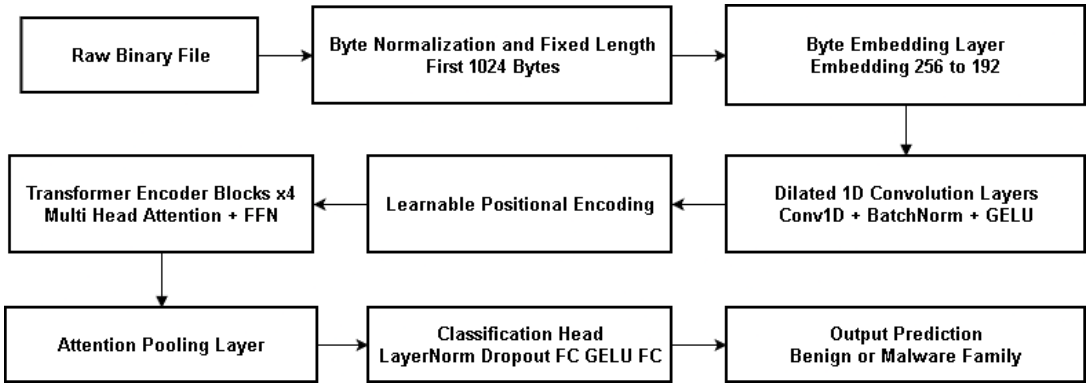

Figure 3. Architecture of the proposed byte-level transformer model for malware detection.

3.3.1. Input Preprocessing (byte normalization)

All executable samples, whether benign or malicious, are normalized to a fixed sequence length of 1,024 bytes (1 KB). Files shorter than this length are zero-padded, while longer files are truncated. Each byte (0–255) is treated as a discrete token, ensuring uniform tensor dimensions and efficient mini-batch processing on the GPU. The selection of 1,024 bytes is

https://doi.org/10.31436/iiumej.v27i1.4009

informed by prior studies on malware at the byte level, which use 1 KB blocks as a practical balance between information coverage and computational efficiency [20]. Following this rationale, we initially evaluated a sampling strategy that extracted byte segments from multiple 1 KB regions across each file. However, this approach yielded lower classification performance in our Transformer-based model. We attribute this behavior to disrupted sequential continuity, which is essential for attention-based architectures to learn meaningful contextual dependencies. Consequently, this study uses the first 1,024 bytes from each binary file. This choice preserves a contiguous byte sequence and yields more stable, consistent performance across all evaluated datasets.

3.3.2. Byte Embedding Layer

A trainable embedding matrix is implemented using the PyTorch module nn.Embedding(256, d), where d = 192 represents the embedding dimension. Each raw byte value is mapped into a 192-dimensional dense vector. This representation allows the model to learn semantic relationships among byte values, similar to word embeddings in NLP tasks. The embedding output has the shape [batch_size × 1024 × 192].

3.3.3. Input Preprocessing (convolution)

To capture local dependencies in the byte sequence, two one-dimensional convolutional layers with dilation are applied:

- nn.Conv1d(d, d, kernel_size=8, stride=4, dilation=2)
- Each convolution is followed by BatchNorm1d(d) and a non-linear activation (GELU).

Dilated convolution exponentially increases the receptive field, enabling the model to detect repeating byte patterns across varying distances, which is crucial for detecting malware with obfuscated payloads.

3.3.4. Positional Encoding

Because byte order is critical, a learnable positional embedding is introduced: nn.Parameter(torch.randn(1, seq_length, d)) with seq_length = 1024. This positional tensor is added element-wise to the byte embeddings after convolution and normalization. Unlike fixed sinusoidal encodings, learnable embeddings adapt to dataset-specific structural patterns.

3.3.5. Transformer Encoder Blocks

The backbone of the architecture is a stack of four Transformer encoder layers, each defined by nn.TransformerEncoderLayer(d_model=192, nhead=8, norm_first=True).

- Multi-Head Self-Attention: Each head processes queries, keys, and values in parallel, enabling the model to capture long-range dependencies across the byte sequence.
- Residual Connections and Pre-Layer Normalization: Implemented to stabilize training and accelerate convergence.

These encoder blocks convert the sequence of embeddings into a context-aware representation where relationships between distant byte patterns are explicitly modeled.

https://doi.org/10.31436/iiumej.v27i1.4009

3.3.6. Attention Pooling Layer

Since Transformer outputs retain the full sequence dimension, an attention pooling mechanism aggregates the sequence into a single feature vector. A learnable weight vector computes attention scores over all sequence positions:

$$\alpha_i = \text{softmax}(W \cdot h_i) \quad (1)$$

where h_i is the hidden representation of byte i. The final vector is obtained by weighted summation, emphasizing critical regions of the byte sequence.

3.3.7. Classification Head

In the final stage, the aggregated feature vector from the attention pooling layer is first normalized using Layer Normalization to ensure stable input distributions. A dropout layer with a probability of 0.4 is then applied to reduce overfitting by randomly deactivating a portion of the neurons during training. The normalized and regularized vector is subsequently passed through a fully connected layer (Linear(d, d/2)) followed by a Gaussian Error Linear Unit (GELU) activation function, which introduces non-linearity and enhances the model's capacity to capture complex patterns. Another dropout operation is applied at this stage to further improve the model's robustness. Finally, the processed representation is mapped to the output space through a linear transformation (Linear(d/2, num_classes)), producing the classification logits that represent the predicted categories, i.e., specific malware families or benign files.

3.4. Training Strategy

The training strategy of the proposed byte-level Transformer was designed to maximize generalization and prevent overfitting through a combination of data augmentation, optimization, and stability mechanisms. Data augmentation was applied using random byte masking and MixUp to increase sample diversity and improve robustness to noise. For optimization, the AdamW algorithm with weight decay was employed, along with a two-phase learning rate schedule: an initial warmup stage to stabilize convergence, followed by cosine annealing to gradually decay the learning rate.

To further enhance training stability, Stochastic Weight Averaging (SWA) was applied during the later epochs to improve generalization. Early stopping with a patience of ten epochs was used to terminate training when validation loss no longer improved. In addition, cross-entropy loss with label smoothing and adaptive class weights was adopted to mitigate overconfident predictions and address class imbalance across malware families.

All experiments were conducted on an NVIDIA Tesla T4 GPU, based on the NVIDIA Turing architecture and equipped with 2,560 CUDA cores and 320 Tensor Cores. The GPU provides 16 GB of GDDR6 memory, enabling efficient training and inference for medium-scale Transformer models with fixed-length byte inputs. Support for mixed-precision computation further accelerated training while maintaining numerical stability. Overall, the chosen hardware configuration allowed the model to converge efficiently within a practical training time and supported fast inference, demonstrating the feasibility of the proposed approach for real-world deployment.

3.5. Comparison with Transformer-Based Malware Detection Models

Transformer-based approaches to malware detection differ primarily in their input representations and architectures. MalBERT, for example, is designed for Android malware analysis and operates on source-code-derived textual artifacts using a pretrained BERT model, requiring platform-specific preprocessing, such as decompilation [21]. In contrast, the

proposed model operates directly on contiguous raw byte sequences from executable binaries and is trained from scratch without relying on pretrained language models or handcrafted feature extraction. By integrating dilated convolution layers with Transformer encoders, the proposed architecture captures both local byte-level patterns and long-range dependencies in a unified end-to-end framework. Due to these fundamental differences in platform and input modality, direct numerical comparison is not feasible. The key architectural distinctions between the proposed approach and MalBERT are summarized in Table 2.

Table 2. Comparison of Transformer-Based Malware Detection Models

Aspect	Proposed Model	MalBERT
Target domain	Windows executable malware (raw binaries)	Android malware
Input representation	Raw byte sequence (first 1,024 bytes)	Source code-derived text artifacts
Feature engineering	None (end to end)	Requires decompilation and text extraction
Architecture	Dilated CNN + Transformer	BERT-based Transformer
Pre-training	No (trained from scratch)	Yes (large-scale LM pretraining)
Key characteristic	Lightweight, contiguous byte modeling	Strong NLP transfer learning
Comparability	Directly applicable to PE/raw bytes	Different platform & modality

4. RESULTS AND DISCUSSION

This section presents the experimental results of the proposed byte-level Transformer model on three publicly available malware datasets. Performance is evaluated using standard classification metrics, namely accuracy, precision, recall, and F1-score. To provide a comprehensive analysis, the results are presented in both tabular and graphical formats, followed by a detailed discussion of each dataset.

4.1. Results on the First Dataset

The first dataset, Malware Detection PE-Based Analysis, comprises 9,970 samples: 8,970 malware and 1,000 benign files. Table 3 summarizes the classification results across six categories.

Table 3. Classification results on the first dataset.

Class	Precision	Recall	F1-Score	Support
Benign	0.99	0.95	0.97	200
Loker	0.81	0.94	0.87	66
Mediyes	1	0.99	1.00	290
Winevesoc	1	1	1	880
Zbot	0.99	0.99	0.99	420
Zerroaccess	1	0.99	0.99	138
Accuracy			**0.99**	**1994**

As shown in Table 3, the model achieved an overall accuracy of 99%, with near-perfect precision, recall, and F1-score for categories such as Mediyes, Winwebsec, and Zeroaccess. However, the Locker class achieved lower scores (precision = 0.81, recall = 0.94), attributable to the smaller number of training samples, which made the feature representation less distinctive. To further illustrate the performance, Figure 4 presents the confusion matrix for this dataset.

https://doi.org/10.31436/iiumej.v27i1.4009

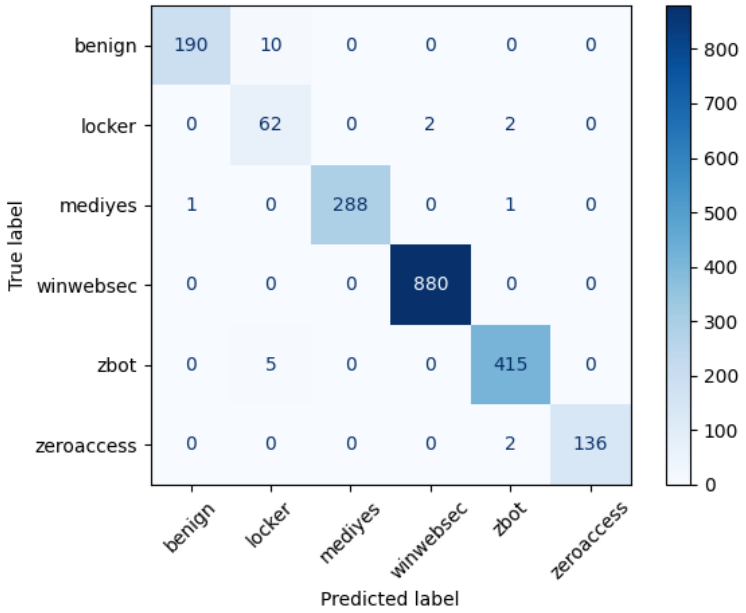

Figure 4. Confusion matrix for the first dataset

The confusion matrix indicates that Winwebsec was perfectly classified, with all samples correctly assigned to their true categories. In contrast, the Locker class exhibits the highest misclassification rate among the six classes, with several instances misclassified as Winwebsec and Zbot. This pattern suggests that Locker shares specific byte-level characteristics with these families, resulting in overlap in learned representations. Such behavior highlights the need for further refinement in feature extraction or modeling strategies to improve separability in future work.

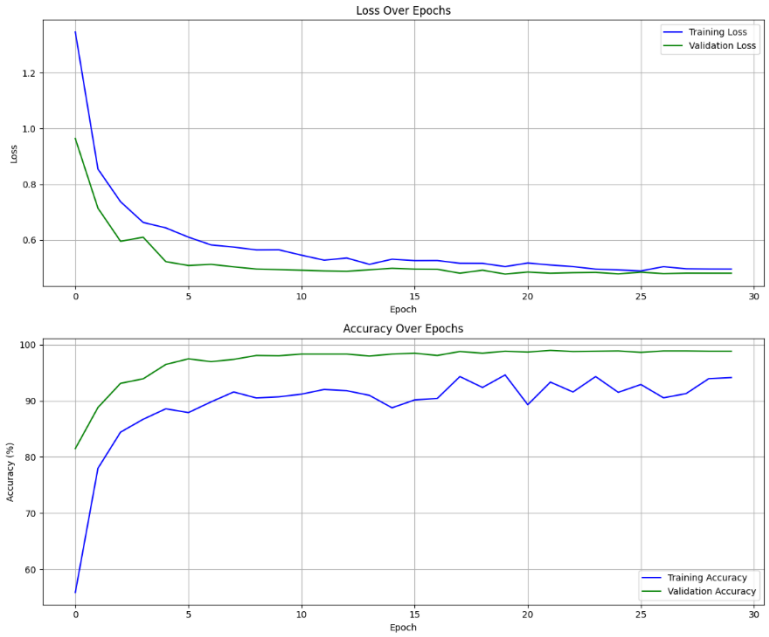

Figure 5. Training and validation performance on the first dataset

Figure 5 shows that the model converges stably, with both training and validation losses decreasing consistently and validation accuracy remaining above 98% across the later epochs. At epoch 30, the model achieved a training accuracy of 94.16% and a validation accuracy of

https://doi.org/10.31436/iiumej.v27i1.4009

98.85%. Early stopping was triggered when no further improvement in validation loss was observed. The full training process required 5 minutes and 1.41 seconds (10.05 seconds per epoch). Following the application of Stochastic Weight Averaging and the restoration of the best-performing checkpoint, the model achieved a peak validation accuracy of 99.00%, demonstrating strong generalization across the complete architectural and regularization setup.

Despite overall high performance, the class distribution in this dataset is imbalanced, which affects minority classes such as Locker, as evidenced by lower precision than for majority classes. To mitigate this issue, adaptive class weighting, Mixup embedding, and label smoothing were employed during training to improve robustness for underrepresented classes. In practical deployment, false positives may lead to unnecessary alerts for benign files, increasing operational overhead, whereas false negatives pose a more critical risk by allowing malware to evade detection. The observed training behavior indicates that the proposed model prioritizes stable generalization while maintaining sensitivity to malicious samples, which is essential for real-world malware detection systems.

To assess the contribution of these components, the following analysis compares the full model with a simplified version that excludes Dilated CNN layers, Attention Pooling, Mixup Embedding, and SWA. This ablation study highlights the performance degradation that occurs when these elements are removed, clarifying their role in enhancing accuracy, convergence dynamics, and overall robustness. The impact of omitting these mechanisms is illustrated in Figures 6 and 7.

Figure 6 shows that the ablation model exhibits higher misclassification rates across several classes than the full model. In the ablation results, errors are more widely distributed, particularly within the benign, Locker, Mediyes, and Zbot classes, which indicate weaker class separation. For example, the Locker class receives fewer correct predictions and exhibits additional confusion with benign, Mediyes, and Zbot, whereas the full model maintains a cleaner diagonal pattern with fewer mistakes. The benign and Mediyes classes also exhibit more scattered errors in the ablation version, whereas the full model maintains greater consistency and higher accuracy.

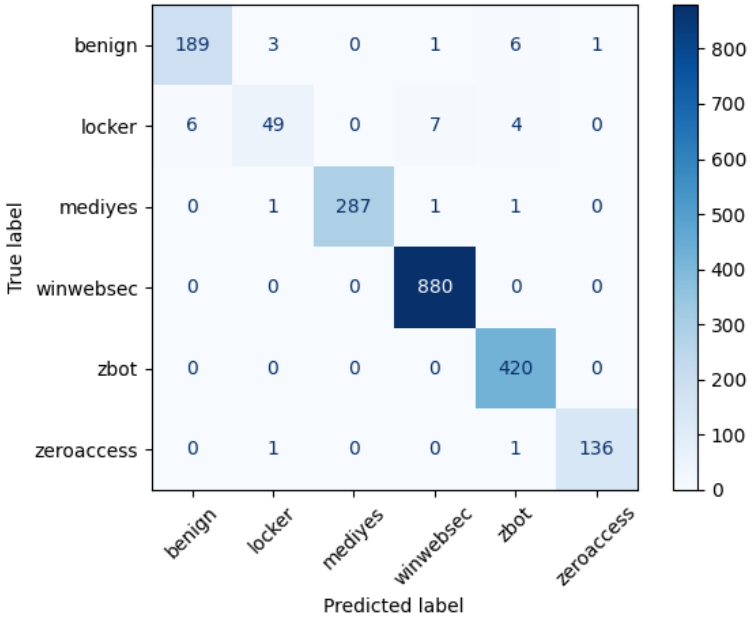

Figure 6. Confusion Matrix for the First Dataset Under Ablation (Without Dilated CNN, Attention Pooling, Mixup, or SWA)

https://doi.org/10.31436/iiumej.v27i1.4009

These differences indicate that the components removed in the ablation configuration reduce the model's ability to learn distinctive features, resulting in greater class overlap. In contrast, the full model exhibits more stable and reliable performance, as evidenced by a sharper diagonal structure and lower misclassification rates.

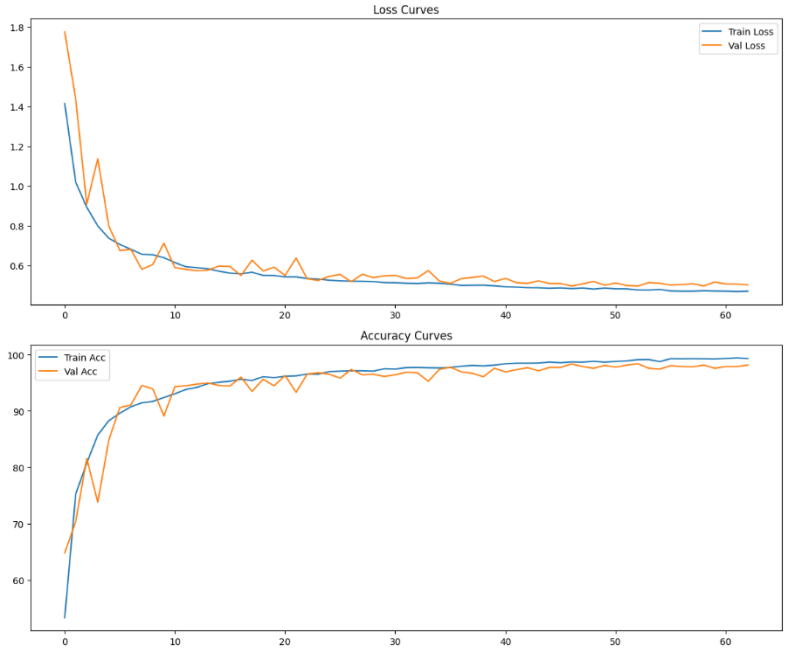

Figure 7. Training and Validation Performance on the first dataset under the Ablation Setting.

Figure 7 shows that the ablation model converges successfully, with both training and validation losses decreasing steadily, and the final validation accuracy reaching 98.35%. However, when compared with the full model, the impact of removing Dilated CNN layers, Attention Pooling, Mixup Embedding, and Stochastic Weight Averaging becomes evident. The full model achieves a best validation accuracy of 99.00 percent in only 30 epochs. In contrast, the ablation model requires 63 epochs and more than two hours of training. The average epoch duration also differs substantially: 134.68 seconds for the ablation model and 10.05 seconds for the complete configuration.

These differences indicate that the removed components play an important role in improving the efficiency and stability of the learning process. In the full model, Mixup and SWA contribute to smoother optimization and stronger generalization, while the Dilated CNN layers and Attention Pooling enhance feature extraction and class separability. As a result, the full model converges more quickly, maintains a cleaner accuracy trajectory, and produces more accurate predictions. The ablation results, therefore, confirm that each component contributes measurable benefits to performance, robustness, and training efficiency.

4.2. Results on the Second Dataset

The second dataset, MC-dataset-binary, comprises 11,480 samples, evenly split between malware and cleanware. Table 4 reports the binary classification performance. As indicated in Table 4, the model achieved an overall accuracy of 92%. Malware samples achieved higher recall (0.94), whereas cleanware samples achieved higher precision (0.95). This imbalance indicates that the model is more effective in detecting malware but occasionally misclassifies benign samples as malicious. The confusion matrix in Figure 8 provides more detail.

https://doi.org/10.31436/iiumej.v27i1.4009

Table 4. Classification results on the second dataset.

Class	Precision	Recall	F1-Score	Support
Cleanware	0.95	0.91	0.92	1148
Malware	0.91	0.94	0.92	1148
Accuracy			**0.92**	**2296**

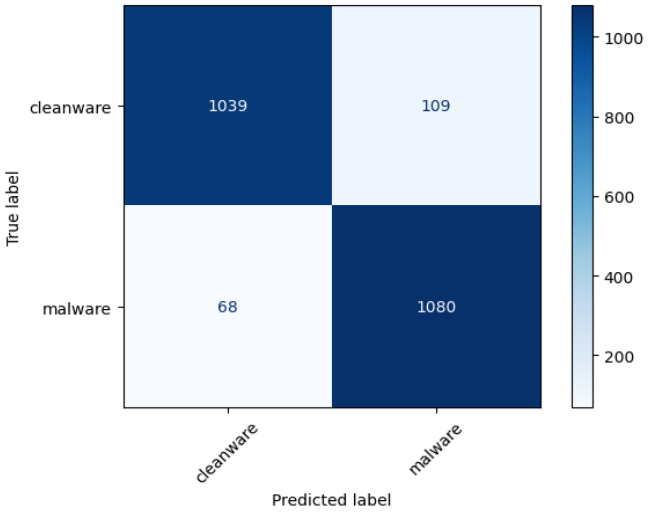

Figure 8. Confusion matrix for the second dataset.

As shown in Figure 8, most cleanware samples were correctly classified, with 1,039 of 1,148 samples predicted accurately. Nevertheless, 109 cleanware samples were misclassified as malware. Conversely, the model misclassified 68 malware samples as cleanware, while 1,080 were correctly identified. These results indicate that the model exhibits slightly greater sensitivity for malware detection, reducing the likelihood of false negatives but still yielding a moderate number of false positives. Training and validation performance are illustrated in Figure 9.

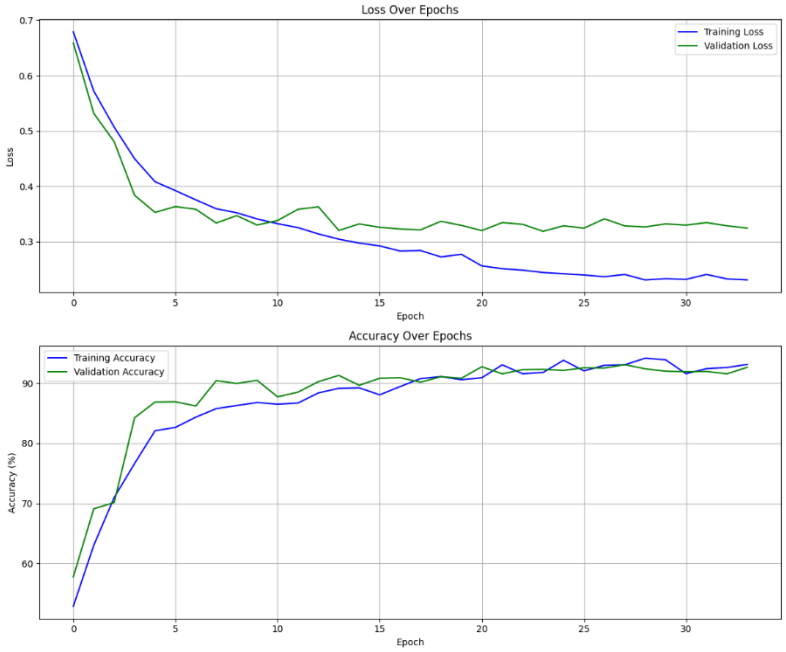

Figure 9. Confusion matrix for the second dataset

https://doi.org/10.31436/iiumej.v27i1.4009

Figure 9 shows that the model converges stably, with losses decreasing and validation accuracy maintaining consistent performance around 92-93%. At epoch 34, the model recorded a training accuracy of 93.12% and a validation accuracy of 92.64%. Early stopping was triggered after ten epochs without improvement in validation loss, resulting in a total training time of 6 minutes and 28.76 seconds (11.43 seconds per epoch). After applying Stochastic Weight Averaging and restoring the checkpoint with the lowest validation loss, the model achieved a best validation accuracy of 93.07%, indicating reliable generalization on the second dataset.

4.3. Results on the Third Dataset

The third dataset, Malware.zip, contains 9,550 samples distributed across 12 malware families with imbalanced class sizes. Table 5 shows that the overall accuracy reached 94%, with perfect predictions for Coinhive and near-perfect scores for Emotet and Icedid. Among all families, Razy exhibited the lowest performance, with a precision of 0.85 and an F1-score of 0.88, indicating a higher rate of misclassification compared to other classes. The recall for Razy remained relatively high at 0.92, suggesting that most Razy samples were correctly identified, although the model exhibited a high rate of false positives. The confusion matrix in Figure 10 provides further insight into these misclassification patterns.

Table 5. Classification results on the third dataset.

Class	Precision	Recall	F1-Score	Support
Coinhive	1	1	1	81
Emotet	0.94	0.98	0.96	181
Fareit	0.97	0.95	0.96	317
Flystudio	0.97	0.93	0.95	73
Gafgyt	0.95	0.97	0.96	297
Gandcrab	0.97	0.92	0.95	78
Icedid	1	0.97	0.98	92
Lamer	0.95	0.85	0.9	138
Mepawo	0.88	0.96	0.92	164
Mirai	0.96	0.95	0.96	288
Ramnit	0.95	0.93	0.94	67
Razy	0.85	0.92	0.88	134
Accuracy			**0.94**	**1910**

As illustrated in Figure 10, most malware families were classified accurately, with high counts along the diagonal of the confusion matrix. Misclassifications occurred between several related categories. For instance, Fareit samples were occasionally predicted as Ramnit, and Gafgyt was sometimes misclassified as Mirai. These patterns suggest shared byte-level characteristics among certain families, which may contribute to overlapping predictions. Figure 11 presents the corresponding training and validation curves for this experiment.

https://doi.org/10.31436/iiumej.v27i1.4009

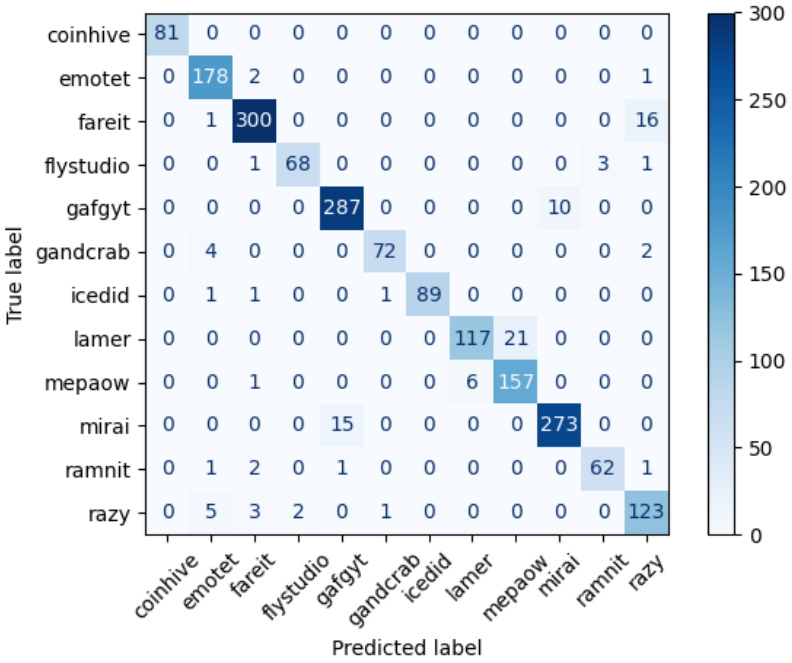

Figure 10. Confusion matrix for the third dataset

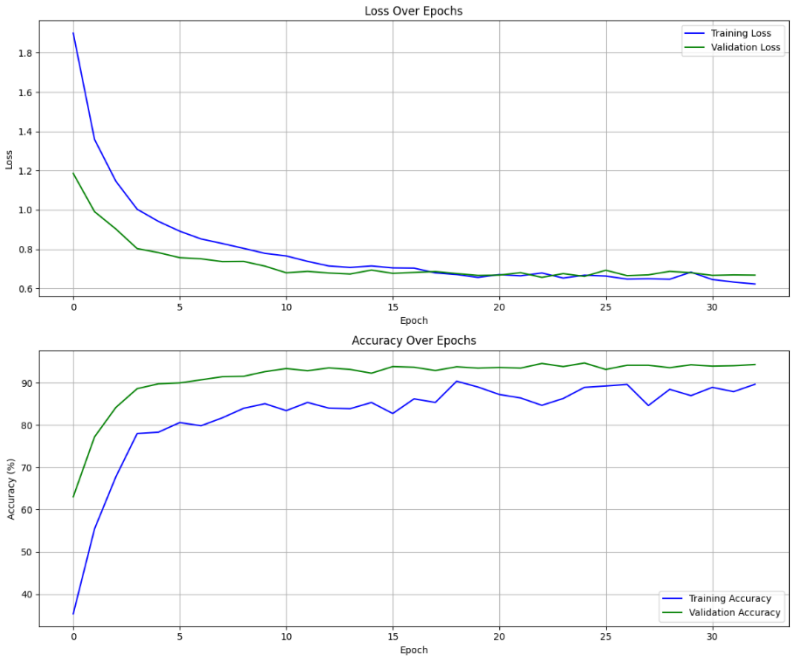

Figure 11. Training and validation performance on the third dataset

Figure 11 shows that the model converges stably on the third dataset, with training and validation losses decreasing consistently and validation accuracy remaining within the 93-95% range. At epoch 33, the model achieved a training accuracy of 89.67% and a validation accuracy of 94.35%. Early stopping was then triggered due to stagnation in validation loss. The full training process required 5 minutes and 12.12 seconds (9.46 seconds per epoch). After applying Stochastic Weight Averaging and restoring the best checkpoint, the model achieved a peak validation accuracy of 94.71%, demonstrating strong and reliable generalization.

https://doi.org/10.31436/iiumej.v27i1.4009

The consistently higher validation accuracy can be attributed to the regularization mechanisms used during training, such as dropout, Mixup embedding, and SWA, which intentionally increase training difficulty and slightly suppress training accuracy while enhancing generalization. Additionally, an imbalanced dataset may yield validation subsets with more representative samples for certain malware families, leading to higher validation performance. This pattern is common in regularized deep learning models and is not necessarily indicative of overfitting.

5. CONCLUSION

This study has successfully designed and implemented a byte-level Transformer model for effective and adaptive malware detection and classification. The proposed approach eliminates the need for handcrafted feature extraction by directly learning from raw byte representations through multi-head self-attention. Experimental evaluations conducted on three publicly available datasets demonstrated strong performance, with classification accuracies of 99% on the Malware Detection PE-Based Analysis dataset, 92% on the MC-dataset-binary dataset, and 94% on the Malware.zip dataset, respectively. These results confirm the effectiveness of byte-level representation in capturing distinctive malware characteristics, enabling the model to generalize across different families and file structures.

The analysis further revealed that malware categories with abundant and distinctive byte patterns, such as Winwebsec, Mediyes, Zeroaccess, Coinhive, and Icedid, achieved near-perfect classification scores. However, minority classes, or those with ambiguous features, such as Locker and Razy, showed comparatively lower performance, underscoring the challenges posed by imbalanced datasets. Beyond experimental validation, the successful deployment of a prototype web-based application demonstrated that the model is not only accurate but also practical for real-world malware detection, providing fast, adaptive, and efficient inference.

Overall, this research highlights the potential of byte-level Transformer architectures as a universal and flexible framework for malware detection. The findings provide direct contributions to advancing deep learning-based cybersecurity solutions and open avenues for future work on handling data imbalance, hybrid feature integration, cross-platform generalization, and zero-day malware detection.

6. LIMITATIONS AND FUTURE WORK

Despite the promising results obtained in this study, several limitations should be acknowledged, which also indicate potential directions for future research.

First, the proposed byte-level Transformer model operates on a fixed input length of 1,024 bytes, analyzing only the initial contiguous segment of each binary file. Although this design choice has been empirically shown to provide stable, consistent performance across all evaluated datasets, it may limit the model's ability to capture malicious patterns beyond the initial byte region, particularly in larger or more complex binaries.

Second, although the model demonstrated strong generalization across three publicly available datasets, no explicit zero-day malware evaluation was conducted. The experimental setup followed a supervised learning paradigm using labeled datasets, without isolating unseen malware families as a dedicated zero-day testing scenario. Consequently, the effectiveness of the proposed model against strictly zero-day malware samples was not directly assessed in this study.

Third, dataset imbalance remains an inherent limitation, especially in multi-class classification tasks. As observed in the experimental results, minority malware families, such as Locker and Razy, exhibited lower precision and F1 scores than the majority classes. Although mitigation strategies, including adaptive class weighting, Mixup embedding, and label smoothing, were applied during training, class imbalance continued to influence the classification performance of underrepresented classes.

Finally, the current study did not evaluate adversarial robustness. The proposed model was assessed under standard classification settings and was not tested against adversarially crafted binaries or evasion-based attacks that may target deep learning-based malware detectors. As a result, the model's robustness under adversarial conditions remains to be investigated.

Future work may address these limitations in several directions. Variable-length input processing or sliding-window mechanisms could be explored to enable the analysis of larger binary regions while preserving sequential continuity. In addition, dedicated zero-day evaluation protocols, such as family-wise holdout or temporal split testing, should be incorporated to assess generalization to previously unseen malware more rigorously. Further investigation into advanced imbalance handling strategies and hybrid feature integration may improve performance for minority classes. Moreover, evaluating adversarial robustness and incorporating adversarial training techniques are essential to enhance the proposed model's resilience in real-world deployment scenarios.

REFERENCES

[1] BSSN, “LANSKAP KEAMANAN SIBER INDONESIA 2024,” Jan. 2025.

[2] K. Khalda and D. K. Wibowo, “Malware Behavior Analysis Using Static and Dynamic Analysis Approaches,” Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi, vol. 4, no. 1, pp. 1–8, Jan. 2025, doi: 10.20885/snati.v4.i1.1.

[3] M. V. Ngo, T. Truong-Huu, D. Rabadi, J. Y. Loo, and S. G. Teo, “Fast and Efficient Malware Detection with Joint Static and Dynamic Features Through Transfer Learning,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.13860

[4] M. Abdur Rahman, G. A. Francia Iii, H. Shahriar, G. Francia III, E. El-Sheikh, and S. Iqbal Ahamed, “A Novel Approach to Fine-tune BERT using Non-Text Features for Enhanced Ransomware Detection,” 2025, doi: 10.13140/RG.2.2.35576.15365.

[5] W. A. Salmon, “Learning in Quantum Mechanics,” 2024.

[6] T. Talaei Khoei, H. Ould Slimane, and N. Kaabouch, “Deep learning: systematic review, models, challenges, and research directions,” Nov. 01, 2023, Springer Science and Business Media Deutschland GmbH. doi: 10.1007/s00521-023-08957-4.

[7] J. Daniel and J. H. Martin, “Speech and Language Processing,” 2024.

[8] Y. Bai, J. Mei, A. Yuille, and C. Xie, “Are Transformers More Robust Than CNNs?,” Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.05464

[9] Y. Fan et al., “Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection,” in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, Aug. 2021, pp. 2831–2839. doi: 10.1145/3447548.3467168.

[10] A. Rahali and M. A. Akhloufi, “MalBERT: Using Transformers for Cybersecurity and Malicious Software Detection,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.03806

[11] T. L. Huoh, T. Miskell, O. Barut, Y. Luo, P. Li, and T. Zhang, “Malware Detection for Portable Executables Using a Multi-input Transformer-Based Approach,” in 2024 International Conference on Computing, Networking and Communications, ICNC 2024, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 778–782. doi: 10.1109/ICNC59896.2024.10556067.

[12] 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.

[13] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware Detection Issues, Challenges, and Future Directions: A Survey," Sep. 01, 2022, MDPI. doi: 10.3390/app12178482.

[14] T. L. Wan et al., "Efficient Detection and Classification of Internet-of-Things Malware Based on Byte Sequences from Executable Files," IEEE Open Journal of the Computer Society, vol. 1, pp. 262–275, 2020, doi: 10.1109/OJCS.2020.3033974.

[15] T. McIntosh, A. S. M. Kayes, Y. P. P. Chen, A. Ng, and P. Watters, "Ransomware Mitigation in the Modern Era: A Comprehensive Review, Research Challenges, and Future Directions," Dec. 31, 2022, Association for Computing Machinery. doi: 10.1145/3479393.

[16] M. Naseer et al., "Malware Detection: Issues and Challenges," in Journal of Physics: Conference Series, IOP Publishing Ltd, Apr. 2021. doi: 10.1088/1742-6596/1807/1/012011.

[17] A. P. Tuan, A. T. H. Phuong, N. V. Thanh, and T. N. Van, "Malware Detection PE-Based Analysis Using Deep Learning Algorithm Dataset," 2018, figshare. doi: 10.6084/m9.figshare.6635642.v1.

[18] E. de O. Andrade, "MC-dataset-binary," 2018, figshare. doi: 10.6084/m9.figshare.5995408.v1.

[19] A. Azab and M. Khasawneh, "MSIC: Malware Spectrogram Image Classification," IEEE Access, vol. 8, pp. 102007–102021, 2020, doi: 10.1109/ACCESS.2020.2999320.

[20] D. Morozovskii, K. Thummar, T. Halabi, and S. Ramanna, "Toward Efficient and Robust Deep Learning-based Malware Detection in Fog Computing," in 2021 International Symposium on Networks, Computers and Communications, ISNCC 2021, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ISNCC52172.2021.9615812.

[21] A. Rahali and M. A. Akhloufi, "MalBERT: Using Transformers for Cybersecurity and Malicious Software Detection," Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.03806