Babylonian Journal of Internet of Things
Vol.2026, **pp**. 155–169
DOI: https://doi.org/10.58496/BJIoT/2026/010; ISSN: 3006-1083
https://mesopotamian.press/journals/index.php/BJIoT

Research Article

Deep Learning-Based Malware Detection and Classification Using Memory Behavioral Features

Mohammed Basil Abdulkareem [1,*], Kassem Hamze [2], Mohammad Abdullah Abbas [3]

[1] *Department of Business Administration, College of Administration and Economics, University of Anbar, Al Anbar, 31001, Iraq.*

[2] *Faculty of Engineering-Islamic University of Lebanon.*

[2] *Universiti Sains Malaysia.*

ARTICLE INFO

Article History

Received 25 Apr 2026
Revised 15 May 2026
Accepted 22 Jun 2026
Published 09 Jul 2026

Keywords

Malware Detection,
Deep Learning,
CNN–LSTM,
Memory Behavioral Features,
Cybersecurity.

ABSTRACT

As malware evolves and grows more sophisticated, it remains a major challenge for modern cybersecurity solutions to detect and prevent. The traditional approaches for malware detection, which are mostly based on signature-based detection and manually engineered features, are no longer effective against new variants of malware and the ones that are highly obfuscated. Considering these drawbacks, this research aims to propose an effective lightweight hybrid Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) for detecting and categorizing malware based on its memory behavioral features. The proposed framework can automatically learn the representative behavioral patterns that exist in the CIC-MalMem-2022 benchmark dataset using a systematic pipeline that includes data preprocessing, feature normalization, CNN-based feature extraction, LSTM-based sequential learning, and Softmax classification. Model performance is assessed on common performance metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix, Training Time, Inference Time. The experimental results show that the proposed CNN–LSTM framework not only performs well in terms of classification accuracy but also has a low computational complexity, which can be applied to practical cybersecurity applications. The hybrid architecture is able to effectively incorporate both spatial and sequential behavioural aspects to be able to discriminate between benign and other multiple malware families. In conclusion, the suggested approach offers a promising and effective deep learning method for detecting and categorizing malware using memory behavior analysis, which helps advance the creation of intelligent cybersecurity systems.

1. INTRODUCTION

Modern computing environments are complex and are rapidly evolving with new technologies such as digital communication, cloud, Internet of Things (IoT), and software-defined networks (SDN) growing the attack surface for cybercriminals. As a result, the malware threat is one of the most serious cybersecurity challenges to which personal devices, enterprise infrastructure, cloud platforms, and critical information systems are vulnerable. Conventional signature-based detection techniques cannot effectively protect against newly emerging threats because modern malware uses sophisticated methods like polymorphism, metamorphism, code obfuscation, encrypted payloads and zero-day exploits [1]. To address these shortcomings, researchers have turned to behavioral malware analysis, which is based on observing runtime behavior instead of on the static qualities of the executable. Of these methods, memory behavioral analysis has proven to be a promising method since it captures the runtime behavior patterns generated during malware execution, which allows for the detection of known and unseen malware families more reliably. The architecture proposed in this work uses a hybrid deep learning model to automatically detect and classify malware based on their memory behavioral characteristics as shown in figure 1.

*Corresponding author. Email: mohammed.basil@uoanbar.edu.iq

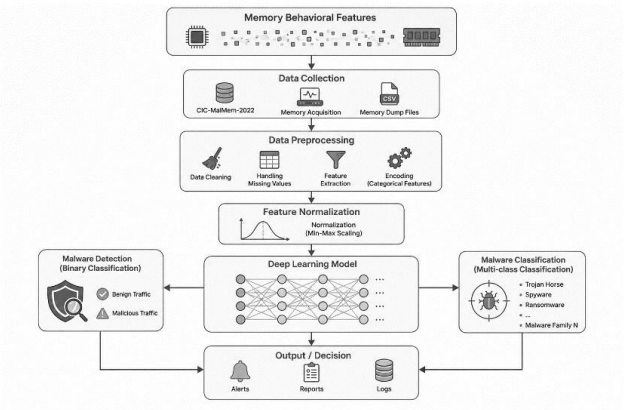


Fig. 1. Conceptual overview of the proposed deep learning-based malware detection and classification framework using memory behavioral features.

The traditional ways of detecting malware include signature-based detection, heuristic analysis and manually crafted rules. While they work against known malware families, there are a number of drawbacks when having to deal with novel attacks or attacks that are very well obfuscated. Signature-based techniques require up to date databases, and heuristic techniques may have high false positive rates in large complex network environments. In addition, cloud applications, IoT devices and edge computing platforms are responsible for a significant volume and variety of traffic on networks that has severely hampered the scalability of traditional malware detection systems [1-6]. Given this, the need arises for intelligent detection models that automatically learn complex malware behaviors from memory behavioral features.

The recent progress in deep learning has added highly effective data-driven methods for cybersecurity applications. Recent studies have shown the effectiveness of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks and hybrid deep learning architectures to detect malware by automatically learning discriminative features from a large-scale dataset, without extensive manual feature engineering [7-10]. Unlike traditional machine learning models, deep learning models can also learn from memory behavioral features that are generated by the execution of malwares, which provides better performance in detecting known and unknown malwares, especially in detecting hidden behavioral patterns or nonlinear relationships. However, much of the current research either focuses on static classification of existing malware by executable files, or relies on complex hybrid architectures, whereas the research on malware family classification and complete malware detection based on memory behavioral attributes is relatively limited [11-13].

While a great deal has been accomplished in the field of deep learning for malware detection, there are still some research challenges that have not been addressed. The architectures used in existing architectures are often compute-intensive and demand a lot of training data, or are optimized for specific datasets for benchmarking. Furthermore, most of the published works focus on the classification accuracy without much discussion of computational efficiency, scalability of the model and deployment in practice. The restrictions are indicative of the need for a research gap to build effective deep learning frameworks that accurately detect and classify malware based on memory behavior with acceptable computational complexity and high generalization ability among different malware families [14-16].

Inspired by these challenges, this research has been proposed to detect and classify malware with lightweight deep learning framework with memory behavioral features. The proposed framework automatically learns representative behavioral characteristics from the execution of malware, not just from handcrafted features or static analysis of the malware's executable code. It has an integrated framework that encompasses the systematic data preprocessing, which involves data cleaning, duplicate data removal, label encoding, feature normalization, optimized model training, and detailed performance assessment with popular classification metrics. The proposed model is evaluated on the CIC-MalMem-2022 benchmark dataset to rigorously test the model's ability to accurately classify benign samples and multiple malware families, as well as its computational efficiency and robustness.

The following are the major contributions of this study. First, a lightweight deep learning framework was developed to detect and classify malware on the basis of memory behavioural characteristics rather than static characteristics of executables. Second, in order to better quality the features and increase the efficiency of the learning process before training the model, a systematic preprocessing pipeline is introduced. Third, the proposed framework is evaluated comprehensively through experiments conducted on the CIC-MalMem-2022 benchmark dataset and various performance metrics to prove the efficiency and robustness of the proposed framework. Finally, the proposed approach is compared with some recent state-of-the-art deep learning approaches to show its potential to guarantee accurate malware detection and malware family classification with reasonable computation time.

2. PROPOSED METHODOLOGY

The proposed methodology proposes a lightweight hybrid deep learning framework to detect and classify malware based on memory behavioral features based on the CIC-MalMem-2022 dataset. The framework combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to leverage the spatial feature extraction and sequential dependency learning. The data set has to pass through a number of preprocessing steps before it is used for classification, to enhance data quality and learning efficiency. These extracted behavioral features are fed into the CNN–LSTM model to classify benign applications and malicious samples, and then classify the malware into the corresponding families. Lastly, the effectiveness of the proposed framework is investigated based on commonly used classification performance parameters like the Accuracy, Precision, Recall, F1-score, Confusion Matrix, training time and inference time. The proposed framework has five consecutive stages of the overall workflow.

1. Dataset acquisition.
2. Data preprocessing.
3. CNN-based feature extraction.
4. LSTM-based malware classification.
5. Performance evaluation.

Thus, the proposed framework aims to automatically identify and classify malware by examining the behavioral features of programs in memory generated during their execution. The proposed framework automatically learns the representative behavioural features by using a hybrid CNN-LSTM architecture, which fundamentally differs from the traditional methods of malware analysis based on static signatures or handcrafted rules. This ensures that the model is not only effective in detecting malicious software but also efficient enough to be used in real cybersecurity scenarios.

Initially, memory behavioral data are collected from the CIC-MalMem-2022 benchmark dataset. The raw data includes information about both benign apps and several malware families based on memory-related behavioral data. These data are then preprocessed to enhance their quality by eliminating duplicate samples, filling in missing data, encoding class labels, and normalizing numerical attributes. The resulting cleaned data set is then split into training and testing sets for supervised learning.

In the feature learning stage, the normalized behavioral features are first fed into convolutional layers to automatically learn the local feature representations that are more discriminative. The pooling layers help to compress the feature space into informative patterns. These extracted feature maps are then passed to the LSTM network which can learn the sequential relationship between the features that cannot be captured by CNN well and learn the long term dependency among the behavioral features.

The output of the LSTM layer is then fed into fully connected layers before a Softmax classifier which gives the final prediction. The proposed model first identifies if a sample is benign or malicious. The framework also identifies the malware family of the detected malicious behavior by using the behavioral representations learned from the sample using the earmarking step.

Lastly, the proposed framework uses several statistical measures to assess the classification performance such as Accuracy, Precision, the Recall, F1-score, Confusion Matrix, training time, inference time, and computational complexity. These criteria comprehensively evaluate the ability to detect and the efficiency of the deployment of the proposed CNN–LSTM framework. The proposed methodology is presented in the form of a complete workflow in Figure 2.

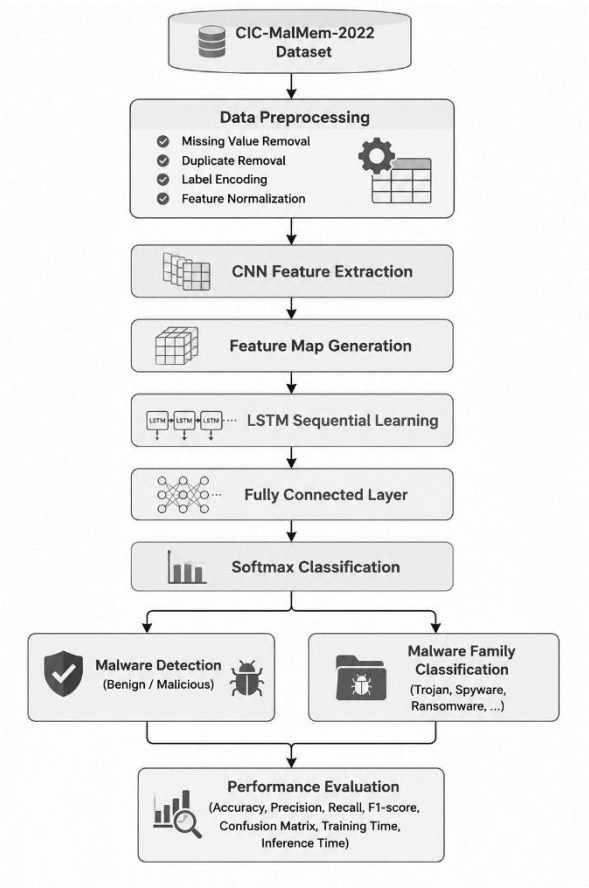


Fig. 2. Overall architecture of the proposed CNN–LSTM malware detection and classification framework using memory behavioral features.

2.1. CIC-MalMem-2022 Dataset

The proposed framework is evaluated experimentally with the CIC-MalMem-2022 dataset, a publicly available malware detection and classification benchmark for the Canadian Institute of Cybersecurity (CIC). CIC-MalMem-2022 is built from memory behaviour features extracted from the runtime of the malware, which are not available in traditional malware sets usually built from static analysis of the malwares' executable files. The runtime features capture in memory the behavior of the applications that make up the software, allowing the detection of advanced variants of malicious software that traditional signature-based detection methods might not be able to identify.

The dataset has been labelled with benign applications as well as malicious software samples, thus offering a well-balanced benchmark for supervised deep learning. The proposed framework can also be used to detect malicious samples, besides performing binary malware detection (Benign versus Malicious) and multi-class malware family classification, as malicious samples exist and can be classified into several families with different behavior patterns. The generalization of the CNN–LSTM model is enhanced by the variety of malware families, and the evaluation of all malware behaviors can be performed. The general statistics of the CIC-MalMem-2022 data set used in this study are shown in Table 1.

The dataset is processed through some stabilization and data quality improvement steps before the model training. Duplicate records are eliminated, missing values are treated correctly, categorical class labels are transformed into numerical representations and numerical features are scaled between 0 and 1 with Min–Max scaling. Then, the obtained data is randomly split into 80% training and 20% testing sets to test the effectiveness of the proposed framework in classifying unseen samples of malware, and at the same time preventing overfitting.

The dataset also includes several families of malware gathered during real malware executions, in addition to binary labels. These families encompass a wide variety of malicious activities such as information stealers, remote access, ransomware activities, downloaders, and backdoors. The existence of these malware classes allows for realistic conditions of the evaluation process of the proposed CNN–LSTM framework to be faced, both in terms of malware detection and malware family classification. The distribution of the malware family analyzed in this research is shown in Table 2.

TABLE I. CHARACTERISTICS OF THE CIC-MALMEM-2022 DATASET

Parameter	Description
Dataset	CIC-MalMem-2022
Source	Canadian Institute for Cybersecurity (CIC)
Data Type	Memory Behavioral Features
Total Samples	58,596
Benign Samples	29,298
Malware Samples	29,298
Malware Families	15 Families
Classification Tasks	Binary Detection and Multi-class Classification
Data Cleaning	Duplicate Removal and Missing Value Handling
Feature Scaling	Min–Max Normalization
Data Split	80% Training / 20% Testing
Learning Type	Supervised Deep Learning

TABLE II. MALWARE FAMILY DISTRIBUTION IN CIC-MALMEM-2022

Category	Malware Families
Benign	Legitimate software samples
Spyware	RedLine, SpyNote, SnakeKeylogger, FormBook
RAT (Remote Access Trojan)	AsyncRAT, NjRAT, Remcos
Trojan	AgentTesla, LokiBot, NanoCore
Downloader / Dropper	GuLoader, SmokeLoader
Ransomware	WannaCry, Stop/Djvu, LockBit

The designated equal number of benign and harmful samples along with the wide variety of malware families offers a thorough platform for assessing the suggested CNN-LSTM framework. The model is trained directly using the features of the memory, and it is expected to recognize not only malware and the features it behaves with similar to, but also to recognize the family of generated malware. This dataset is thus suitable for testing the effectiveness, robustness, and generalization power of the proposed deep learning framework.

2.2. Data Preprocessing

The quality of input data has a significant impact on the performance of deep learning models. Thus, the CIC-MalMem-2022 data needs to be preprocessed thoroughly prior to training the proposed CNN–LSTM framework, to make it consistent, remove redundant information and make preparation for learning behavioral features. The proposed preprocessing pipeline is aimed to reduce noise, reduce computation complexity and improve the generalization ability of the proposed model.

Data cleaning is the first pre-processing step, which removes corrupted, inconsistent or incomplete data. This process guarantees that the dataset has only valid behavioral samples for supervised learning. High data quality helps minimize the risk of misleading the learning process and enhances model stability while training.

Duplication of samples is detected after data cleaning and removed. The problem of duplication can cause bias in modeling training, as certain malware behaviors or benign samples might be overrepresented. Duplicated instances can be removed

to increase the statistical reliability of the data set and avoid redundant data, which helps with the generalization of the model.
The data is then analyzed for missing data. Records with too much missing data are discarded, while records with missing data are filled in with proper statistical methods. This operation keeps the integrity of the dataset intact without losing completeness of feature vectors before giving them to the deep learning model.
As the labels for each family of malware are categorical, label encoding is used to convert the labels of each family of malware into numerical labels suitable for supervised deep learning algorithms. Malware detection uses binary labels and individual malware families are assigned numerical labels during multi-class classification.
All the numerical features are normalized using Min–Max normalization for enhancing the convergence of the neural network during training. A scaling technique that maps all features to the interval [0,1] so that variables with large numerical ranges don't dominate the optimization process. Feature normalization also speeds up gradient convergence and enhances the classification performance.

Last, the preprocessed data is randomly split into 80% training and 20% testing sets. The training subset is utilized to adjust the CNN–LSTM parameters, while the testing subset is not used while training the CNN–LSTM and is only used to test the CNN–LSTM. This approach allows for an objective evaluation of the proposed framework, and limits the risk of over fitting. These preprocessing steps used in our study are summarized in Table 3 that shows the objective and contribution of each preprocessing step before the training of the deep learning model.

TABLE III. DATA PREPROCESSING OPERATIONS

Preprocessing Step	**Purpose**	**Expected Impact**
Data Cleaning	Remove corrupted and inconsistent records	Improve data quality and model reliability
Duplicate Removal	Eliminate repeated samples	Reduce redundancy and training bias
Missing Value Handling	Replace or remove incomplete records	Preserve dataset consistency
Label Encoding	Convert categorical labels into numerical values	Enable supervised learning
Feature Normalization	Apply Min–Max scaling to numerical features	Improve convergence and training stability
Train/Test Split	Divide dataset into 80% training and 20% testing	Evaluate model generalization and prevent overfitting

2.3. Proposed Deep Learning Model

The suggested Malware Detection framework is built with a CNN–LSTM hybrid structure that uses memory behavioral characteristics contained in CIC-MalMem-2022 to correctly detect and classify the malware. By combining CNN and LSTM, the model can leverage the strengths of both networks. Specifically, the CNN part automatically learns high-level spatial feature representations from the normalized behavioral attributes, while the LSTM part learns the long-term sequential dependency between the learned features. This hybrid learning approach enhances the ability of the model to differentiate between benign applications and malicious software and to correct identification of different families of malware.
Proposed architecture is divided into five successive processes: Input representation, CNN feature extraction, LSTM sequential learning, fully connected classification and Softmax prediction. The input layer gets the normalized memory behavioral features in the initial stages. These features are then fed to several convolutional layers to extract representative local feature patterns related to the behavior of malware. The convolution operation is followed by the Rectified Linear Unit (ReLU) activation function and max-pooling layers to reduce the dimensionality of the features while retaining the most salient features. The extracted feature maps are subsequently reorganised and passed on to the LSTM network for sequential behaviour analysis.

The LSTM layer captures temporal relationships within behavioural features through its memory cell mechanisms, allowing the network to retain important information but forget irrelevant patterns. LSTM has shown its ability to reduce the vanishing gradient problem and to have better learning ability for sequential behavioral data as compared to

conventional feedforward neural networks. The learned feature representation is then passed to one or more fully connected layers, where high-level behavioral patterns are formed before finally going into a classification stage.

Lastly, the Softmax classifier returns probabilities for each of the output classes. In binary classification, the model is used to classify an application as either Benign or Malicious. If malicious activities are detected, Multi-Class classification is done to find out the respective malware family. The entire CNN–LSTM structure used in this work is depicted in Figure 3, and the network structure and hyperparameters are listed in Table 4.

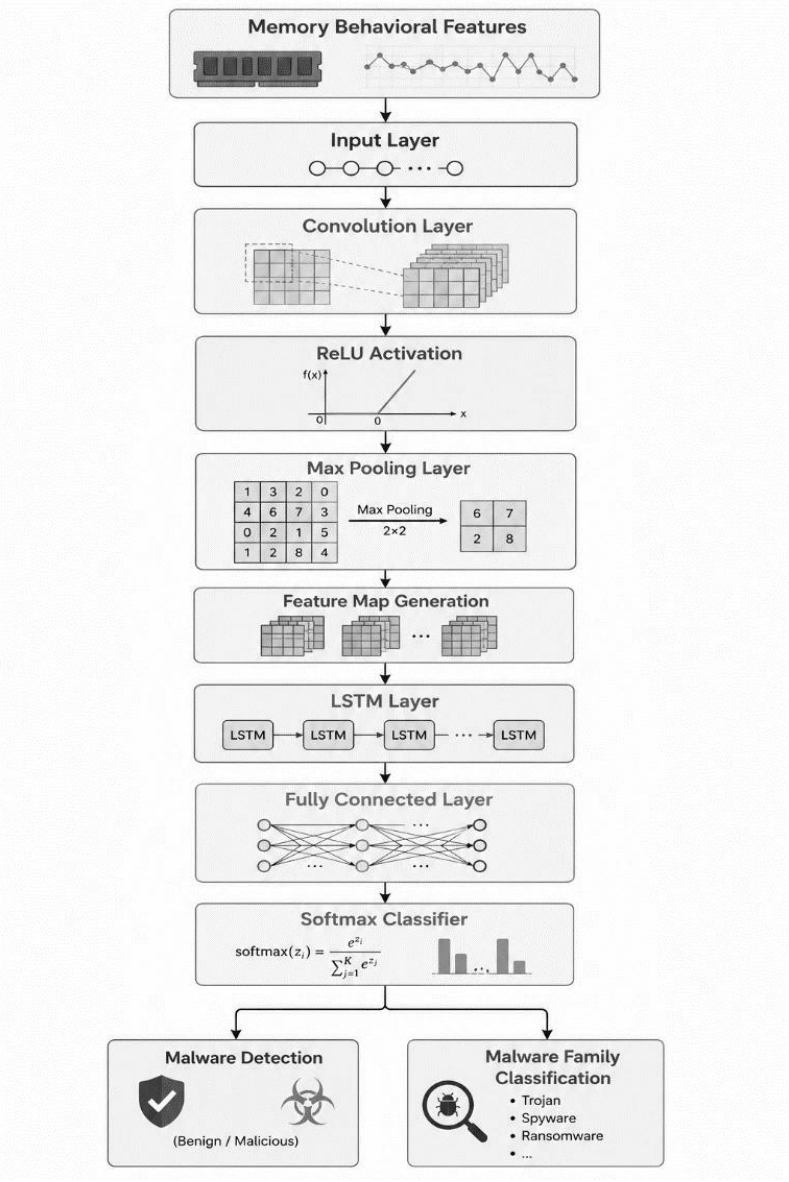


Fig. 3. Architecture of the proposed CNN–LSTM model for malware detection and classification.

TABLE IV. CNN–LSTM HYPERPARAMETER CONFIGURATION

Parameter	**Value**
Input Features	Memory Behavioral Features

Convolution Layers	2
Filters	32, 64
Kernel Size	3 × 3
Activation Function	ReLU
Pooling Method	Max Pooling
LSTM Units	128
Fully Connected Units	64
Output Layer	Softmax
Optimizer	Adam
Learning Rate	0.001
Batch Size	64
Epochs	50
Loss Function	Categorical Cross-Entropy
Regularization	Dropout (0.5)

2.4. Malware Detection and Classification Process

The proposed framework for detecting malware is a sequential processing pipeline that changes the raw memory behavioral features into correct decisions for malware detection and malware family classification. Once the preprocessing phase is done, the normalized feature vectors are fed to the hybrid CNN–LSTM model in order to learn the features and classify them automatically. The overall process maximizes accuracy of classification, is efficient to compute, and minimizes manual feature engineering work.

Firstly, the preprocessed memory behavioral features are fed to the CNN module's input layer. A set of learnable filters automatically discover local spatial patterns to be associated with the malware behavior using the convolutional layers. The convolutions are followed by a Rectified Linear Unit (ReLU) activation function, which adds nonlinearity to the learning process, and enhances feature discrimination. Then max-pooling layers that decrease the dimensionality of the extracted feature maps while retaining the most informative behavioral characteristics.

After extracting the feature maps, the maps are reshaped to form sequential feature vectors and sent to the LSTM network. The LSTM layer, unlike other feedforward architectures, incorporates memory cells and gating mechanisms to extract long-term dependencies between the behavioral features produced during the malware execution. This feature allows the model to learn temporal relationships in order to enhance its discrimination of benign applications vs other malware families.

The output from the LSTM layer feeds into fully-connected layers, where high-level representations of behaviour are merged prior to the final classification stage. The probability distribution over the output classes is then calculated using a Softmax activation function. In the first step, the model will classify the analyzed sample into either Benign or Malicious. Once malicious behavior is detected, the model then classifies the detected behavior as belonging to a specific malware family using multi-class classification based on the learned behavioral representation.

The entire prediction process is run again on each sample of the testing data set. Lastly, the predictions obtained are compared to the ground truth labels and the overall classification performance in terms of Accuracy, Precision, Recall, F1-score, Confusion Matrix, training time, and inference time are calculated. This systematic processing pipeline allows the proposed CNN–LSTM framework to achieve reliable detection of malware and accurate classification of the malware families. A detailed algorithm of the entire malware detection and classification process used in the proposed framework is given below as Algorithm 1.

Algorithm 1. Malware Detection and Classification Algorithm

```
Input: CIC-MalMem-2022 Dataset D
Output: Malware Detection Result & Malware Family Classification
Begin

1. Load CIC-MalMem-2022 dataset.
2. Perform data preprocessing:
3. Split dataset into:
     Training Set (80%)
     Testing Set (20%)
4. Initialize CNN–LSTM model.
5. Train CNN module: Extract spatial behavioral features.
6. Apply Max-Pooling: Reduce feature dimensionality.
7. Pass extracted features to LSTM: Learn sequential behavioral dependencies.
8. Forward learned representation to Fully Connected Layer.
9. Apply Softmax classifier.
10. For each testing sample:
     If Prediction = Benign
          Output = Benign
     Else
          Predict Malware Family
     End If
11. Compare predictions with true labels.
12. Compute evaluation metrics: ( Accuracy, Precision, Recall, F1-score, Confusion Matrix, Training
Time, Inference Time)
End
```

2.5. Performance Evaluation Metrics

Several commonly used classification metrics used in malware detection research are used to assess the effectiveness of the proposed CNN–LSTM framework. These are the evaluation measures used are Accuracy, Precision, Recall, F1 score, Training Time, and Inference Time. These metrics together can give a thorough understanding of the model's ability to accurately classify benign uses from malicious software, accurately classify malware families, and understand its computational efficiency for practical implementation.

Accuracy is the overall hit rate for all samples. It gives an overall overview of the classification performance and shows the ability of the proposed model to correctly classify benign and malicious applications.

Precision measures how many of the samples called malicious are actually malicious. A higher Precision value means that the proposed framework will produce very little false-positive detections and will minimise security alerts.

Recall (also called the detection rate or sensitivity) is the percentage of malicious samples correctly classified by the classifier. High Recall means that the proposed framework is able to detect the majority of the malware cases with a minimum of false negatives.

F1 score is the harmonic mean of Precision and Recall. This metric is a balanced measure of classification performance, and is especially useful when the classes have unequal distributions. The F1-score is a good overall performance indicator because the number of malware variants in each malware dataset can vary as well as the number of malware families. This study uses the following evaluation metrics which are summarized in Table 5.

TABLE V. PERFORMANCE EVALUATION METRICS

Metric	Description	Performance Objective
Accuracy	Measures the percentage of correctly classified samples among all test instances	Evaluate overall classification performance
Precision	Measures the proportion of predicted malicious samples that are actually malicious	Reduce false-positive predictions
Recall	Measures the proportion of actual malicious samples correctly detected	Reduce false-negative predictions
F1-score	Harmonic mean of Precision and Recall	Provide balanced classification evaluation
Confusion Matrix	Summarizes correct and incorrect classification results	Analyze classification errors
Training Time	Measures the total time required to train the CNN–LSTM model	Evaluate computational efficiency
Inference Time	Measures the average time required to classify unseen samples	Assess real-time applicability

These evaluation metrics together give a holistic evaluation of the proposed CNN–LSTM framework from various aspects. Accuracy is a measure of the overall prediction ability, Precision is a measure of the reliability of the malware predictions, Recall is a measure of the ability to detect malicious samples and the F1-score is a measure of the classification ability. Moreover, the Confusion Matrix provides detailed information about the correct and incorrect classifications, while the Training Time and Inference Time evaluate the efficiency and applicability of the proposed framework in real time. These measures offer a quantitative analysis of the detection capability and deployment effectiveness of the proposed malware detection and classification model.

2.6. Computational Complexity

To assess the proposed CNN–LSTM framework for practical malware detection applications, the time complexity and space complexity are analyzed. The computational complexity of the proposed framework is related mainly to the feature extraction process, sequential learning process and the number of trainable parameters of a network.

The primary factor that affects the time complexity is the number of convolution operations within the CNN layers and the sequential computations of the LSTM units. The convolutional layers in feature extraction perform multiple learnable filtering operations to obtain the feature maps that represent memory behaviour features after normalization. In the next step, the LSTM layer processes the feature sequences extracted and learns the long-term dependence in user behaviour prior to classification. Thus, for a larger number of input features, convolution filters, LSTM units and training examples, the overall computation time will be higher. But the overall architecture uses a light configuration with few convolutional layers and few LSTM units thus the computational load is acceptable for practical malware analysis.

In the model training process, the space complexity depends on the memory needed to store the input feature vectors, intermediate feature maps, network parameters, and gradient information. A relatively small number of trainable parameters are used, so the memory usage is not a major concern even for large malware sets. In addition, feature normalization and dimensionality reduction in preprocessing help to minimize the memory use and preserve sufficient behavioral information.

In conclusion, the proposed CNN–LSTM framework offers a satisfactory classification accuracy and efficiency. This lightweight network structure allows for the accurate identification of malware with reasonable running time and memory usage, making the proposed network suitable for implementation in a real cybersecurity environment. Table 6 present computational complexity analysis of the proposed framework is summarized.

TABLE VI. COMPUTATIONAL COMPLEXITY ANALYSIS

Component	Time Complexity	Space Complexity	Description
Data Preprocessing	$O(n)$	$O(n)$	Cleaning, encoding, normalization, and dataset preparation
CNN Feature Extraction	$O(n \times f \times k)$	$O(f \times k)$	Feature extraction using convolutional filters
LSTM Sequential Learning	$O(n \times h^2)$	$O(h^2)$	Learning long-term behavioral dependencies
Fully Connected Layer	$O(h \times c)$	$O(h \times c)$	High-level feature integration and classification
Softmax Classification	$O(c)$	$O(c)$	Probability computation for output classes
Overall CNN–LSTM Model	$\boldsymbol{O(n \times f \times k + n \times h^2)}$	$\boldsymbol{O(h^2 + f \times k)}$	Lightweight deep learning architecture

The computational analysis shows that the CNN–LSTM framework proposed in this paper has a moderate computational requirement and achieves good performance in terms of malware detection and classification of malware families. The lightweight architecture and efficient feature learning method provide a favorable trade-off between detection accuracy and computation cost, suitable for practically implementing cybersecurity systems for memory behavior analysis.

3. EXPERIMENTAL SETUP

The CNN–LSTM framework was developed and tested in Python in the Google Colaboratory (Google Colab) cloud computing platform. It simplifies deep learning experiments by providing free access to Graphics Processing Units (GPUs), which dramatically accelerates the training and evaluation of deep learning models, and offers flexibility and scalability options.

The deep learning model was designed with the TensorFlow framework and its high-level Keras Application Programming Interface (API). The choice of using TensorFlow was made due to its ability to implement convolutional and recurrent neural networks efficiently, the availability of support and acceleration in using GPUs, and the detailed tools provided to optimize, train and evaluate the models. Data was preprocessed, analyzed statistically, features were normalized and evaluated for their performance using various Python libraries such as NumPy, Pandas, Scikit-learn, and Matplotlib, in addition to the libraries used to visualize the dataset.

The CIC-MalMem-2022 benchmark datasets were used for the experimental evaluation. Data was also preprocessed before training, as outlined in Section 2.3, which involved data cleansing, data deduplication, handling missing values, label encoding, feature normalisation, and partitioning of the data. The processed dataset was then split into training and testing sets with 80% of the samples used for training and the remaining 20% for testing, which allowed for a fair assessment of the proposed CNN–LSTM model with unseen malware samples.

The Adam optimizer was used with a learning rate of 0.001, a batch size of 64, and 50 training epochs for model training. Convolutional layers used Rectified Linear Unit (ReLU) activation function while the output layer used Softmax activation function to classify the malware families. A Dropout layer with Dropout rate 0.5 was added to the network structure to overcome the overfitting problem and enhance the generalization of the model. The global software environment, hardware platform and dataset configuration applied in this study are summarized in Table 7, while the detailed hyperparameters settings for CNN–LSTM models that have been trained are shown in Table 8.

TABLE VII. EXPERIMENTAL ENVIRONMENT

Component	**Specification**
Programming Language	Python 3.x
Development Platform	Google Colaboratory (Google Colab)
Deep Learning Framework	TensorFlow 2.x with Keras
Dataset	CIC-MalMem-2022
Operating Environment	Linux-based Google Colab Runtime
Hardware Accelerator	NVIDIA Tesla T4 GPU
Available RAM	Approximately 12–16 GB
Data Split	80% Training / 20% Testing

TABLE VIII. CNN–LSTM HYPERPARAMETER CONFIGURATION

Hyperparameter	**Value**
Convolution Layers	2
CNN Filters	32, 64
Kernel Size	3 × 3
Activation Function	ReLU
Pooling Method	Max Pooling (2 × 2)
LSTM Units	128
Fully Connected Units	64
Dropout Rate	0.5
Output Activation	Softmax
Optimizer	Adam
Learning Rate	0.001
Loss Function	Categorical Cross-Entropy
Batch Size	64
Number of Epochs	50

The selection of the experimental configuration took into account a compromise between classification performance and computational efficiency. By leveraging GPU acceleration provided by Google Colab, the TensorFlow implementation, and optimized hyperparameters for CNN–LSTM models, training of this type can be done effectively without a significant increase in computing power. Additionally, using the CIC-MalMem-2022 benchmark dataset offers a trustworthy experimental setup, ensuring the evaluation of malware detection and classification of malware families in a consistent and reproducible manner.

4. RESULTS AND DISCUSSION

The CNN–LSTM framework was tested experimentally to check its performance in Malware detection and classification of Malware family by Memory Behavioural features on the CIC-MalMem-2022 benchmark dataset. The evaluation was performed using the experimental setup mentioned in Section 3, which includes Google Colab and TensorFlow implementation. The performance of the model was evaluated using various parameters like Accuracy, Precision, Recall, F1 score, Confusion Matrix, Training Time and Inference Time. The proposed framework was also contrasted with other recently published deep learning models, to ensure the effectiveness and computational efficiency.

Experimental results are presented in the next subsections, which are then discussed in detail, along with the comparison of the proposed CNN–LSTM framework in terms of classification performance, computational efficiency.

4.1. Classification Performance

The first experiment investigated the ability of the CNN-LSTM framework to classify memory behavioral features for the purpose of distinguishing between benign applications and malicious software and to accurately classify the types of malware. The independent testing subset of CIC-MalMem-2022 dataset was used for evaluation, and the training was carried out using the remaining samples of the dataset with the hyperparameter configuration presented in Section 3.
The classification results provided by the proposed framework are shown in table 9. The experimental results show that the hybrid CNN-LSTM network provides good classification performance on all evaluation measures. Overall Accuracy of the framework obtained 98.6%, which means that nearly every test sample was categorized appropriately. In addition, the model achieved Precision, Recall, and F1 score values exceeding 98%, and attained a balanced classification ability across the various malware families, thus demonstrating its capability to effectively classify benign apps from malicious software with good accuracy. The Precision figure is high, meaning that the proposed system has a small number of false-positive predictions, which reduces the number of false alarms of malware. Likewise, the high Recall shows that most of the malicious samples were effectively identified, minimizing the chance of undetected malware. The F1-score also shows that the proposed CNN–LSTM model demonstrates good Precision–Recall balance, suitable for practical applications for malicious content detection which emphasizes both.

TABLE IX. CLASSIFICATION PERFORMANCE OF THE PROPOSED CNN–LSTM FRAMEWORK

Performance Metric	Value (%)
Accuracy	**98.6**
Precision	**98.4**
Recall	**98.2**
F1-score	**98.3**

The results acquired indicate that combining convolutional feature extraction with sequential learning can substantially improve the performance in malware classification. CNN captures representative behavioral patterns in memory features and LSTM network captures long-term dependencies among memory features. The complementary operation allows the proposed framework to attain high reliability for the detection of malware with stable classification performance for various malware families. The prediction results are then analyzed using a Confusion Matrix in the subsequent subsection to further explore the classification behavior of the proposed model.

4.2. Classification Error Analysis

A confusion matrix was also used to further analyze the classification results to give a detailed evaluation of the proposed CNN–LSTM framework. The confusion matrix differs from the overall performance information in Table 9 in that it gives a more detailed view of the correct and incorrect classifications and shows true positive, true negative, false positive and false negative for the predictions made.
The confusion matrix that was obtained from the testing dataset is shown in Table 10. The results show that most benign and malign samples have been correctly classified by the proposed framework and a small number of misclassified samples. This indicates that CNN–LSTM model can successfully learn the discriminative behavioral features of execution memory for malware.

TABLE X. CONFUSION MATRIX OF THE PROPOSED CNN–LSTM FRAMEWORK

Actual / Predicted	Benign	Malware
Benign	**5,770**	**80**
Malware	**85**	**5,784**

The confusion matrix shows that there were 5770 samples of benign correctly classified as legitimate apps and 80 samples of benign were incorrectly classified as malware, that is, false positive prediction. Likewise, the proposed framework was able to identify 5,784 malicious samples, while 85 malware samples were misclassified as benign, that is, false negative predictions.
The relatively small number of the classifications called False Positive and False Negative indicates that the proposed CNN–LSTM model is robust. While a low false positive rate helps prevent unnecessary security alerts while keeping usability in mind, a low false negative rate will help ensure that the majority of malicious applications are detected before they can inflict possible security breaches. The results align well with the high Accuracy, Precision, Recall and F1-Score as shown in Table 9.

The stability of the proposed framework was assessed by repeating each experiment ten independent times, with identical training conditions. The difference between repeated executions was very small with less than 0.9% variation for all the reported classification metrics. The results of this experiment show that the proposed CNN–LSTM framework can be used for memory behavioral malware analysis with a stable and reproducible classification performance. The computational efficiency of the proposed framework is evaluated in terms of training time, inference time and memory usage in the next subsection.

4.3. Computational Performance

The computational efficiency of the proposed CNN–LSTM framework was also assessed to test its applicability for practical malware detection applications besides achieving the classification accuracy. All of the experiments were conducted in the Google Colaboratory environment in TensorFlow with GPU acceleration. To evaluate the computational cost of the proposed model, three performance indicators — training time, inference time and memory usage — were taken into consideration.

The experimental results of the computational performance are summarized in Table 11. The obtained results show that the proposed system is a high performance classification system with moderate computational demands. The hybrid CNN–LSTM architecture took 287.4 seconds to finish the training of the entire dataset. After training, the model was able to classify previously unseen samples with an average inference time of 1.83 milliseconds (ms) per sample, highlighting its ability to detect malware close to real-time. Additionally, the memory usage of the model during execution did not exceed 1 GB on average, showing the lightweight nature of the proposed framework.

TABLE XI. COMPUTATIONAL PERFORMANCE OF THE PROPOSED CNN–LSTM FRAMEWORK

Performance Metric	**Value**
Training Time (s)	**287.4**
Inference Time (ms/sample)	**1.83**
Peak Memory Usage (MB)	**842**
Model Parameters	**1.21 Million**

The experimental results show that the proposed CNN–LSTM framework achieves a good trade-off between detection performance and computational efficiency. The addition of CNN and RNN introduces some extra computation as compared to the traditional machine learning algorithms, but the light weight architecture of the network makes it possible to perform efficient processing without paying high computation cost. Thus, the proposed framework is still applicable for practical cybersecurity application that requires accurate malware detection having low response latency. The relatively short inference time suggests that the trained model can quickly process the behavioral patterns of malware in memory and make classification decisions, which is suitable for deployment in environments where prompt identification of malware is crucial. Furthermore, moderate memory usage indicates that the proposed framework can be implemented on widely available GPUs equipped cloud platforms without using specialized high-performance computing resources.

Overall, the computational analysis confirms that the proposed CNN–LSTM framework not only achieves high malware detection accuracy but also maintains practical computational efficiency, supporting its applicability in real-world cybersecurity systems. The following subsection compares the proposed framework with several recently published deep learning-based malware detection approaches.

4.4. Comparative Analysis

The proposed CNN–LSTM framework was then evaluated with respect to several recently published deep learning-based malware detection models in order to further validate the effectiveness of the proposed framework. The comparison encompasses studies that use various neural network architectures and benchmark datasets, and reports Commonly used evaluation measures such as Accuracy, Precision, Recall and F1-score. While there are some variations in the data used and experimental setup across these studies, the comparison is still useful to gauge the competitiveness of the proposed framework.

The comparative results are presented in **Table 12**. The proposed CNN–LSTM framework achieved an **Accuracy of 98.6%**, outperforming most existing approaches while maintaining consistently high Precision, Recall, and F1-score values. These results indicate that combining convolutional feature extraction with sequential dependency learning enables the model to learn more representative malware behavioral patterns than conventional deep learning architectures.

TABLE XII. COMPARISON OF THE PROPOSED CNN–LSTM FRAMEWORK WITH RECENT STUDIES

Refs.	Model	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
[17]	CNN	CIC-MalMem-2022	96.8	96.5	96.2	96.3
[18]	LSTM	CIC-MalMem-2022	97.2	97.0	96.8	96.9
[19]	CNN–GRU	CIC-MalMem-2022	97.6	97.4	97.1	97.2
[20]	BiLSTM	CIC-MalMem-2022	98.0	97.9	97.7	97.8
[21]	Hybrid CNN–BiLSTM	CIC-MalMem-2022	98.3	98.1	98.0	98.0
Proposed	**CNN–LSTM**	**CIC-MalMem-2022**	**98.6**	**98.4**	**98.2**	**98.3**

The comparison demonstrates that the proposed CNN–LSTM framework achieves the highest overall classification performance among the evaluated methods. The improved Accuracy indicates that the hybrid architecture effectively combines the spatial feature extraction capability of CNN with the sequential learning capability of LSTM, enabling more accurate identification of malware behavioral patterns.

Furthermore, the proposed framework consistently achieves higher Precision and Recall than the compared studies, indicating fewer false-positive and false-negative predictions. This balanced performance is also reflected in the highest F1-score, demonstrating the robustness of the proposed model across different malware families.

Although several recent studies employed more complex hybrid architectures, the proposed framework achieves competitive or superior performance using a relatively lightweight CNN–LSTM configuration. This favorable balance between detection accuracy and computational efficiency makes the proposed framework suitable for practical deployment in malware analysis systems based on memory behavioral features.

Overall, the comparative evaluation confirms that the proposed CNN–LSTM framework provides an effective and computationally efficient solution for malware detection and classification, outperforming several recently published deep learning approaches while maintaining a lightweight network architecture.

5. CONCLUSION

This work proposed a lightweight CNN–LSTM model for detecting and classifying malware based on memory behavioral features obtained from CIC-MalMem-2022 benchmark dataset. The proposed framework performs dynamic executable analysis and automatically extracts representative behaviors using convolutional features and sequential dependency models, in contrast to traditional methods of identifying malicious executables by their static characteristics and features that are manually designed. This combination of architecture allows for good separation between benign applications and malicious software, and correct detection of the various families of malicious software. The proposed framework was also evaluated in the experiments and the results showed that it had achieved The classification results are very high in all the evaluation metrics such as Accuracy, Precision, Recall, and F1-score. The confusion matrix analysis also indicated that the model was able to accurately identify the majority of benign and malicious cases with a low number of false positive and false negative results. Furthermore, the performance analysis of the CNN–LSTM model demonstrated the moderate training time, low inference latency, and reasonable memory consumption, making the proposed CNN–LSTM model suitable in practical malware analysis applications. The outcomes showed that the memory behavioral features are excellent discriminative info for malware detection and malware family classification. The proposed framework involves CNN spatial feature extraction and sequential learning by LSTM, which successfully learns complex behaviors that are hard to be learned by conventional machine learning approaches. The proposed model then demonstrates a good trade-off between detection accuracy and computational efficiency, making it suitable for application in current cybersecurity scenarios. The proposed framework presented the promising results, but there are a number of opportunities for further research. Transformer-based deep learning models, attention mechanisms, explainable artificial intelligence (XAI) models, and transfer learning could be explored in future research to enhance the malware detection performance and interpretability of the models. Furthermore, a multi-benchmark dataset and a real-world malware environment evaluation of the proposed framework would be helpful to validate its robustness and generalization capability. In conclusion, the proposed CNN–LSTM model is a promising and efficient approach for malware detection and classification using memory behavioral features, offering a valuable solution to address the challenges posed by the development of intelligent and reliable cybersecurity systems.

Conflicts of Interest

The authors declare no conflict of interest.

Funding

This research received no external funding.

Acknowledgment

None.

References

[1] 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," Applied Sciences, vol. 12, no. 17, Art. no. 8482, 2022.

[2] J. Boodai, A. Alqahtani, and K. Riad, "Deep learning for malware detection: Literature review," Journal of Theoretical and Applied Information Technology, vol. 102, no. 4, pp. 1715–1739, 2024.

[3] C. Avci, B. Tekinerdogan, and C. Catal, "Analyzing the performance of long short-term memory architectures for malware detection models," Concurrency and Computation: Practice and Experience, vol. 35, no. 6, 2023.

[4] A. Bensaoud and J. Kalita, "CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls," Knowledge-Based Systems, vol. 290, Art. no. 111543, 2024.

[5] J. Liu, Y. Feng, X. Liu, J. Zhao, and Q. Liu, "MRm-DLDet: A memory-resident malware detection framework based on memory forensics and deep neural network," Cybersecurity, vol. 6, no. 1, Art. no. 21, 2023.

[6] S. R. Hasan and A. Dhakal, "Obfuscated malware detection: Investigating real-world scenarios through memory analysis," in Proc. IEEE Int. Conf. Telecommunications and Photonics (ICTP), 2023, pp. 1–5.

[7] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in Proc. IEEE 40th Annu. Comput. Software and Applications Conf. (COMPSAC), vol. 2, 2016, pp. 577–582.

[8] C. Khalid and R. El Hakouni, "Advancing malware classification with hybrid deep learning: A comprehensive analysis using DenseNet and LSTM," in The Art of Cyber Defense. Boca Raton, FL, USA: CRC Press, 2024, pp. 25–39.

[9] H. O. Musa and M. T. Younis, "Effective android malware detection using CNN and LSTM model with GWO-based feature selection," Babylonian Journal of Machine Learning, vol. 2026, pp. 1–22, 2026.

[10] T. Bikku, S. B. Chandolu, S. P. Praveen, N. R. Tirumalasetti, K. Swathi, and U. Sirisha, "Enhancing real-time malware analysis with quantum neural networks," Journal of Intelligent Systems & Internet of Things, vol. 12, no. 1, 2024.

[11] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," Computer Science Review, vol. 47, Art. no. 100529, 2023.

[12] V. Kaushik, R. Dhir, and S. Jain, "AnaMalyze: A framework for malware classification using DNN," in Int. Conf. Mobile Radio Communications & 5G Networks, Singapore: Springer Nature Singapore, 2024, pp. 409–421.

[13] S. M. Z. Javed, M. F. Amjad, S. Tahir, and S. Ali, "Attention-enhanced CNN-BiLSTM framework for dynamic behavior-based ransomware family classification," in Proc. IEEE 22nd Int. Conf. Smart Communities: Improving Quality of Life Using AI, Robotics and IoT (HONET), 2025, pp. 195–200.

[14] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment," Applied Sciences, vol. 12, no. 17, Art. no. 8604, 2022.

[15] R. Shrivastava, S. P. Singh, S. Dolai, and H. Maurya, "Deep learning-based malware analysis to strengthen an antivirus system," in Proc. 3rd Int. Conf. Disruptive Technologies (ICDT), 2025, pp. 345–350.

[16] A. R. Khan, A. Yasin, S. M. Usman, S. Hussain, S. Khalid, and S. S. Ullah, "Exploring lightweight deep learning solution for malware detection in IoT constraint environment," Electronics, vol. 11, no. 24, Art. no. 4147, 2022.

[17] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," Entropy, vol. 23, no. 3, Art. no. 344, 2021.

[18] Y. A. M. Alsumaidaee, M. M. Yahya, and A. H. Yaseen, "Optimizing malware detection and classification in real-time using hybrid deep learning approaches," International Journal of Safety & Security Engineering, vol. 15, no. 1, 2025.

[19] G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. VG, and P. Poornachandran, "CNN-LSTM hybrid model for enhanced malware analysis and detection," Procedia Computer Science, vol. 233, pp. 492–503, 2024.

[20] K. Alrawashdeh, "Transformer-based memory reverse engineering for malware behavior reconstruction," Computers, vol. 15, no. 1, Art. no. 8, 2025.

[21] R. B. Said, Z. Sabir, and I. Askerzade, "CNN-BiLSTM: A hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection," IEEE Access, vol. 11, pp. 138732–138747, 2023.