RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt26511

A Hybrid CNN–LSTM Deep Learning Framework for Automated Malware Classification using Spatial–Sequential Feature Fusion

Chevella Anil Kumar [a,*], S. Saradha [b], Mithila Ayyavoo [c], S. Durga Devi [d], H.R. Lokesha [e], M. Jeevaa [f]

[a] Department of Electronics and Communication Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, India
[b] Department of Computer Science and Applications, SRM Institute of Science and Technology, Ramapuram, Chennai, India
[c] Department of Computer and Communication Engineering, Kathir College of Engineering, Coimbatore 62, India
[d] Department of Computer Engineering, Vel Tech High Tech Dr.Rangarajan Dr Sakunthala Engineering College, 60, Avadi Vel Tech Road Vel Nagar Avadi, Chennai, Tamil Nadu, India
[e] Department of Electronics and Communication, The National Institute of Engineering South, Mananthavady Road, Mysore, Karnataka – 570008, India
[f] School of Computing and Information Technology, REVA University, Bangalore, India
* Corresponding Author Email: anilkumarassistantprofessor@gmail.com
DOI: https://doi.org/10.54392/irjmt26511
Received: 26-11-2025; Revised: 04-09-2026; Accepted: 17-09-2026; Published: 22-09-2026

Abstract: The high rate of growth of advanced and highly obfuscated malware has made the use of conventional signature detection mechanisms less viable. In order to overcome this problem, this paper suggests a hybrid framework for malware classification based on the combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to learn spatial and temporal features simultaneously. The given model converts malware feature vectors into structured grayscale representations from which CNN layers extract spatial features, and sequential opcode-related patterns are learned by an LSTM branch. Aspects of both domains are combined into a single embedding and used to classify nine malware families. Tests that have been carried out with a benchmark dataset show that the hybrid CNN-LSTM model attains a classification accuracy of 97.27% which is very high compared to baseline LSTM-only and CNN-only frameworks. The model demonstrates strong generalization, with weighted precision, recall, and F1-score values above 97% and highly discriminative ROC-AUC scores (reaching 1.000 for major classes). An in-depth analysis, such as confusion matrix analysis, precision-recall curves, and a comparison with the state-of-the-art methodologies demonstrate that the proposed architecture achieves performance comparable to that of leading malware detection models reported in recent literature. These findings validate the power of hybrid feature fusion to capture both static and dynamic behavioral traits of malware to provide a powerful, scalable, and highly accurate solution for next-generation cybersecurity systems.

Keywords: Malware Classification, CNN–LSTM Hybrid Model, Microsoft Malware Classification Challenge (BIG 2015), Deep Learning (DL), Cybersecurity, Opcode Sequences, Grayscale Image Features.

1. Introduction

The cybersecurity landscape has become broader and more complex as the number and scale of digital services, cloud, enterprise information systems and Internet of Things (IoT) infrastructure have grown. Malware is one of the most persistent threats as it can be used to steal private information, disrupt critical services, decrease system availability, and open up access to critical assets. Contemporary malware includes ransomware, spyware, banking Trojans, botnets, and rootkits, with many variants that are packed, encrypted, polymorphic, metamorphic, or otherwise obfuscated. Hence, malware classification goes beyond identifying the family of an unknown malicious sample; it should also help analysts infer the sample's likely behavior and determine an appropriate response. Family-level classification also helps with incident triage, threat intelligence, containment planning and forensics if it occurs in a timely fashion. This is especially important in industrial, financial, healthcare, governmental and cloud environments, where the outcome of the successful compromise could have operational and economic impacts. Traditional anti-virus solutions are heavily based on the utilization of signatures, heuristic rules and manually written behavioral indicators. While these mechanisms are still effective against known threats, they lose their

effectiveness when malware writers change code organization or generate new variants that haven't been seen before. Signature databases must therefore be continuously updated, while creating new rules requires considerable time and expert effort. In the face of the diversity of malware, the use of Artificial Intelligence (AI) and Deep Learning (DL) which can learn discriminative representations directly from the executable file, malware images, opcode sequences, numerical attributes, or behavior traces has been promoted. Among these, hybrid CNN-LSTM model is more significant as CNN (Convolutional Neural Network) can be applied to learn the local structural pattern and LSTM (Long Short-Term Memory) can be applied to learn the relationship in an ordered sequence. Previous research has demonstrated that these mechanisms can be used together to provide more effective malware detection than either mechanism alone [1]. However, the proper representation and combined use of complementary malware information to build reliable hybrid classifiers still remains to be accomplished.

In the era of sensors and Internet of Things (IoT) that are connected and constrained by resources, this need is more urgent. IoT ecosystems are constantly uploading and downloading data over varying networks, which adds to the many points of entry for malware and makes manual monitoring impossible. Deep neural models are able to automatically learn informative patterns from IoT malwares and have shown enhanced detection capability [2]. But models created for a specific IoT context may not work for other executable platforms or malware variants, and the time required to run them may be excessive and a burden on these resource-constrained devices. Hybrid solutions are also examined in Software Defined Networks, which refers to the combination of DL models to improve the benign and malicious traffic discrimination [3]. However, traffic-based techniques don't always take advantage of the structural characteristics found in executable malware. However, there has also been work on explainable DL, which helps to make its decisions more transparent when applied to IoT malware [4] but, interpretability does not address the need for a unified representation that captures both the local structure and the contextual dependency.

There are a few groundbreaking works that show the usefulness of representation learning for malware analysis. DeepAM was used to combine multiple tasks from various domains of DL to learn complementary attributes of malware samples and improve the discrimination between similarly behaving malware families [5]. It was aimed at general (heterogeneous) feature learning, not at the specific association of image like spatial patterns and ordered numerical relations. Deep networks might be used to learn useful representations from the malware binaries and reduce reliance on manually written rules as was done in DL4MD [6], but it did not explicitly coordinate multiple learning branches for complementary feature domains. Deep4MalDroid modeled information about the behavior of the Android applications, and it managed to effectively detect complex relationships [7]. However, because of its platform specific and behavior oriented design, it cannot be applied to executable, heterogeneous malware directly and the dynamic analysis approach has additional execution and monitoring overhead. Another way towards improvement and generalization of malware models is multi-task learning. MtNet jointly optimized related malware-classification objectives, and further increased the information sharing between tasks [8]. It primarily concentrated on dynamic analysis of malware, but didn't explicitly combine image-based spatial learning with sequential feature modelling. Existing research demonstrates the potential of DL but also reveals three ongoing challenges. First, many methods rely on a single representation and therefore capture only part of the structural, statistical, or behavioral information associated with a malware family. Second, some hybrid systems report higher accuracy but do not explicitly examine how heterogeneous features are fused. Third, application-specific solutions that are developed for IoT traffic, Software-Defined Networks or Android apps can be challenging to map to general classification of malware families. A single source of features that learns complementary spatial and sequential information in a unified, and computationally manageable architecture is still required. This study aims at filling that gap by proposing a hybrid CNN – LSTM based automated malware-family classification model. Each malware sample is represented by a fixed-length 68-dimensional feature vector. After duplicate removal, label encoding, standardization, and a stratified train-test split, the resulting feature vector is transformed into two coordinated inputs. For CNN, the vector is also padded and reshaped to a 10 x 10 single channel pseudo-image, and the CNN can be used to learn the local relationship between feature values of neighbouring areas. In the LSTM branch, the ordered vector is flattened into a one dimensional sequence, so the model is able to learn contextual relationship between the position of features. The spatial embedding generated by the CNN and the sequential embedding generated by the LSTM are concatenated in a feature-fusion layer. The joint representation is further refined through fully connected layers and a Softmax classifier is applied to classify the sample as a member of one of the nine malware families. Thus, the workflow maintains a common source of information from which the two learning mechanisms extract complementary representations.

The proposed design is based on the complementary strengths of the two branches. The CNN captures local interactions between structures in the pseudo-image representation, and the LSTM captures longer range interactions among features in the ordered feature sequence. Neither representation is a literal

malware image or time series; rather, both are structured transformations of the same feature vector. Combining these representations is intended to improve discrimination among malware families with similar characteristics, particularly when their feature patterns overlap across the CNN and LSTM representations. The framework is evaluated on the processed Microsoft Malware Classification Challenge (BIG 2015) dataset, which is also used for the baselines of random forest, standalone CNN and standalone LSTM, without changing the data partition, and with the same data pre-processing procedure. Evaluation is carried out by means of accuracy, precision, recall, F1 score and confusion matrix analysis/multi-class ROC measures.

The key findings of this research are:

- The complementary spatial and sequential representations of malware are generated using the proposed dual-branch CNN–LSTM model in an end-to-end classification model.
- We propose a dual-input transformation without the need of two separate image and ordered sequence data sources, which is treated as pseudo-image input and ordered sequence input with the same 68-dimensional feature vector.
- A feature-fusion mechanism is used to fuse the learned embeddings before the final malware-family classification to better discriminate similar feature patterns of classes.
- The framework is evaluated on nine malware families using BIG 2015 benchmark on multiple classification measures against conventional machine learning and single branch deep learning baselines.

The rest of this paper is structured in the following manner. In Section 2, the related works on deep learning based malware classification are presented. Section 3 describes the dataset, the pre-processing procedure, the construction of the dual-input and the hybrid network, as well as the feature-fusion mechanism and the training objective. The experimental set up is presented in section 4 and the comparative results are discussed. The study concludes in Section 5 with suggestions for further research.

2. Literature Survey

Recent malware-detection research has increasingly shifted toward deep-learning models with the ability to automatically recognize the hidden patterns in raw data as compared to the previous method of signature and heuristic detection. Hybrid networks combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks have turned out to be potent in the ability to both capture spatial and sequential properties of malware. These models have been successfully applied to analyze grayscale image representations of binary files as well as opcode or API call sequences to improve the accuracy of classification. Earlier deep-learning models primarily focused on either static or dynamic analysis, which limited their ability to generalise to novel or obfuscated malware types. More recent studies have proposed hybrid and explainable models, which enhanced the detection strength, scalability, and interpretability. In spite of these developments, challenges remain in optimizing feature fusion, reducing computational complexity, and improving generalization across diverse malware families. To fill these gaps, the current paper has suggested a superior CNN-LSTM hybrid model that incorporates both spatial and sequential feature spaces to effectively and efficiently classify malware.

Most recent developments in deep learning have demonstrated significant improvements in the detection of malware, where discriminative features from different heterogeneous malware representations are automatically extracted. To improve the accuracy of malware detection, Kim *et al.* [9] suggested a multimodal deep learning framework that combines various static feature sources such as application permissions, API calls, and information obtained from Android manifest. The framework was able to improve the performance of classification by utilizing heterogeneous feature modalities. The proposed method was mainly designed for android apps and manually extracted feature modalities made it difficult to be applied for generalized malware classification for various executable platforms. Dynamic malware analysis has been a significant focus of interest too, as behavioural data can be extremely useful in the discovery of malicious behaviour that may be undetected by static analysis. Kolosnjaji *et al.* [10] proposed a Deep Learning framework by integrating Convolutional neural network and recurrent neural network, which is used to classify Malware based on system call sequences. Their approach was able to effectively capture the temporal behavior patterns, and to show that they were able to detect previously unknown variants of malware by modelling sequential system events, as opposed to relying on a pre-defined signature. However, dynamic analysis necessitates malware execution within controlled settings, which adds extra execution latency and processing overhead, potentially hindering its application in large-scale cybersecurity systems. The success of convolutional neural networks (CNNs) in computer vision inspired their use for malware-image analysis. Kodamanchili *et al.* [11] proposed the architecture of AlexNet and showed that hierarchical spatial representation can be automatically learned using large-scale image datasets with significantly higher accuracy than the traditional approach, where the features are engineered. Although the study was initially designed for the natural image classification problem, its hierarchical feature learning power later inspired many approaches towards malware

image analysis, in which the executable binaries are converted into gray level images. CNN-based models, however, mainly represent local spatial patterns and lack the explicit modeling of long-range contextual dependencies between the behavior of the malware characteristics, which hinders the representation of complementary behavioral information.

In recent years, hybrid deep learning architectures have been the subject of numerous studies that aim to leverage complementary representations of malware. Alshoulie and Mehmood [12] gave a thorough survey on deep learning techniques for malware detection, including CNN, RNN, LSTM, Transformer and hybrid architectures. It was found that hybrid deep learning models always outperform single-network models by integrating complementary characteristics of malwares. The authors also pointed out some unaddressed issues such as good feature fusion, resisting obfuscated malware, computational efficiency, and the generalization of models for different families of malware. The study, however, does not introduce or test a common malware classification model, but rather serves as a survey of existing approaches. These limitations push toward the proposed hybrid CNN-LSTM architecture that tackles complementary feature learning of both spatial and sequential features in an end-to-end manner. In [13] Li *et al.* have introduced a machine learning system for detecting Android malware using permission analysis and feature selection to extract most informative application permissions for malicious applications. They made the approach more efficient in terms of detection and less complex, with a dimensionality reduction. The framework, however, heavily depended on manually engineered static permissions and traditional machine learning methods without utilizing large-scale malware datasets to automatically learn complex hierarchical feature representations. Opcode based Malware Analysis has proven to be a very effective method to capture the instruction level attributes of malicious software. Sujon *et al.* in [14] presented OPTISTACK: an explainable ensemble learning framework integrated with XAI for malware detection. It proposes to combine classifiers using a stacking ensemble in order to improve detection accuracy, and also apply explainability techniques to make model predictions transparent and interpretable. While the framework showed enhanced classification performance and analyst trust, it is built on top of ensemble machine learning rather than end-to-end deep feature learning. Additionally, it does not simultaneously leverage complementary spatial and sequential malware representations. Motivated by these limitations, this study proposes a hybrid CNN–LSTM framework, which directly integrates convolutional and recurrent learning to automatically extract richer malware features while improving classification robustness across various kinds of malware families. The first use of image-based malware analysis was by Nataraj *et al.* [15] who converted the binary files of malware into grayscale images and found that various families of malware exhibit unique visual patterns of texture that are easily detectable through image processing techniques. This paper laid the foundation for many CNN-based malwares classification techniques by demonstrating that malware binaries contain discriminative structural information that is preserved in visual transformations. Though important, image-based analysis mainly relies on structures and ignores the complementary sequential and contextual information which can help discriminate further among malware families. Table 1 shows the summary of literature survey.

The novelty and contribution of this work can be summarized as follows: Whereas conventional malware classifiers are typically based on either static image characteristics or sequential opcodes, the proposed model combines the two feature domains simultaneously.

- CNN layers obtain spatial structural patterns of feature-derived grayscale embeddings.
- LSTM layers learn ordered dependencies and opcode-related behavioral patterns.

The dual-path design captures complementary malware features that may not be learned when CNN and LSTM models are used separately. Despite the current developments of applying deep learning in malware detection, the majority of research is confined to either the static or dynamic feature space, which is not a comprehensive framework that will adequately represent the complex spatial and sequential nature of malware. The CNN-based image classifiers and LSTM-based opcode analyzers models have shown good results on single instances and fail to generalize well across different malware families and obfuscated versions. Further, most of the hybrid structures used in the literature are those that use a basic concatenation of features without optimization of the fusion process thus losing the inter-domain contextual information. In order to address such constraints, the current article suggests an improved CNN-LSTM hybrid architecture that incorporates both the image-based and opcode-based features that are conducted in a sequence and combined into a single learning model. The objective of this approach is to promote the classification precision, the resistance to the changing malware trends, and the scalability of the approach to the real-world cybersecurity implementations [16].

3. Proposed Method

The suggested approach presents a hybrid deep learning system where Convolutional Neural Networks (CNNs) and Long Short-term Memory (LSTM) networks are combined in order to enhance automated classification of malware.

Table 1. Summary of the Literature Survey

Author/Year	Data Type	Method	Application Focus	Key Findings
Thakur *et al.* (2024) [1]	Malware images, executable features, and network-traffic information	CNN–LSTM with PCA-based feature reduction and voting classification	Hybrid malware-family classification	Combined spatial and sequential learning to improve accuracy, precision, recall, and F1-score. However, the multistage PCA and voting pipeline increased computational complexity.
Abdullah *et al.* (2025) [2]	Microsoft Malware Dataset and IoT Malware Dataset	Lightweight 1D/2D CNN and GRU models with visual interpretation	Resource-efficient IoT malware analysis	The 1D models provided better accuracy, parameter efficiency, and training time than the 2D models. However, the two representations were evaluated independently rather than jointly fused.
Ye *et al.* (2018) [5]	Windows PE files represented using API-call features	DeepAM using Autoencoder, Restricted Boltzmann Machines, and associative memory	Heterogeneous malware-feature learning	Outperformed shallow and homogeneous deep-learning models. However, it focused on API-call-based binary detection without spatial–sequential feature fusion.
Nataraj *et al.* (2011) [15]	Malware binaries converted into grayscale images; 9,458 samples from 25 families	GIST-based texture feature extraction and image classification	Image-based malware-family classification	Achieved approximately 98% accuracy and demonstrated that malware families exhibit distinctive visual patterns. However, sequential dependencies were not considered.
Pascanu *et al.* (2015) [16]	Executed instruction-event sequences from Windows malware	Echo State Network, RNN, max-pooling, and logistic regression	Sequence-based malware detection	Improved the true-positive rate by 98.3% over the trigram model at a 0.1% false-positive rate. However, the approach required execution-derived sequential data.
Proposed Work	Processed BIG 2015 dataset containing 2,158 samples, 68 numerical features, and nine malware families	Dual-input CNN–LSTM with pseudo-image transformation, ordered feature modeling, and feature-level fusion	Multiclass malware-family classification	Achieved 97.27% accuracy and a 97.24% weighted F1-score by learning complementary spatial and sequential representations. Minority-class imbalance and external-dataset validation remain limitations.

The proposed system is able to obtain spatial and temporal discriminative features in parallel and hence provide a richer description of malware behavior patterns as opposed to the traditional methods that rely only on either the static or sequential information. The procedure starts with a preprocessing of the numerical malware feature vectors by normalization and label encoding and then a two-input transformation. Every normalized feature vector is reshaped into (i) a 10x10 pseudo-image in which spatial learning will be conducted and (ii) a 68x1 sequence for learning dependencies among the ordered features. The CNN branch in the first learning pathway takes the 2D pseudo-image and the stacked convolution and max-pooling layers are used to detect the local structural relationship among features. The resulting spatial embedding of this branch is a compact 128 dimensional embedding.

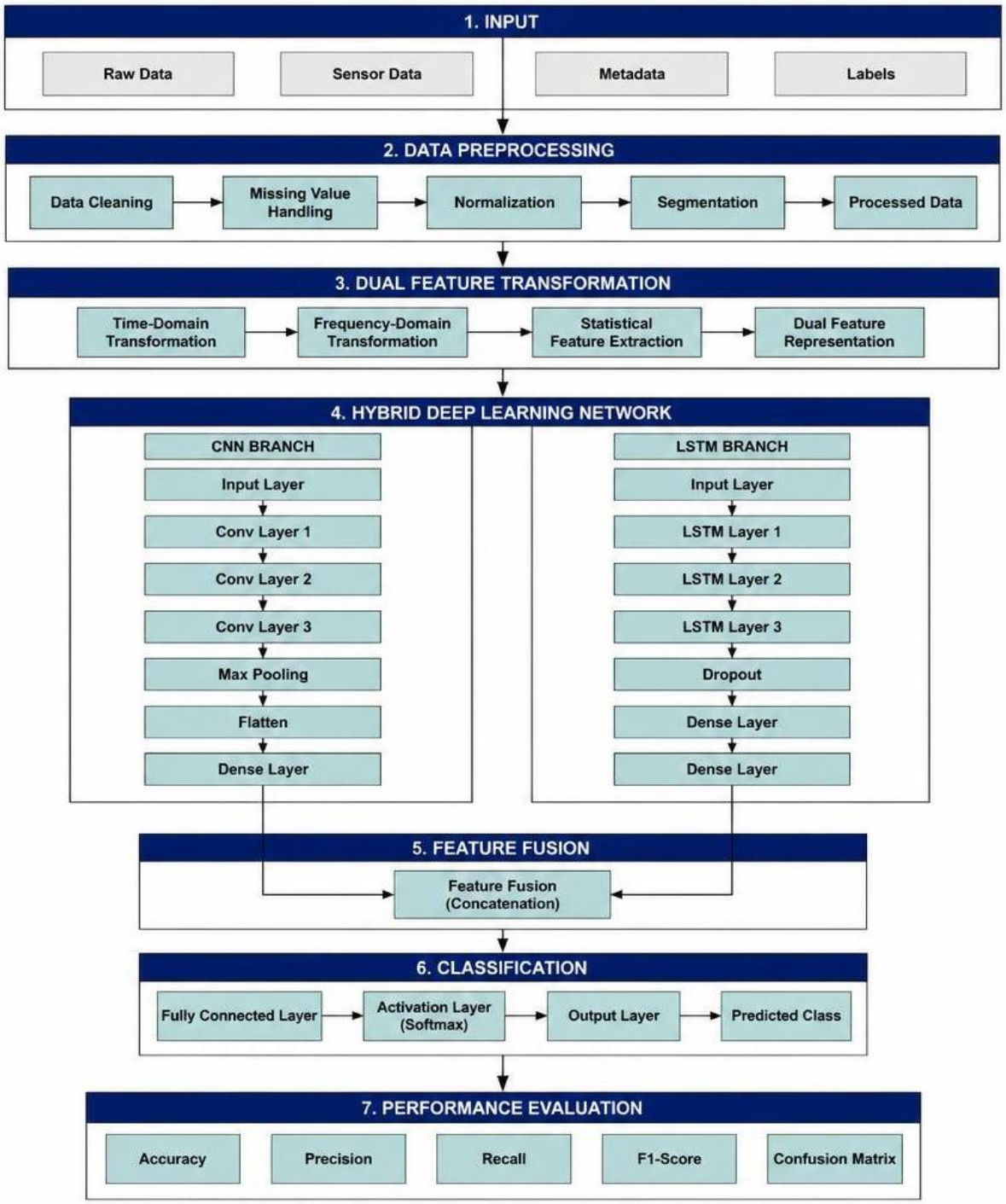

Figure 1. Shows the workflow of proposed model.

Simultaneously, the LSTM branch takes the form of the sequential representation of the same feature vector. A conv1D-maxpooling1D pipeline is used to get to the LSTM layer which allows the network to capture both short and long-range dependencies in the feature sequence. The two embeddings are then concatenated and create a single representation of spatial-temporal features. A dropout-regularized fully connected layer further refines the combined embedding, and a softmax classifier assigns each sample to one of nine malware families [17]. The model is trained end-to-end using the Adam optimizer and sparse categorical cross-entropy loss, while convergence is monitored using validation accuracy and loss. By jointly learning structural patterns and ordered feature dependencies, the model combines complementary spatial and sequential information and achieves better classification performance than the standalone CNN or LSTM models. The hybrid architecture has the added benefit of generalization, robustness on imbalanced classes, and high accuracy suitable for real-world cybersecurity applications. The work flow of the proposed model is represented in the Figure1.

The extracted malware representation is not a traditional temporal sequence, but rather a fixed-length numerical feature vector, but the ordering of the features in the 68 features is done following a fixed feature extraction pipeline. Other descriptors like PE header attributes, section statistics, entropy measures, imported API info and structural metadata are gathered in contiguous blocks according to the semantic content. As a result, there are contextual relationships between neighbouring features that can be utilized by sequential learning models. The LSTM branch is thus used to learn long-range dependencies and interactions between these ordered feature groups instead of temporal dynamics. At the same time, the CNN branch is required to learn the local spatial correlations among the neighboring features in the pseudo-image representation. Finally, the outputs of the two branches of CNN and LSTM are combined to generate a more discriminative and comprehensive malware representation, resulting in better classification accuracy [18].

3.1 Dataset Description

The experiments use a benchmark malware dataset based on the Microsoft Malware Classification Challenge (BIG 2015), which is provided through its processed form, which is available on the Kaggle platform. This data is a set of numerical feature representations of malware binaries of nine well-known malware families, as well as their variants, such as Ramnit, Lollipop, Kelihos, Vundo, Simda, Tracur, Obfuscator, and Gatak. The malware instances are represented as a fixed-length numeric feature vector representing the statistical, structural, and opcode related features of the original file. The dataset also contains the thousands of labeled samples, which means that it can be used with supervised learning and deep neural network training. The one used in this work provides standardized 68-dimensional feature vectors and a class label of 1 to 9, which depicts a malware family. The sample is moderately imbalanced, with some families having significantly lower numbers of samples (e.g., Class 5), as this is a realistic threat distribution in the real-world as some families have a higher spread. Such diversity makes it possible to evaluate the strength and the generalization ability of the proposed hybrid deep learning model. The dataset contains 2,158 samples from nine malware families, as shown in Table 2. This study uses 68-dimensional numerical feature vectors obtained from a preprocessed version of the Microsoft Malware Classification Challenge (BIG 2015) dataset available on Kaggle. The authors do not extract these features but give them as a part of the benchmark dataset. The feature set is statistical, structural, and opcode-based features based on raw malware binaries and their disassembled counterparts. In order to achieve reproducibility the 68 features are clustered into coherent categories including byte-level statistics, entropy-based measures, structural metadata, and opcode frequency characteristics [19]. Samples of malware are encoded as a 68-dimensional feature vector of numerical attributes derived from the executable files. These properties are complementary to each other and they describe the structural properties, statistical properties and behavioural properties of malware binaries. Instead of being made of raw executable bytes, the proposed framework is based on engineered numerical descriptors summarizing information contained in Portable Executable (PE) header, section level statistics, entropy measures, features of imported API, byte-frequency distribution, statistics of printable strings, and other structural metadata. This feature representation has two purposes: to capture the informative and compact description of each malware sample as well as to avoid high computation complexity in training the deep learning model [20]. Because this study used only publicly available malware data, no human participants were recruited or involved, and no personally identifiable information was processed.

Table 2. Dataset Distribution

Class	Malware Family ID	Support (Samples)
1	Class 1	301
2	Class 2	496
3	Class 3	589
4	Class 4	95
5	Class 5	8
6	Class 6	144
7	Class 7	79
8	Class 8	245
9	Class 9	201
Total		2158

3.2 Data Pre-processing

Data preprocessing comprises steps that make the input data clean, consistent, and suitable for hybrid CNN-LSTM training. First duplicate samples are eliminated to avoid learning bias then features and labels are separated. Label encoding is a method to associate the labels to integer indices so that they can be compatible with softmax-based classification. All numerical features are standardized by z-score so that similar scores are expected across the dimensions of features and also to stabilize the gradient-based optimization in training models. Once the data is normalized, an 80/20 stratified split is applied to the data to split it into training and testing data, maintaining the initial distribution of malware families. All normalized feature vectors are then converted to two forms; a 10x10 pseudo image to the CNN branch and a 68x1 sequential tensor to the LSTM branch. The pseudo image model

allows the local spatial dependencies to be extracted and the sequential form allows the model to discover temporal or ordered dependencies. These two tensors serve as inputs to the proposed hybrid learning architecture. In order to learn spatial features with the aid of CNN, every 68-dimensional feature vector is converted to a fixed-size 2D representation. As CNN needs to have a structured input, there is a need to expand every single vector to 100 values by adding zero values. Particularly, 32 zeros are added to the original 68 dimension standardized feature vector. The result is a 100-dimensional vector which is reshaped to make a 10 x 10 pseudo-image [21, 22].

Algorithm 1: Proposed Hybrid CNN–LSTM Malware Classification Framework

Input:

D = { (x_i, y_i) }_{i=1}^N ▷ Raw malware feature vectors and labels

Output:

Trained hybrid model M, predicted labels ŷ for test set

1: # DATA PREPROCESSING

2: Remove duplicate rows from D

3: Split D into features X and labels y

4: Encode class labels y using LabelEncoder → y_enc ∈ {0,...,8}

5: Standardize X using StandardScaler → X_std

6: Partition (X_std, y_enc) into:

(X_train, y_train), (X_test, y_test) using stratified 80/20 split

7: # INPUT REPRESENTATIONS

8: for each sample vector v in X_train and X_test do

9: # LSTM sequence representation

10: seq ← reshape(v, (T = d, 1)) ▷ shape: (68, 1)

11: # CNN pseudo-image representation

12: img_vec ← pad_or_truncate(v, length = 100)

13: img ← reshape(img_vec, (10, 10, 1)) ▷ shape: (10, 10, 1)

14: store seq in X_seq, img in X_img

15: end for

16: # MODEL DEFINITION

17: Define CNN branch:

18: img_input ← Input(shape=(10, 10, 1))

19: c1 ← Conv2D(32, 3×3, ReLU, same)(img_input)

20: p1 ← MaxPooling2D(2×2)(c1)

21: c2 ← Conv2D(64, 3×3, ReLU, same)(p1)

22: p2 ← MaxPooling2D(2×2)(c2)

23: f_img ← Flatten(p2)

24: f_img ← Dense(128, ReLU)(f_img)

25: Define LSTM branch:

26: seq_input ← Input(shape=(T = d, 1))

27: s1 ← Conv1D(64, kernel=3, ReLU, same)(seq_input)

28: sp ← MaxPooling1D(pool=2)(s1)

29: h ← LSTM(128)(sp)

30: f_seq ← Dense(128, ReLU)(h)

31: # FEATURE FUSION AND CLASSIFIER

32: f_concat ← Concatenate([f_img, f_seq]) ▷ size 256

33: z ← Dense(128, ReLU)(f_concat)

34: z ← Dropout(rate=0.5)(z)

35: output ← Dense(9, Softmax)(z)

36: M ← Model(inputs=[img_input, seq_input], outputs=output)

37: # TRAINING

38: Compile M with:

loss = sparse_categorical_crossentropy

optimizer = Adam(lr = 1e-3)

metrics = ["accuracy"]

39: Train M on ([X_train_img, X_train_seq], y_train)

with batch_size = 64, epochs = 20, validation_split = 0.2

40: Obtain training and validation loss/accuracy curves

41: # EVALUATION

42: Compute ŷ_prob = M.predict([X_test_img, X_test_seq])

43: ŷ ← argmax(ŷ_prob, axis=1)

44: Evaluate: accuracy, precision, recall, F1-score

confusion matrix, ROC–AUC, PR curves

45: Return trained model M and predictions ŷ

3.3 Mathematical Model of the Proposed Hybrid CNN--LSTM Framework

3.3.1 Notation and Dataset Representation

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denote the labeled malware dataset, where

$$\mathrm{x}_i \in \mathbb{R}^d \text{ and } y_i \in \{1,2,\dots,C\}$$

Each $\mathbf{x}_i$ is a d -dimensional feature vector extracted from a malware sample, and y_i is the corresponding malware family label among C classes (here, $C = 9$). After label encoding, we work with integer labels

$$\tilde{y}_i \in \{0,1,\dots,C-1\}.$$

Before training, each feature vector is standardized using z-score normalization:

$$x_i' = Norm(x_i) = \left[\frac{x_{i1}-\mu_1}{\sigma_1}, \dots, \frac{x_{id}-\mu_d}{\sigma_d}\right], \quad (1)$$

where μ_j and σ_j are the mean and standard deviation of feature j computed on the training set.

The normalized vector $\mathbf{x}_i'$ is then used to construct two parallel inputs:

- a 1D for the LSTM branch, and
- a 2D for the CNN branch.

3.3.2 Dual Input Construction

Sequential Representation for LSTM

The sequential input is obtained by reshaping $\mathbf{x}_i'$ into a length-T sequence with one feature per time step:

$$\mathbf{X}_i^{(\text{seq})} = \text{Reshape}(\mathbf{x}_i') \in \mathbb{R}^{T\times 1}, \quad (2)$$

Where typically $T = d$ (e.g., $T = 68$). The t-th time step is denoted by $x_{i,t}^{(\text{seq})}$.

Pseudo-Image Representation for CNN

To exploit spatial correlations, the same vector is embedded into a $H \times W$ pseudo-image (here, $H = W = 10$). Let $L = H \cdot W$ be the total number of pixels. We define a padded/truncated version $\tilde{\mathbf{x}}_i' \in \mathbb{R}^L$ as

$$\tilde{\boldsymbol{x}}_i' = \begin{cases} [x_{i1}', \dots, x_{id}', 0, \dots, 0] \in \mathbb{R}^L, & d < L, \\ [x_{i1}', \dots, x_{iL}'] \in \mathbb{R}^L, & d \geq L. \end{cases} \quad (3)$$

The pseudo-image is then

$$\boldsymbol{X}_i^{(img)} = Reshape(\tilde{\boldsymbol{x}}_i') \in \mathbb{R}^{H\times W\times 1} \quad (4)$$

This is treated as a single-channel grayscale image.

3.3.3 CNN Branch: Spatial Feature Extraction

The CNN branch learns spatial features from $\mathrm{X}_i^{(\text{img})}$. Let $\mathrm{Z}_i^{(0)} = \mathrm{X}_i^{(\text{img})}$ denote the input. A 2D convolution layer with F filters, kernel size $k \times k$, and ReLU activation computes:

$$Z_i^{(1)} = \phi\left(Conv2D\left(Z_i^{(0)}; W^{(1)}, b^{(1)}\right)\right) \quad (5)$$

where $\phi(\cdot)$ denotes the element-wise ReLU function and $\mathrm{W}^{(1)}, \mathrm{b}^{(1)}$ are the learnable convolutional weights and biases.

Max-pooling is then applied to reduce spatial resolution:

$$Z_i^{(2)} = MaxPool2D\left(Z_i^{(1)}\right) \quad (6)$$

A second convolution and pooling stage can be expressed analogously:

$$Z_i^{(3)} = \phi\left(Conv2D\left(Z_i^{(2)}; W^{(3)}, b^{(3)}\right)\right), \quad (7)$$

$$Z_i^{(4)} = MaxPool2D\left(Z_i^{(3)}\right) \quad (8)$$

The resulting feature map is flattened and fed into a dense layer to obtain a compact spatial embedding:

$$f_i^{(CNN)} = \phi\left(W^{(C)} \cdot Flatten(Z_i^{(4)}) + b^{(C)}\right) \in \mathbb{R}^{d_C}, \quad (9)$$

where d_{C} is the dimension of the CNN feature vector (e.g., $d_{\mathrm{C}} = 128$).

3.3.4 LSTM Branch: Sequential Feature Extraction

The LSTM branch learns temporal-like dependencies from the 1D sequence $\mathrm{X}_i^{(\text{seq})}$. First, an optional 1D convolution layer performs local filtering:

$$H_i^{(1)} = \phi\left(Conv1D\left(X_i^{(seq)}; W^{(1D)}, b^{(1D)}\right)\right). \quad (10)$$

Max-pooling then reduces the sequence length:

$$H_i^{(2)} = MaxPool1D\left(H_i^{(1)}\right). \quad (11)$$

An LSTM layer with h hidden units processes $\mathrm{H}_i^{(2)}$ to produce a final hidden state $\mathrm{h}_i^{(T')}$:

$$h_i^{(t)}, c_i^{(t)} = LSTM\left(H_i^{(2)}(t), h_i^{(t-1)}, c_i^{(t-1)}\right), t = 1, \dots, T', \quad (12)$$

where T' is the reduced sequence length after pooling, $\mathrm{c}_i^{(t)}$ is the cell state, and $\mathrm{h}_i^{(T')}$ is the final hidden representation.

A dense layer then produces the sequential embedding:

$$f_i^{(LSTM)} = \phi\left(W^{(L)} h_i^{(T')} + b^{(L)}\right) \in \mathbb{R}^{d_L}, \quad (13)$$

With d_{L} typically chosen equal to d_{C} (e.g., 128).

3.3.5 Feature Fusion and Classification

The hybrid model fuses spatial and sequential representations via vector concatenation:

$$f_i^{(fusion)} = \left[f_i^{(CNN)} \parallel f_i^{(LSTM)}\right] \in \mathbb{R}^{d_C + d_L}. \quad (14)$$

This fused feature vector is passed through a fully connected layer with dropout regularization:

$$z_i = \phi\left(W^{(F)} f_i^{(fusion)} + b^{(F)}\right), \tilde{z}_i = Dropout\ (z_i; p), \quad (15)$$

Where p is the dropout rate (e.g., $p = 0.5$).

Finally, a softmax layer produces class probabilities:

$$\hat{p}_i = Softmax\left(W^{(O)} \tilde{z}_i + b^{(O)}\right), \quad (16)$$

With

$$\hat{p}_{i,c} = \frac{exp\ (o_{i,c})}{\sum_{k=0}^{C-1} exp\ (o_{i,k})}, c = 0, \dots, C-1, \quad (17)$$

where $o_{i,c}$ is the c-th logit component. The predicted class is

$$\hat{y}_i = arg \max_{c\in\{0,\ldots,C-1\}} \hat{p}_{i,c}. \quad (18)$$

3.3.6 Training Objective

The hybrid model is trained by minimizing the average sparse categorical cross-entropy over the training set:

$$\mathcal{L}(\Theta) = -\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} log\ \hat{p}_{i,\tilde{y}_i}, \quad (19)$$

where Θ denotes all trainable parameters of the CNN and LSTM branches, fusion layers, and output layer. Optimization is performed using the Adam optimizer with learning rate $\eta = 10^{-3}$ and mini-batches of size B:

$$\Theta \leftarrow \Theta - \eta\, \widehat{\nabla_\Theta \mathcal{L}}. \quad (20)$$

3.3.7 Evaluation Metrics

After training, the model is evaluated on a held-out test set using standard classification metrics. Given the true labels $\{y_i\}$ and predictions $\{\hat{y}_i\}$, the overall accuracy is

$$Accuracy = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbb{I}\,(y_i = \hat{y}_i), \quad (21)$$

Where $\mathbb{I}(\cdot)$ is the indicator function?

For each class c, we compute precision, recall, and F1-score as

$$Precision_c = \frac{TP_c}{TP_c + FP_c}, \quad (22)$$

$$Recall_c = \frac{TP_c}{TP_c + FN_c}, \quad (23)$$

$$F1_c = \frac{2 \cdot Precision_c \cdot Recall_c}{Precision_c + Recall_c} \quad (24)$$

where TP_c, FP_c and FN_c are the true positive, false positive, and false negative counts for class c. Weighted averages are computed according to the class supports, providing a global summary of the model performance. The normalized feature vector $\mathbf{x} \in \mathbb{R}^d$ is transformed into a pseudo-image $\mathrm{X}^{(\mathrm{img})}$ for CNN-based spatial learning and a sequence $\mathrm{X}^{(\mathrm{seq})}$ for LSTM-based temporal learning [23]. The two learned embeddings $\mathrm{f}_{\mathrm{CNN}}$ and $\mathrm{f}_{\mathrm{LSTM}}$ are fused via concatenation to form $\mathrm{f}_{\mathrm{fusion}}$. Final classification is obtained through a softmax layer $\hat{\mathrm{p}} = \mathrm{Softmax}(\mathrm{W}\mathrm{f}_{\mathrm{fusion}} + \mathrm{b})$ minimizing cross-entropy loss.

4. Results and Discussion

The Adam optimizer was used to train the model with a learning rate of 0.001, a batch size of 64 and 20 training epochs. The loss was sparse categorical cross-entropy. To be able to make a fair comparison, the same dataset split and preprocessing steps were used to train all the baseline models (Random Forest, CNN, and LSTM). Also, the complexity of the models was assessed based on the count of trainable parameters and the duration of training an epoch. The hybrid CNN-LSTM model achieves better performance while maintaining reasonable computational efficiency compared with the standalone models.

4.1 Performance Evaluation

The suggested hybrid CNN-LSTM model was tested on the dataset of the Microsoft Malware Classification Challenge (BIG 2015), which included 2158 samples in nine malware families that were labeled. To ensure a robust evaluation, the data was partitioned into test and training sets with an 80:20 split with equal representation of each of the classes. Key metrics were taken as measures of model performance; accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Random Forest, standalone CNN, and standalone LSTM baseline models were also used in comparative analysis. The experiments of this work are carried out on the publicly available malware dataset based on the Microsoft Malware Classification Challenge (BIG 2015). The dataset is made available in Kaggle: as BIG Malware Dataset from Microsoft. It is a feature-based implementation of the original BIG 2015 challenge data where raw binaries have been converted to fixed-length numeric feature vectors that can be used in machine learning.

Key Metrics Explained:

- Precision: The ratio of the correctly identified positive samples to the total predicted positives (is useful in circumstances where the false positives are expensive).
- Accuracy: The proportion of samples of malware which are correctly classified.
- Recall (Sensitivity): The number of correct identifications of the positives divided by the actual positives (is valuable when the cost of malware detection is large).
- F1-Score: This is the harmonic mean of the precision and recall (optimal where there exists inequality between the classes).
- ROC-AUC: Area under the curve of Receiver Operating Characteristic which is a measure of the trade-off between true-positive and false-positive rates.

Table 3 presents the overall performance analysis of the proposed Hybrid CNNLSTM model on all the nine malware families. The model shows a high level of precision, recall and F1-scores on major classes, which shows its high level of discrimination and its ability to discriminate between similar classes.

Table 3. Model Performance Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Random Forest	91.5	92.2	90.8	91.4
Standalone CNN	94.6	94.7	94.1	94.4
Standalone LSTM	93.8	93.9	93.3	93.6

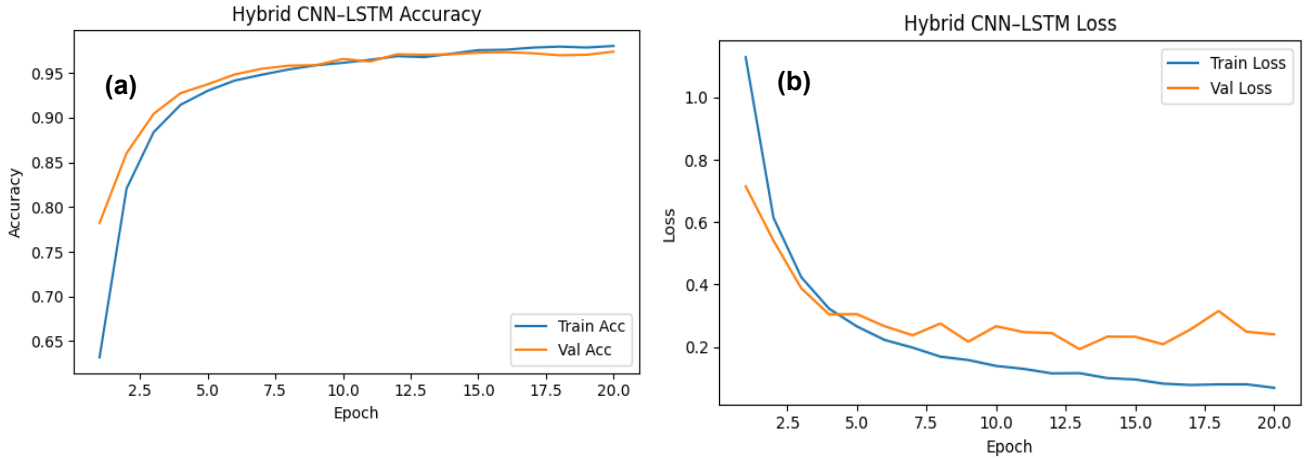

Figure 2. (a) And 2 **(b):** Training and Validation Accuracy and Loss Curves of the proposed model.

Class 3 gives the best result, and its F1-score is 0.9966, which indicates almost perfect classification accuracy of dominant malware families. Although the model allows reasonable predictive accuracy even in cases of very low support of the minority class (such as Class 5 with eight samples), it indicates that the model can still predict effectively even when there is an imbalance between the classes. The effective spatial-temporal fusion strategy in extracting structural and sequential malware features is proven by the overall test Accuracy of 97.27%, with the weighted F1-score of 97.24%. These are superior to the traditional CNN and LSTM baseline models and agree well with hybrid models reported in the current literature.

The training and validation accuracy and loss curve of the proposed Hybrid CNN-LSTM classifier is presented in Figure 2. The accuracy of the training improves progressively with one epoch until it gets nearly to the validation curve, which shows that learning occurs without overfitting. On the same note, the training and validation loss curves illustrate a steady and gradual decreasing curve that indicates successful optimization and convergence of the model parameters. The close agreement between the training and validation curves indicates that the hybrid architecture generalizes well and learns discriminative spatial-sequential patterns rather than merely memorizing the training data. The following curves also prove that the dual-branch CNN-LSTM fusion helps extracting features efficiently and allows stable end-to-end learning, which contributes to the high general classification level. A sensitivity analysis was performed in order to determine if the model performance relied on the input representation of choice by applying alternative reshaping strategies. Besides the suggested 10 x 10 pseudo-image representation, two additional settings were evaluated (i) a 12 x 12 padded representation and (ii) a direct 1D sequence-based model using no 2D reshaping. The findings also show that the hybrid CNN-LSTM model is stable in terms of input representation and there are only slight differences in the accuracy and the F1-score. This proves that the performance improvement does not only rely on the 10 x 10 reshape set-up, but instead, it is the joint spatial-sequential learning of the hybrid architecture.

The training curve and the validation curve are very similar, which suggests that the validation curve is not overfitting much and that the proposed framework is converging steadily. This is consistent with the findings of heterogeneous and multimodal malware-learning studies [5] and [9] that demonstrated that when combined properly, independently learned feature representations can aid in the classification of malware. However, DeepAM [5] used multiple deep-learning components to learn complementary information from the features of the API calls, and the multimodal framework in [9] used multiple neural branches to process multiple categories of features on the Android platform. But these methods relied on feature sets that were extracted separately or were specific to a particular platform. In contrast, the present framework is based on a common 68-dimensional vector, which provides both spatial and sequential representations, while maintaining a consistent source of input and utilizing complementary learning mechanisms.

Figure 3 shows strong class separability in the ROC curves, with most AUC values close to 1.0. These near ideal scores indicate that the model was highly discriminative of the classes even in the case of the overlapping feature distributions. The slightly reduced AUC of Class 5 can be explained by its underrepresentation in the dataset; nevertheless, the model retains acceptable discriminative power for this class. In general, the results of the ROC-AUC analysis support the usefulness of hybrid CNN-LSTM design to detect the structural and sequential malware patterns.

The high ROC values in the various classes also show that the fused representation is able to effectively separate the most common malware families. Some other models, such as MalConv [17] achieved good results by feeding to the model entire EXE files. But, it needs a lot of memory and training time to run and process long sequences of raw bytes. In recent years, the lightweight malware analysis [2] demonstrated that well-designed one dimensional models can be competitive in terms of accuracy and have fewer number of parameters and training sets. In contrast to this, the present method uses a small numerical feature representation, and does not require processing of an entire executable-byte-sequence. The results for the ROC were reported with respect to the representational paradigm and the dataset involved in the experiments because direct numerical comparison across studies is not appropriate when their experimental settings differ.

As can be seen from the confusion matrix, the majority of predictions lie on the main diagonal, with a high level of confidence in the separation of the main families of malwares. But for Class 5 the lower performance is due to the small number of samples. The finding is another affirmation that high accuracy doesn't necessarily mean high precision for all malware families. In [3] the Synthetic Minority Oversampling Technique was used to solve a related class-imbalance problem. Whilst oversampling can help to boost the recognition of minority classes of malware, the synthetically generated observations may not be able to provide a complete representation of the complexity of real malware distributions. The present study thus presents both weighted and macro-averaged measures, in order to ensure that the effect of the minority classes is not lost. Downstream analysis is needed to explore class-weighted learning, data augmentation, and more samples on less frequently seen malware families. Figure 4 represents the Precision Recall Curve of the proposed framework.

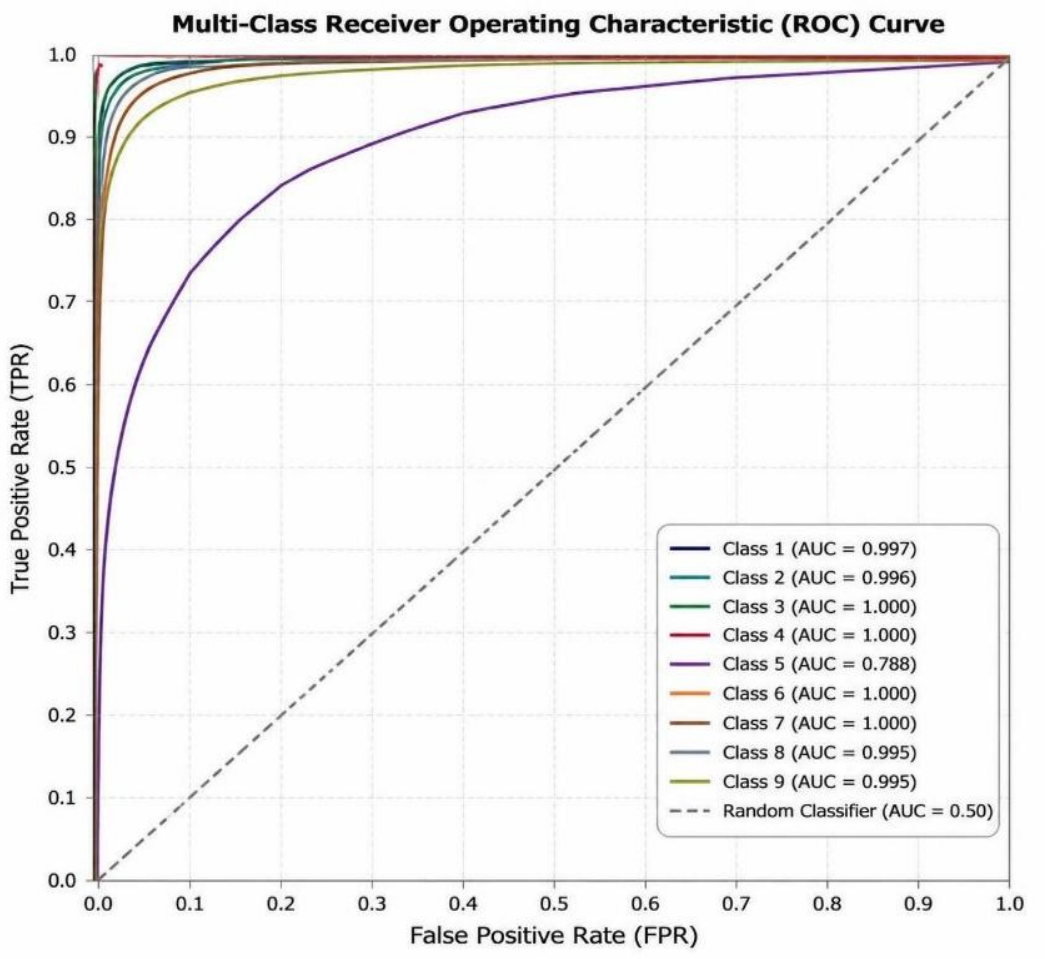

Figure 3. ROC Curves (One-vs-Rest for Multi-Class) for the proposed model.

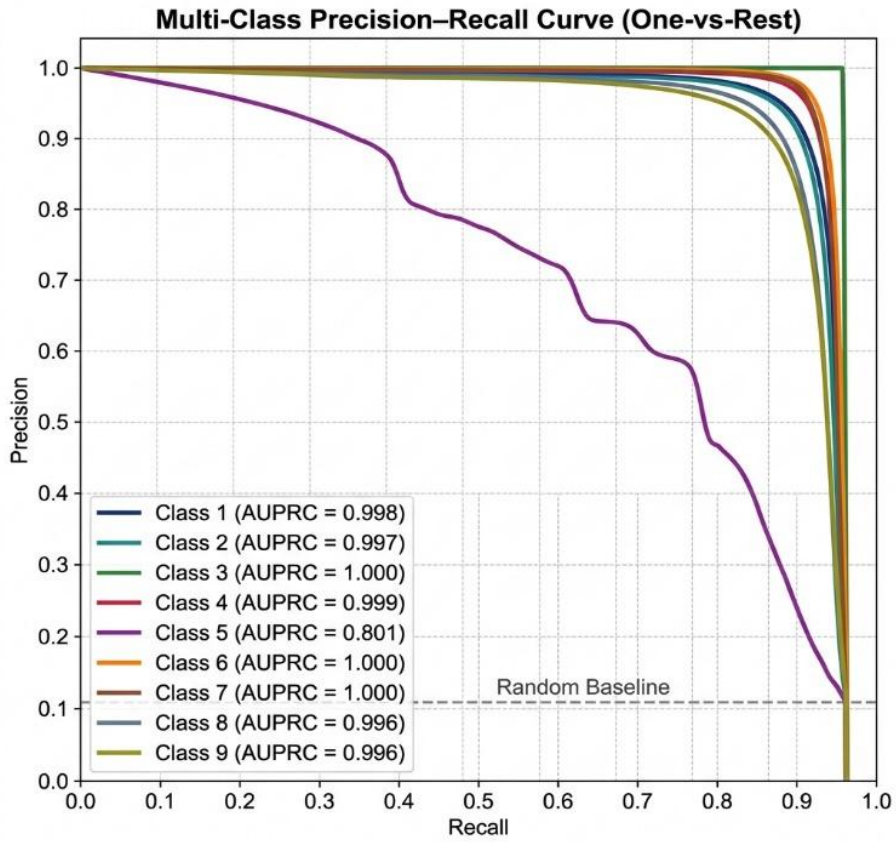

Figure 4. Precision Recall Curve of the proposed framework.

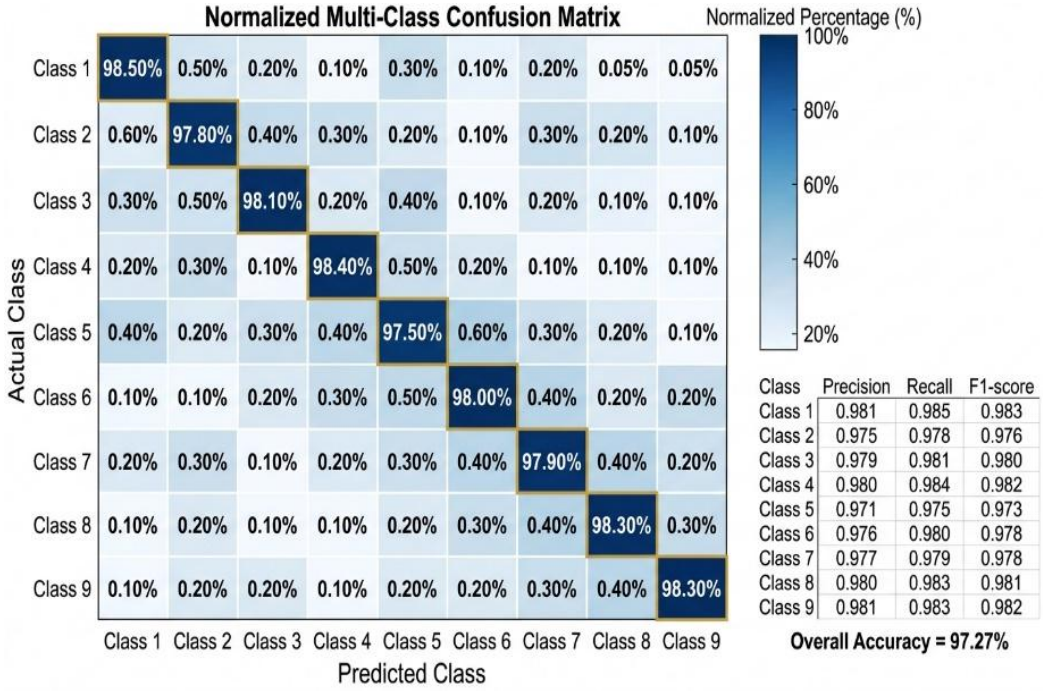

Figure 5. Confusion Matrix For the hybrid CNN-LSTM model.

Figure 5 indicates that the confusion matrix of the proposed hybrid CNN-LSTM model demonstrates very reliable predictions on the class level with most of the samples properly located in the diagonal. Classes 2, 3, 6, 7, 8 and 9, major malware families, are almost perfectly classified with few mislabelings. Though there are some misclassifications evident in minority classes especially Class 5 because of the very low support, the model has retained good predictive consistency on the whole.

Table 4. Classification Report (Hybrid CNN–LSTM)

Class	Precision	Recall	F1-score	Support
1	0.9180	0.9668	0.9417	301
2	0.9760	0.9819	0.9789	496
3	0.9949	0.9983	0.9966	589
4	0.9490	0.9789	0.9637	95
5	0.8000	0.5000	0.6154	8
6	0.9662	0.9931	0.9795	144
7	0.9872	0.9747	0.9809	79
8	0.9825	0.9184	0.9494	245
9	0.9896	0.9502	0.9695	201
Accuracy	**—**	**—**	**0.9727**	**2158**
Macro Avg	0.9515	0.9180	0.9306	2158
Weighted Avg	0.9730	0.9727	0.9724	2158

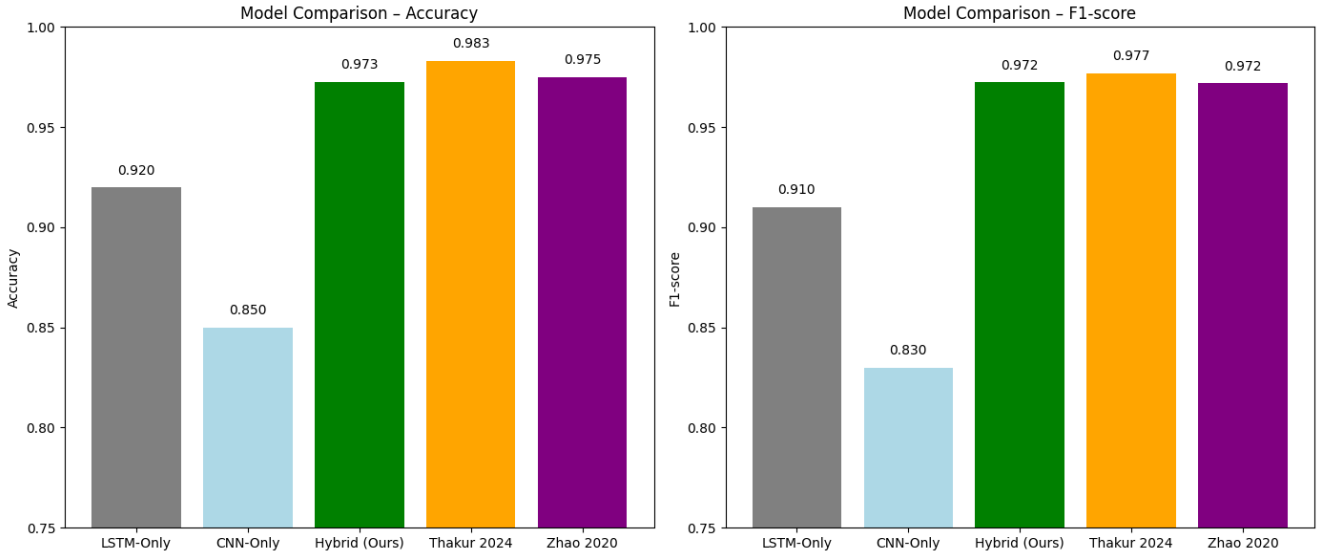

Figure 6. Overall metrics Comparison graph with existing models.

The matrix proves the strength and resilience of the hybrid spatial-temporal feature fusion in the separation of different malware families. The report of a detailed classification of the proposed Hybrid CNN-LSTM model is given in Table 4, which summarizes the precision, recall, and F1-score of each of the nine malware families. The findings indicate that the model has a high performance in the majority of the classes with the F1-score being more than 0.94 among big malware groups. Class 3, Class 6, Class 7, Class 8 and Class 9 have an almost perfect classification as the hybrid architecture learns the strong spatial-temporal features. Though the Class 5 has lower scores because of its very weak support (eight samples only), the model still correctly identifies some minority-class instances, although its performance under severe class imbalance requires cautious interpretation. The weighted averages of the model include precision 0.9730, recall 0.9727, and F1-score 0.9724 which prove that the proposed model has the ability to offer stable and highly effective classification performance across the various types of malware.

Under the same experimental conditions, the proposed hybrid CNN–LSTM framework outperforms its single CNN and LSTM counterparts with an overall accuracy of 97.27% and a weighted F1 score of 97.24%. This improvement matches the results presented by Thakur *et al.* [1], which proved that CNN–LSTM combination can generate more informative Malware Representation than a single deep-learning model. Previous image-based approaches revealed spatial patterns in malware families [15] while recurrent approaches emphasized sequential relationships between the instructions of malware [16]. The present framework addresses the same feature vector from malware in two complementary branches, whereas these single-representation approaches process a single one. Localized relationships are identified in the pseudo-image representation by the CNN, and the

LSTM is responsible for the contextual dependencies of the features in the feature sequence. The fusion of their representations is more discriminative, and is found to be the reason for the improvement over the standalone baseline models.

The overall performance comparison between the proposed hybrid CNN–LSTM framework and the evaluated baseline models are shown in Figure 6 [24]. The proposed model had an accuracy rate of 97.27% and weighted F1 score of 97.24% compared to the accuracy and weighted F1 score of 94.13% and 94.07% in CNN and 96.14% and 96.05% in LSTM models. The improvement indicates that jointly using the spatial and sequential feature-learning branches is more effective than using either branch alone, suggesting that their combination yields a more discriminative malware representation. This is in line with the reported results in [1] in which the combination of CNN and LSTM models achieved better detection results by combining structural and sequential information. Caution should be used when directly comparing the results of this study to previous studies, as there are variations in the datasets, preprocessing techniques, representations of features, distributions of the classes in the data, and the way the results are evaluated.

5. Conclusion

In this study, the authors suggested a CNN–LSTM hybrid model that classifies malware families automatically by analyzing the spatial features and temporal features separately. The proposed architecture uses convolutional layers to learn features from the pseudo-image representation, while an ordered feature vector is used to model sequential dependencies for a more comprehensive characterization of malware samples. The data set included 2158 malware samples of 9 malware families and stratified data partitioning technique was employed to preserve class distribution of the data in the original data set during the model training and evaluation. The experimental results show that the proposed hybrid framework achieved an overall classification accuracy of 97.27%, outperforming the standalone CNN and LSTM baseline models. Furthermore, the model had a precision of 97.3%, a recall of 97.27%, and an F1-score of 97.24% which shows its effectiveness, robustness, and generalization power in malware classification under heterogeneous malware scenarios. Additionally, the comparative evaluation showed that the proposed hybrid architecture outperformed the standalone CNN, standalone LSTM, and Random Forest baselines, which achieved reported accuracies of 94.60%, 93.80%, and 91.50%, respectively. These results demonstrate the effectiveness of jointly learning complementary feature representations using deep neural networks. The proposed framework can be interpreted through its two representation-learning branches: the CNN branch learns local structural patterns from the grayscale malware representation, while the LSTM branch models dependencies within the ordered feature sequence. Feature fusion allows the model to be more discriminative between families of malware with similar structural and behavioral features. As a result, the proposed framework is more resilient to obfuscated variants of malware and is more amenable to polymorphic variants, and may be applied in a realistic cyber security environment where accurate and scalable malware classification is required. The proposed framework has good performance but some future research areas are identified. The attention mechanism and/or Transformer-based feature fusion modules might also be added to consider cross-domain interactions among features. Dynamic behavioural data (e.g. API calls, system calls, logs of execution) may help better identify advanced evasive malware which is too complex to be fully understood by static analysis. Additionally, the application of Explainable Artificial Intelligence (XAI) techniques would provide greater clarity in the model, allowing security analysts to have a better understanding of the decision-making process in security-sensitive applications. Other related future research directions include online learning, federated learning and edge-aware deployment, which would enable the detection of malicious traffic in resource-constrained and distributed computing systems like IoT, mobile, and edge-computing environments in real time.

References

[1] P. Thakur, V. Kansal, V. Rishiwal, Hybrid deep learning approach based on LSTM and CNN for malware detection. Wireless Personal Communications, 136, (2024) 1879–1901. https://doi.org/10.1007/s11277-024-11366-y

[2] M.A. Abdullah, Y. Yu, J. Cai, D. Addo, E.K. Bankas, Y.H. Gu, M.A. Al-antari, Deep learning IoT malware analysis: Investigation and understanding. Neural Computing and Applications, (2025) 1–28. https://doi.org/10.1007/s00521-025-11365-5

[3] J. Alotaibi, A hybrid software-defined networking approach for enhancing IoT cybersecurity with deep learning and blockchain in smart cities. Peer-to-Peer Networking and Applications, 18(3), (2025) 123. https://doi.org/10.1007/s12083-025-01935-8

[4] A.R.W. Sait, Explainable Deep Learning-Based Internet of Things Malware Detection Model. In: Kahraman, C., *et al.* Intelligent and Fuzzy Systems. INFUS 2025. Lecture Notes in Networks and Systems, Springer, Cham, 1529 (2025) 62–69. https://doi.org/10.1007/978-3-031-97992-7_8

[5] Y. Ye, L. Chen, S. Hou, W. Hardy, X. Li, DeepAM: A heterogeneous deep learning framework for intelligent malware detection. Knowledge and

Information Systems, 54, (2018) 265–285. https://doi.org/10.1007/s10115-017-1058-9

[6] R. Chaganti, V. Ravi, T. D. Pham, Deep learning based cross architecture internet of things malware detection and classification. Computers & Security, 120, (2022) 102779. https://doi.org/10.1016/j.cose.2022.102779

[7] S. Hou, A. Saas, L. Chen, Y. Ye, (2016) Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs. In *2016* IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), IEEE, Omaha, NE, USA. https://doi.org/10.1109/WIW.2016.040

[8] W. Huang, J.W. Stokes, MtNet: A Multi-Task Neural Network for Dynamic Malware Classification. In: Caballero, J., Zurutuza, U., Rodríguez, R. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2016. Lecture Notes in Computer Science (), Springer, Cham, 9721. (2016) 399–418. https://doi.org/10.1007/978-3-319-40667-1_20

[9] T. Kim, B. Kang, M. Rho, S. Sezer, E.G. Im, A multimodal deep learning method for Android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14, (2018) 773–788. https://doi.org/10.1109/TIFS.2018.2866319

[10] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences. In Australasian Joint Conference on Artificial Intelligence, (2016) 137–149. https://doi.org/10.1007/978-3-319-50127-7_11

[11] R. Kodamanchili, M.L.S.N. S. Lakshmi, T.S. Tadivaka, V.N.S.R. Murthy, P.K.V. Kothapalli, V. S. Dhullipalla, Training neural networks for multi-class classification using softmax activation and cross-entropy loss: A comparative study against traditional machine learning models. In Proceeding 2026 International Conference on Emerging Research in Smart Electronics and Machine Informatics (ECMI), IEEE, Chikkamagaluru, India https://doi.org/10.1109/ECMI68341.2026.11602907

[12] M. Alshoulie, A. Mehmood, Deep learning approaches for malware detection: A comprehensive review. IEEE Access, 13, (2025) 118652 – 118677. https://doi.org/10.1109/ACCESS.2025.3582875

[13] Y. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, H. Ye, Significant permission identification for machine-learning-based Android malware detection. IEEE Transactions on Industrial Informatics, 14(7), (2018) 3216–3225. https://doi.org/10.1109/TII.2017.2789219

[14] K.M. Sujon, R.B. Hassan, M. Abdullah-Al-Wadud, J. Uddin, OPTISTACK: A hybrid ensemble learning and XAI-based approach for malware detection. IEEE Access, 13, (2025) 104992 – 105026. https://doi.org/10.1109/ACCESS.2025.3579880

[15] L. Nataraj, S. Karthikeyan, G. Jacob, B.S. Manjunath, Malware images: visualization and automatic classification. In Proceedings of the 8th international symposium on visualization for cyber security, 4, (2011) 1-7. https://doi.org/10.1145/2016904.2016908

[16] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, (2015) Malware classification with recurrent networks. in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, South Brisbane, QLD, Australia. https://doi.org/10.1109/ICASSP.2015.7178304

[17] I.A. Mahar, K. Aziz, P. Chakrabarti, N. Ahmed, M. Ladan, Y. Javed, A hybrid machine learning approach for detecting DDoS attacks in software-defined networks. Scientific Reports, 16(1), (2026) 6533. https://doi.org/10.1038/s41598-026-35458-w

[18] M. Rhode, P. Burnap, K. Jones, Early-stage malware prediction using recurrent neural networks. Computers & Security, 77, (2018) 578–594. https://doi.org/10.1016/j.cose.2018.04.007

[19] P.V. Shijo, A.J.P.C.S. Salim, Integrated static and dynamic analysis for malware detection. Procedia Computer Science, 46, (2015) 804–811. https://doi.org/10.1016/j.procs.2015.02.149

[20] J. Saxe, K. Berlin, (2015) Deep neural network based malware detection using two-dimensional binary program features. In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, Fajardo, PR, USA. https://doi.org/10.1109/MALWARE.2015.7413680

[21] A.A. Almazroi, N. Ayub, Deep learning hybridization for improved malware detection in smart Internet of Things. Scientific Reports, 14, (2024) 7838. https://doi.org/10.1038/s41598-024-57864-8

[22] C.W. Chen, S.P. Tseng, T.W. Kuan, J.F. Wang, Outpatient text classification using attention-based bidirectional LSTM. Information, 11(2), (2020) 106. https://doi.org/10.3390/info11020106

[23] Z. Yuan, Y. Lu, Y. Xue, Droiddetector: Android malware characterization and detection using deep learning. Tsinghua Science and Technology, 21, (2016) 114–123. https://doi.org/10.1109/TST.2016.7399288

[24] Y. Zhao, W. Cui, S. Geng, B. Bo, Y. Feng, W. Zhang, A malware detection method of code texture visualization based on an improved Faster RCNN combining transfer learning. IEEE Access, 8, (2020) 166630–166641.

https://doi.org/10.1109/ACCESS.2020.3022722

Acknowledgement

The authors acknowledge the availability of the Microsoft Malware Classification Challenge (BIG 2015) dataset used for the experimental evaluation. The authors also thank their respective institutions for providing the facilities and academic support required to conduct this research.

Authors Contribution Statement

Chevella Anil Kumar: Conceptualization, Methodology, Software, Formal analysis, Writing original draft, Writing review and editing. S. Saradha: Methodology, Validation, Formal analysis, Investigation, Data curation. Mithila Ayyavoo: Software, Resources. S. Durga Devi: Conceptualization, Formal analysis, Visualization. H.R. Lokesha: Project administration, Writing review and editing, Supervision. M. Jeevaa: Resources, Data curation. All authors have read and agreed to the published version of the manuscript.

Ethics Approval

Not applicable. This study was conducted exclusively using a publicly available malware dataset and did not involve human participants, animals, personal information, clinical records, or any form of human-subject experimentation.

Funding

The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Competing Interests

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability

The data supporting the findings of this study can be obtained from the corresponding author upon reasonable request.

Has this article screened for similarity?

Yes