International Journal of Data and Network Science 10 (2026) 1377–1384

Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

# Lightweight malware detection model in a resource-limited environment using single-head attention with LSTM/GRU

**Mariam Al Ghamri[a*], Afaf Edinat[a], Mohammad Shehab[a*], Ibtisam Obaidat[a] and Ahmed E Fakhry[a]**

[a]*College of Information Technology, Amman Arab University, Amman 11953, Jordan*

**C H R O N I C L E**

*Article history*:
Received January 19, 2026
Received in revised format March 14, 2026
Accepted March 25 2026
Available online
March 25 2026

*Keywords*:
*Malware threats*
*Lightweight malware detection*
*LSTM/GRU*
*Cybersecurity*
*Resource-constrained systems*

**A B S T R A C T**

The rising malware threats in the past years are due to the proliferation of smart devices and resource-constrained systems, such as the Internet of Things (IoT) and mobile devices. This phenomenon creates great difficulties for conventional defense mechanisms, which frequently lag behind because of changes in the malware landscape. In this work, a lightweight and powerful network combining a single-head attention mechanism with LSTM or GRU for this task, based on the CIC-MalMem-2022 dataset, is introduced. The focus is on learning temporal features retrieved from memory data, considering model efficiency for resource-constrained devices. Results indicate that the generated model can retain an accuracy of 92%, and it can save training time by 30%, which is highly beneficial for time-critical tasks and efficient resource usage in real-life applications. This model contributes to the enhancement of cybersecurity by offering good practice against the growing range of new threats in today's technology-advanced environments.

© 2026 by the authors; licensee Growing Science, Canada.

## 1. Introduction

Malware has become one of the most significant threats to information security due to its rapidly evolving nature and the constant emergence of new variants that can bypass traditional defense mechanisms. These mechanisms, such as signature-based detection systems that rely on static rules and known threat patterns, struggle to identify polymorphic, metamorphic, and zero-day attacks. The limitations of these classical approaches are well-documented, often failing to capture the complex behavioral traits of advanced malware (Ferdous & Islam, 2024). Moreover, the proliferation of smart devices and the Internet of Things (IoT) has exacerbated the problem of effective malware detection, considering that resource-constrained environments require the application of protection schemes that are both resource-efficient and able to perform real-time processing. Past studies have established that deep learning approaches, such as convolutional and recurrent neural networks, are able to utilize the features of the provided datasets more efficiently compared to traditional machine learning approaches. However, these approaches require large computational resources, thus limiting their application in edge computing (Al-Ghanem et al., 2025). Practical observations reveal that organizations face significant challenges in securing their systems, as many existing approaches either demand excessive computational resources or yield high false-positive rates. These issues limit their effectiveness in real-world scenarios where speed and efficiency are critical. To address these challenges, this work presents a novel and computationally efficient malware detection model. This model combines a single-head attention mechanism with Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, harnessing the strengths of recurrent models to capture temporal dependencies while reducing complexity through selective attention. The focus is on analyzing memory- derived behavioral features, which have proven to be strong indicators of malicious activity and help overcome the limitations of static analysis techniques, especially in detecting obfuscated or memory-resident threats (Zhang et al., 2022).

The proposed solution is a lightweight architecture for fast inference which is useful for low-power resources not consuming much memory and energy who need low latency decision making. The goal of this research is to improve malware detection

* Corresponding author
E-mail address m.alghamry@aau.edu.jo (M. A. Ghamri) m.shehab@aau.edu.jo (M. Shehab)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)
© 2026 by the authors; licensee Growing Science, Canada.
doi: 10.5267/j.ijdns.2026.3.009

performance on resource-constrained end devices that incorporate the attention mechanism that improves detection performance by capturing relevant temporal dependencies. Also, standardized benchmark datasets are used to evaluate the model for a fair comparison with existing methods and reliable outputs. An important contribution to this work is a compact hybrid framework combining single-head attention with LSTM/GRU models for memory analysis. The experimental tests show that our model attains high accuracy with a specially reduced cost for all instances which is another contribution of this work. Testing on the CIC-MalMem-2022 dataset confirms the usability of the model for real-world deployment, balancing high detection performance and efficiency on constrained platforms.

The subsequent sections of this paper are outlined below. Section 2 provides an overview of related works. Section 3 delves into the specifics of the proposed methods. This is succeeded by the presentation of results experimental in Section 4. Lastly, Section 5 encapsulates conclusions.

## 2. Literature review

Significant researchers' interest has been given to advancements in malware detection systems in recent times, especially in developing lightweight systems that can be implemented in resource-constrained devices. Various studies have employed traditional machine learning approaches like Support Vector Machines (SVM) and Random Forest. However, these approaches heavily rely on feature extraction, which might not be sufficient in dealing with evolving malware. The memory analysis has been recognized as a powerful tool in understanding malicious activities and has become a prominent feature in the detection of Trojan horse malware. In the paper presented by Abualhaj and Al-Khatib (2024), the effectiveness of the features of the memory image in the detection and classification of the malware using various machine learning models is discussed. Specifically, the study focused on the decision tree classifier using the CIC-MalMem-2022 dataset. The results revealed that DT achieved a detection accuracy of 99.96%, which is more than Gradient Boosted Trees, Naive Bayes, Linear SVM and KNN, outperforming them. Most classifiers produced good results, though there could be variations in performance; nonetheless, memory data proved useful in detecting Trojan Horse events. Abualhaj et al. (2025) proposed a malware forensic examination, which integrated the Gray Wolf Optimization algorithm with several machine-learning classifiers including Naive Bayes, Random Forest (RF), Decision Trees, Support Vector Machines, K-Nearest Neighbors. Research that used the memory-based CIC-MalMem-2022 dataset and Python-based model development showed the promising potential GWO-enhanced models offer for memory-resident malware analysis. The RF classifier achieved top performance: 75.6% accuracy in a 16-class multiclass classification. Meanwhile, our proposed performance reached 99.2% in binary classification. Furthermore, the four-class classification accuracy of the model on same dataset is 86.34%. The sixteen-class accuracy achieved is 75.64%. The sixteen-class accuracy achieved by the model is 75.64%. The new model performs better than existing methods. The convergence of GWO with RF can significantly improve malware forensic investigation as well as detection accuracy with varying levels of classification complexities.

Akinshola-Awe et al. (2024) Built a machine learning framework to enhance malware detection and classification, solving the limitations of traditional signature-based antimalware tools that are ineffective against polymorphic and zero-day attacks. They applied feature selection techniques such as Principal Component Analysis coupled with Chi-Square, along with Grid Search for hyperparameter tuning. The study analysed performance on two datasets, CIC-MalMem and another dataset, using Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Decision Tree models. CIC-MalMem is a balanced dataset with benign and malicious samples. Results showed that SVM achieved the most accuracy of 100% on CIC-MalMem, whereas KNN and Decision Tree achieved accuracy of 99% and 98.64% respectively with KNN producing zero false positive. A second dataset was tested using SVM and KNN with 96% and 97.7% accuracy respectively while Decision Tree performed poorly achieving 70%. This research demonstrates that by optimising the hyperparameters and combined methods of feature selection the detection rate can be improved significantly especially with the balanced dataset. Furthermore, it is suggested to try out the deep learning and ensemble method as they have shown superior performance in other studies for malware detection on larger datasets architectures.

Akinshola-Awe et al. (2025) presented an enhanced malware detection and classification framework that addresses limitations in traditional antimalware systems, particularly their inability to detect polymorphic and zero-day threats. Their study employed K-Nearest Neighbor, Decision Tree, and Support Vector Machine classifiers trained on a balanced dataset, and applied Grid Search with cross-validation to optimize model hyperparameters. Using standard evaluation metrics, the optimized models demonstrated substantial improvements, with the Support Vector Machine achieving 100% accuracy, only two false positives, and zero false negatives. The findings highlight the effectiveness of hyperparameter optimization in improving detection performance and reducing misclassification rates in machine learning-based malware analysis. Additionally, Alashjaee et al. (2025) proposed a new intrusion detection system using a combination of CNN, LSTM, and the attention mechanism. In this model, CNN and LSTM are used together for better feature representation in the context of network security. The attention mechanism plays a vital role in this model. Evaluation of the model using the NSL-KDD and Bot-IoT datasets showed that the model achieved between 94.8% and 97.5% in terms of accuracy. Moreover, the model showed significant improvements in F1 score and Matthews Correlation Coefficient. Ablation study of the model proved the significant contribution of every component of the model, especially the attention mechanism. Performance evaluation of the model showed that the model is capable of processing data in real-time, with latency less than 35 ms. Furthermore, Alhady and Abdulbasit (2024) suggested a real-time AI-based security system that includes the integration of malware,

phishing, and breach detection using a combination of machine and deep learning models. The authors have proposed a system that uses a combination of CNN and LSTM models for static and dynamic malware analysis, the Transformer model for phishing email and URL classification, and unsupervised models like Isolation Forest for detecting unknown network threats. The proposed system has shown significant results using datasets like CIC-MalMem-2022, CICIDS-2017, and a phishing dataset. Reported results include an average F1-score of 98.2% for malware detection, 97.5% for phishing classification, and 96.8% for anomaly detection, demonstrating the effectiveness of consolidated AI-based systems in providing proactive and scalable cybersecurity protection. Ali et al. (2025a) developed MalwareVision, a deep learning framework to enhance malware classification to overcome challenges associated with signature-based detection, particularly for new and obfuscated variants. Using the Malimg dataset, which comprises 9,339 malware images from 25 families, the authors trained and tested a CNN model that scored 95.09% on the training and testing datasets, respectively, with an overall strong precision and recall on most classes. Findings point out the efficacy of image-based deep learning techniques for recognizing multiple malware families. Furthermore, authors highlight that scalable CNN-driven systems can help enhance automated malware analysis in modern security environments.

## 3. Research methodology

This paper presents a new lightweight system that is designed for realtime malware detection, especially on resource-constrained devices. The framework combines the Single Head Attention Mechanism using LSTM and GRU networks. The approach deals with a lot of aspects. The section gives details of how the dataset was collected. Moreover, it also includes pre-processing steps, model architecture, algorithm design, and implementation of the system. All in all, the aim is to achieve detection accuracy and efficiency.

### *3.1 Dataset and Pre-processing*

The CIC-MalMem-2022 dataset has been used for experiments, which has nearly 58,000 samples of benign and malicious processes (Tahir et al., 2022). All numerical features in the dataset were transformed to the range between 0 and 1 using Min-Max scaling. It makes sure that no particular feature would dominate the training process due to its absolute size. Values were rounded to four decimal places to enhance precision and improve generalization. Normalization is especially crucial for models like LSTM, GRU, and attention-based architectures, as it stabilizes training, enables effective learning of temporal dependencies, and mitigates bias from features with larger numeric ranges. The normalization formula used is:

$$Normalized\ Value\ = \frac{Orginal\ Value - Min(A)}{Max(A) - Min(A)} \tag{1}$$

where original value is the actual value of a feature in the dataset, Min(A) is the minimum value of feature A, and Max(A) is the maximum value of feature A. To make the model more adaptable and to avoid overfitting, numeric values were rounded to four decimal places. In this way, a consistent level of precision across all features was maintained. Normalization is critical for models like LSTM, Gru, and attention-based models, as these models are more sensitive to the scaling of the input data. Features with significantly varying values can result in unstable training, slower training, and reduced accuracy. By using Min-Max scaling, the training of temporal dependencies and attention can be performed without the influence of the scale of any single feature, thereby producing more reliable predictions. Through this pre-processing pipeline, well-structured and informative input is provided to the model, and behavioral differences between benign and malicious actions are emphasized.

### *3.2 Recurrent Neural Networks (RNNs)*

Recurrent Neural Networks (RNNs) are considered particularly effective for handling sequential data, such as time series or curves that vary with conditions like voltage, temperature, or light intensity (Daubechies et al., 2022). Unlike traditional Artificial Neural Networks (ANNs), feedback loops are incorporated in RNNs, allowing information from previous steps in the sequence to be retained. Because of this capability, RNNs are regarded as suitable for complex tasks such as pattern recognition, machine translation, image captioning, and modeling I–V characteristics under changing environmental factors (Raman et al., 2024). In RNNs, the flow of information is structured differently from that in standard ANNs, as illustrated in Fig. 1.

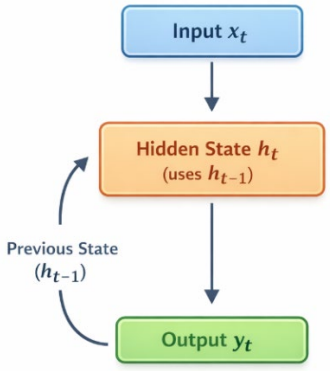


**Fig. 1.** RNN with data flow between neurons

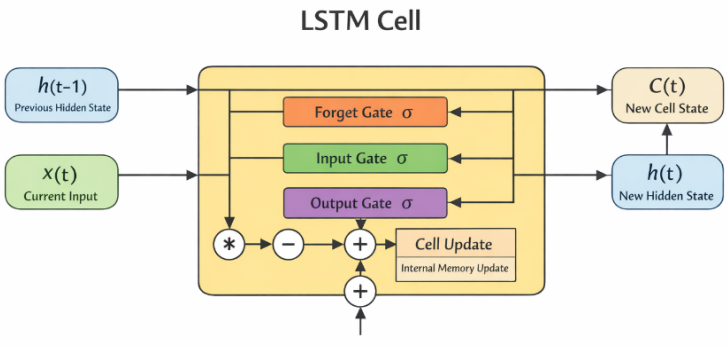


**Fig. 2.** Diagram of the structure of an LSTM

The weights are represented by $h_{t-1}$ and $X_t$. The term $h_{t-1}$ denotes a weight that is multiplied by the output of the previous neuron, enabling context to be maintained throughout the sequence. As in ANNs, activation functions are applied in RNNs to introduce non-linearity. During training, weight adjustments are performed using Backpropagation through Time (BPTT), in which the error across all time steps is calculated and the weights are updated accordingly to enhance performance. However, simple RNNs encounter a well-known issue called the gradient vanishing problem, which makes it tough for them to learn from very long sequences of data. During training, recurrent neural networks adjust their weights using the backpropagation through time (BPTT) algorithm to minimize errors. But when the sequences are lengthy, the weight updates from earlier steps become so small that they practically disappear. Over time, these weights shrink to the point of vanishing, which prevents the network from remembering information from the start of the sequence. This limitation hinders the model's ability to learn long-term dependencies and understand relationships that span many steps (Ali et al., 2025b). The long short-term memory (LSTM) network provides a more efficient solution compared with the simple recurrent neural networks (RNNs) due to the unique features of the LSTM architecture. The LSTM framework has the ability to overcome the limitations associated with the RNNs (Krichen and Mihoub, 2025). The LSTM has a structural component in the form of gates that play a critical role in the memory of the neurons. Each LSTM has three self-connected gates that are responsible for reading, writing, and erasing the information stored in memory. The LSTM architecture is shown in Fig. 2, where the complicated computations performed by the LSTM cell and the three gates are described. The gates can be considered as filters in LSTM architecture.

The forget gate helps decide what data to erase from memory, using an activation function and mathematical calculations. The input gate determines what should be stored in memory, while the output gate decides which information from memory to use. Like a simple RNN, this cell takes in previous data, but the LSTM has two states: the cell state (represented as $C_{t-1}$) and a hidden state (represented as $h_{t-1}$), along with the current data input (denoted as $X_t$). Each LSTM cell has hidden weights linked to the gates, meaning there are many weights to adjust. LSTMs employ Backpropagation through Time (BPTT) for weight adjustment during training. Although they excel in modeling sequential data, LSTMs can be computationally heavy due to the numerous weights that need training. To tackle this challenge, Chung introduced the Gated Recurrent Unit (GRU) in 2014 as a more streamlined alternative to the LSTM (Chung et al., 2014). The GRU simplifies the gating mechanism while still effectively managing long-term dependencies. It consists of two gates: the reset gate and the update gate, which control the flow of information. The update gate decides how much of the existing memory to keep and how much new input to include, determining what information is still relevant. Meanwhile, the reset gate decides how much of the previous memory to discard when processing new information, allowing the network to essentially "start fresh" when needed (Khdhir et al., 2024). Compared to LSTMs, which have two states a cell state and a hidden state, GRUs have only one state, only the hidden state ($h_{t-1}$). In addition, GRUs contains the current input Xt. The Tanh activation function is frequently employed to normalize the output, allowing for smoother transitions in the network. Thanks to its streamlined structure, GRUs requires reduced computational power while still capturing sequential dependencies effectively, being an efficient and powerful method for modeling sequential data (Zheng & Chen, 2021).

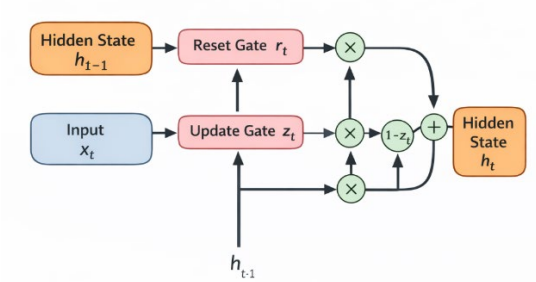


**Fig. 3.** Diagram of the structure of a GRU

Neural networks rely on a range of activation functions, like Sigmoid, Tanh, ReLU, and Linear, to add some nonlinearity to their models. These functions change input values in various ways, enabling the network to identify complex patterns. For instance, ReLU gives a zero output for negative inputs, the sigmoid function converts inputs into a range between 0 and 1, while Tanh adjusts inputs to fall between -1 and 1. Each of these activation functions offers a unique transformation, tailored to enhance learning in specific situations (Daubechies et al., 2022).

### *3.3 Single Head Attention Mechanism*

Single-head attention can be defined as an initial and basic form of attention used in neural sequence models. The basic idea behind single-head attention is that neural networks should not be forced to encode an entire sequence into a fixed-size representation, as this can cause a loss of information and a reduction in network performance for longer sentences. The first impactful introduction of this idea was based on the introduction of an alignment-based attention mechanism for neural machine translation models (Song et al., 2021). The contribution of this initial impactful work was to prove that a neural machine translation model could be allowed to "look back" at relevant parts of the input sequence to improve its accuracy. In the single-head attention mechanism, the model makes use of three learnable matrices that project the input representations onto queries (Q), keys (K), and values (V). Each of the vectors has a different role: queries represent the

current position that seeks information, keys represent the potential positions that have the information, and values hold the information that is aggregated. This allows the network to generate relevance scores between the query and the keys, thereby enriching the output. The mathematical formulation of the attention mechanism, as described in (Dong et al., 2021) and commonly used in scaled dot-product attention, is expressed as:

$$Attention\ (Q\ ,K\ ,V)softmax\ \left(\frac{(QK^T)}{\sqrt{d_k}}\right)\ V \tag{2}$$

In this context, the denominator$\sqrt{d_k}$ normalizes the dot product results, which would otherwise potentially result in large values that would destabilize the softmax operation. The softmax component plays an important role in that it normalizes the results of the dot product operation into a probability distribution that reflects the degree to which each of the keys would contribute to the final result. This operation is simple yet powerful in that it enables the model to reason contextually by amplifying or reducing information as is necessary in the query. It is important to note that the operation is performed across all the positions in parallel, hence justifying its use in modern architecture. While single-headed attention is only capable of capturing a single relational pattern at any given time, the approach is still theoretically significant. However, the main drawback of single-headed attention is the fact that it has a high probability of being limited by the ability of the attention mechanism to learn only a single dominant relationship between queries and keys. This has led to the development of the multi-head attention approach (Li et al., 2021). However, the single-head attention approach is still being extensively researched due to the fact that the approach provides a clear insight into the internal dynamics of attention mechanisms. Subsequent refinements and analyses have continued to shed light on the flexibility of the concept of attention through the incorporation of novel scoring functions and structural variations (Yan et al., 2024). Yet, through these various developments, the mathematical and conceptual underpinnings of the concept of single-head attention have continued to endure. It remains an integral part of encoder-decoder architecture, recurrent neural networks equipped with attention, and even reduced versions of the transformer architecture that are geared towards computational or interpretability considerations. Its beauty lies in its ability to transform linear algebra into contextual reasoning, making it arguably the most impactful concept in modern-day deep learning.

In the following subsection, a hybrid model is proposed that combines an LSTM/GRU network with a single head attention mechanism. This allows the network to learn complex time dependencies while focusing on the most relevant input features, thus improving overall prediction accuracy.

### *3.4 Proposed Model*

The proposed model is designed to be efficient in learning temporal relationships while emphasizing features that illustrate malicious (or fraudulent) behavior. The proposed architecture consists of the following components:

**1. LSTM/GRU layer:** This is a recurrent layer with 64 units used to learn long-term dependencies in the sequence.

**2. Single-head Attention layer:** This layer is used to weigh each time frame in the sequence. This allows the highlighting of relevant time frames, such as anomalous memory accesses or system calls. This is achieved at less computational cost by using only one head.

**3. Dense layer:** This layer consists of 64 units with ReLU activation used to learn the interactions between complex features.

**4. Output layer:** This layer consists of a single node with sigmoid activation used in binary classification between malicious or benign behavior. This is shown in Table 1.

**Table 1**
The Proposed Model Architecture

| Layer | Configuration |
|---|---|
| **LSTM/GRU** | 64 units |
| **Single-Head Attention** | Single attention head |
| **Dense** | 64 units, ReLU activation |
| **Output** | 1-unit, sigmoid activation |

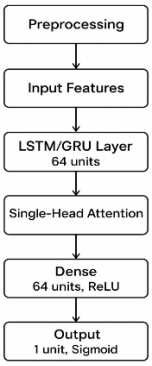


**Fig. 4.** Flowchart of the Proposed Model

The entire process of the model, starting from normalization and leading up to the final classification, is shown in Fig. 4.

## 4. Performance evaluation and results analysis

This section presents the results obtained from the simulation and will elaborate on the methodology of the evaluation. The process of implementation will be described, and the proposed model will be compared with existing ones. The metrics used in the evaluation will be identified, and the results obtained from the simulation will be discussed in the context of existing literature. The model has been implemented in the Python environment using popular libraries such as TensorFlow, NumPy, pandas, and scikit-learn. It has been demonstrated that the model runs efficiently on both CPU and GPU environments. The data has been divided into 70% for training, 15% for validation, and 15% for the test. Experiments have been performed on a personal computer with Intel Core i7 processor, 16 GB of RAM, and 64-bit Windows 10 as the operating system.

### *4.1 Performance evaluation*

The study utilizes three key parameters to evaluate the efficiency of classification models:

1- Accuracy: Refers to the number of correct predictions in relation to the total number of predictions made by the model.

2- Recall (Sensitivity), Precision (PPV): These parameters can be used to better evaluate the efficiency of the model, particularly its ability to identify relevant instances.

3- F1 Score: Represents the harmonic meaning of the precision and recall of the model. The F1 score considers both false positives and false negatives. The F1 score can be used when the class distribution of the data set is imbalanced. More accurate results can be achieved when the cost of false positives and false negatives is presumed to be equivalent (Rodríguez et al., 2020).

### *4.2 Result analysis and discussion*

The proposed model is based on binary classification, where the loss function is binary cross-entropy. For the training process, the Adam optimizer is used with a learning rate of 0.001. For the performance of the proposed model, accuracy, precision, recall, and F1-score are used. The desired value is set to more than 90%. For avoiding overfitting, dropout is used with a dropout rate of 0.2. Additionally, batch normalization is used. The sequential learning process is utilized, where the hidden state is updated at each time step to learn the temporal pattern. The single-head attention mechanism allows the model to automatically focus on the most important time steps, eliminating the need for manual feature selection. The proposed approach balances the trade-off between accuracy and efficiency, making it suitable for real-time applications, especially for IoT and mobile devices. In terms of computational complexity, it is found that the LSTM/GRU layer has a time complexity of $O(T \times d^2)$, whereas the single-head attention mechanism has a time complexity of $O(T^2 \times d)$. Even though there is a quadratic term, it is found that the training speed of the single-head mechanism is up to 30% faster than multi-head attention mechanisms, as presented in (Zhao et al., 2022).

This proposed model is found to show strong results in terms of accuracy, precision, and recall, as presented in Table 2, with an accuracy of 92%, precision of 91%, and recall of 93%. The proposed model is found to show better results compared to previously conducted studies, as presented in (Santos et al., 2022; Rashid eta l., 2023), by successfully modeling the time dependencies of prominent features like QRR, LRSD, and HRSD, as well as automatically focusing on the most informative parts. This is found to be useful in identifying even slight indications of malicious activity, as might be missed by traditional approaches like CNN, SVM, Random Forest, and LSTM.

**Table 2**
Performance Comparison

| Model | Dataset | Accuracy (%) | Precision (%) | Recall (%) | Training Time |
|---|---|---|---|---|---|
| **The Proposed Model** | CIC-MalMem-2022 | 92 | 91 | 93 | 4 hours |
| **CNN** | CIC-MalMem-2022 | 88 | 87 | 89 | 8 hours |
| **SVM** | CIC-MalMem-2022 | 85 | 84 | 84 | 5 hours |
| **Random Forest** | CIC-MalMem-2022 | 87 | 86 | 88 | 3 hours |
| **GRU with Attention** | CIC-MalMem-2022 | 91 | 90 | 92 | 6 hours |
| **Baseline LSTM** | CIC-MalMem-2022 | 86 | 85 | 87 | 4.5 hours |

It maintains an optimal balance between precision and recall, hence reducing false positives while ensuring that most of the malicious instances are correctly classified. The high precision of the model ensures that it sends very few false alerts, and its high recall ensures that it rarely misses any malicious behavior. This balance between precision and recall is very important for a cybersecurity application. In conclusion, the proposed model shows which temporal features most contribute to the classification decision-making process. By understanding how the automated system works, it bolsters trust in the system while revealing insights into malicious behavior. To sum up, the model shows impressive detection performance, efficient training, and clear feature interpretability. Thus, it shows several advantages over existing approaches and can benefit real-world cybersecurity applications.

**Table 3**
Comparison of statistical significance

| Comparison | t-statistic | p-value | Significance (Î± = 0.05) |
|---|---|---|---|
| **The Proposed Model vs. CNN** | 4 | 0.00006 | Significant |
| **The Proposed Model vs. Random Forest** | 5 | 0.00002 | Significant |
| **The Proposed Model vs. GRU with Attention** | 1 | 0.317 | Not Significant |
| **The Proposed Model vs. Baseline LSTM** | 6 | 0.000009 | Significant |

Table 3 shows the performance of the proposed model compared with several baseline methods. To verify the observed improvements, a statistical significance test was conducted using Welch's t-test with a significance level of $\alpha = 0.05$. The hypotheses were defined as follows: $H_0$ states that no significant difference exists in mean accuracy between the proposed model and each baseline model, while $H_1$ states that a significant difference is present. The proposed model reached 92% accuracy and outperformed the CNN, Random Forest, and Baseline LSTM models, with p-values below 0.05, indicating statistically significant improvements. In contrast, the comparison with the GRU with Attention model resulted in a p-value of 0.317, showing no significant difference and suggesting that both attention-based models perform at a similar level. These results highlight that the proposed model not only offers better accuracy than traditional baselines but also competes well with advanced attention-based methods. Along with its high precision and recall, efficient training, and confirmed statistical significance, these findings showcase the model's robustness, reliability, and its suitability for real-time detection of malicious behavior. The experimental results were analyzed using Python and the SciPy library. Descriptive statistics, t-tests, and visualization methods were applied to compare the performance of the proposed model with baseline approaches as shown in Fig. 5.

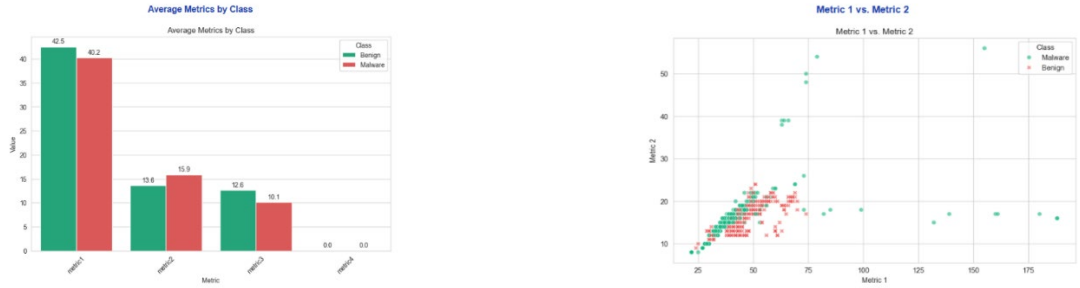


**Fig. 5.** Performance of the proposed model

The complete Python code for the statistical and exploration analysis is available upon request and can be shared with academic institutions or researchers to support transparency and reproducibility.

## 5. Conclusion

This study proposes an efficient malware detection framework that integrates Single-Head Attention into LSTM/GRU architecture. The CIC-MalMem-2022 dataset was used to evaluate and optimize the model. The model gives incredible results as it shows 92% accuracy, 91% precision, 93% recall, and 92% F1-score. Furthermore, it reduces the training time by nearly 30% when compared with multi-head attention models. As a result, the method in this paper is suitable for real-time implementation in the context of resource-constrained environments such as IoT devices and mobile platforms. Despite this, certain limitations still exist. The model may have difficulty with significantly obfuscated malware or malware that has previously not been seen, if the training data contains no representation of their behaviours. Moreover, using memory-based features may lose out on significant indications of a malicious event, especially when the attacker uses advanced evasion techniques or targets components other than memory. To verify how the model works with other data and in real-life situations, validating the model in other data sets is needed. In future studies, further characteristics like network traffic and file system activities could be investigated to possibly increase the detection probability. Another direction that could be explored is the hybrid/ensemble approach by combining various detection methods. Another direction that could be explored is the application of this methodology in other domains of cybersecurity, like intrusion detection or fraud detection, and validating it with various datasets.

## References

Abualhaj, M. M., Al-Khatib, S., Al Shafi, N., Qaddara, I., & Hyassat, A. (2025). Utilizing gray wolf optimization algorithm in malware forensic investigation. *Journal of Computational and Cognitive Engineering*, *4*(4), 591-602.

Abualhaj, M. M., & Al-Khatib, S. N. (2024). Using decision tree classifier to detect Trojan Horse based on memory data. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *22*(2), 393–400. https://doi.org/10.12928/telkomnika.v22i2.25753

Akinshola-Awe, F. J., Obiniyi, A. A., Aimufua, G., Anyachebelu, K., & Ajayi, B. A. (2025). Detection And Classification of Malware Using Grid Search Optimization Technique. *Science World Journal*, *20*(2), 740–747.

Akinshola-Awe, F. J., Obiniyi, A. A., Aimufua, G. I. O., & Anyachebelu, T. K. (2024). Framework for the detection and classification of malware using machine learning. *Dutse Journal of Pure and Applied Sciences*, *10*(3a), 177–186.

https://doi.org/10.4314/dujopas.v10i3a.17
Alashjaee, A. M. (2025). Deep learning for network security: An Attention-CNN-LSTM model for accurate intrusion detection. *Scientific Reports*, *15*(1), 21856. https://doi.org/10.1038/s41598-025-07706-y
Al-Ghanem, W. K., Qazi, E. U. H., Zia, T., Faheem, M. H., Imran, M., & Ahmad, I. (2025). *MAD-ANET: Malware Detection Using Attention-Based Deep Neural Networks*. Computer Modeling in Engineering & Sciences, 143(1),1009–1027. https://doi.org/10.32604/cmes.2025.058352
Alhady, M., & Abdulbasit, A. (2024). A Real-Time AI-Powered Framework for Integrated Malware, Phishing, and Security Breach Detection. *Surman Journal of Science and Technology*, *6*(2), 601–607.
Ali, A., Akram, M. A., Farooq, W., Ali, M., Nazir, M., Muhammad, A., & Mazhar, T. (2025a). MalwareVison: A Deep Learning-Driven Approach For Malware Classification. *Journal of Computing & Biomedical Informatics*, *8*(02). https://www.jcbi.org/index.php/Main/article/view/961
Ali, A., Ullah, I., Singh, S. K., Sharafian, A., Jiang, W., I. Sherazi, H., & Bai, X. (2025b). Energy-Efficient Resource Allocation for Urban Traffic Flow Prediction in Edge-Cloud Computing. *International Journal of Intelligent Systems*, *2025*(1), 1863025. https://doi.org/10.1155/int/1863025Digital Object Identifier (DOI)
Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. https://doi.org/10.48550/arXiv.1412.3555
Daubechies, I., DeVore, R., Foucart, S., Hanin, B., & Petrova, G. (2022). Nonlinear approximation and (deep) ReLU networks. *Constructive Approximation*, *55*(1), 127-172. https://doi.org/10.1007/s00365-021-09548-z
Dong, Y., Cordonnier, J. B., & Loukas, A. (2021, July). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning* (pp. 2793-2803).
Ferdous, J., & Islam, R. (2024). A Survey on Machine Learning Techniques in Multi-platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments.
Khdhir, R., Othmen, S., Belghith, A., & Alamri, J. (2024, May). Breast cancer segmentation in mammograms using antlion optimization and CNN/GRU architectures. In *2024 International Wireless Communications and Mobile Computing (IWCMC)* (pp. 1030-1035). IEEE. 10.1109/IWCMC61514.2024.10592614
Krichen, M., & Mihoub, A. (2025). Long short-term memory networks: A comprehensive survey. *AI*, *6*(9), 215. https://doi.org/10.3390/ai6090215
Li, J., Wang, X., Tu, Z., & Lyu, M. R. (2021). On the diversity of multi-head attention. *Neurocomputing*, *454*, 14-24. https://doi.org/10.1016/j.neucom.2021.04.038
Raman, R., Kumar, V., Pillai, B. G., Rabadiya, D., Divekar, R., & Vachharajani, H. (2024, May). Forecasting Bitcoin value with hybrid LSTM-GRU neural networks. In *2024 Second International Conference on Data Science and Information System (ICDSIS)* (pp. 1-5). IEEE.
Rashid, M. M., Islam, M. S., & Hossain, M. S. (2023). Malware detection using ensemble learning: A review. *Sensors, 23*(2), 567. https://doi.org/10.3390/s23020567
Rodríguez, G., Mateos, C., & Misra, S. (2020, October). Exploring web service QoS estimation for web service composition. In *International Conference on Information and Software Technologies* (pp. 171-184). Cham: Springer International Publishing.
Santos , J. C. , Peruma , A. , & Mirakhorli , M. (2022). CNN-based malware detection on memory features. Software: *Practice and Experience*, *52*(10),2267–2281. https://doi.org/10.1002/spe.3123
Song, J., Kim, S., & Yoon, S. (2021). AligNART: Non-autoregressive neural machine translation by jointly learning to estimate alignment and translate. *arXiv preprint arXiv:2109.06481*.
Tahir , R. , Asghar , S. , & Zafar , S. (2022). CIC-MalMem-2022: A new dataset for memory-based malware detection. *Computers & Security*, 113, 102552. https://doi.org/10.1016/j.cose.2021.102552
Taşcı, B. (2024). Deep-learning-based approach for iot attack and malware detection. *Applied Sciences*, *14*(18), 8505. https://www.mdpi.com/2076-3417/14/18/8505
Yan, S., Fan, C., Zhang, H., Yang, X., Tao, J., & Lv, Z. (2024). Darnet: Dual attention refinement network with spatiotemporal construction for auditory attention detection. *Advances in Neural Information Processing Systems*, *37*, 31688-31707.
Zhao, Z., Chen, W., & Luo, X. (2022). A review of deep learning-based malware detection in the Internet of Things. *Sensors*, 22(4), 1479. https://doi.org/10.3390/s22041479
Zhang, W., Li, H., Tang, L., Gu, X., Wang, L., & Wang, L. (2022). Displacement prediction of Jiuxianping landslide using gated recurrent unit (GRU) networks. *Acta Geotechnica*, *17*(4), 1367-1382. https://doi.org/10.1007/s11440-022-01495-8
Zheng, W., & Chen, G. (2021). An accurate GRU-based power time-series prediction approach with selective state updating and stochastic optimization. *IEEE Transactions on Cybernetics*, *52*(12), 13902-13914. 10.1109/TCYB.2021.3121312

© 2026 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC, BY) license (http://creativecommons.org/licenses/by/4.0/).