j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection

Nadia Mahmood Ali[1], Munera A. Jabaar[1], Ahmed Majid Taha[2&3]

[1] *Institute of Medical Technology Al-Mansur, Middle Technical University, Baghdad, Iraq.*

[2] *College of Biomedical Informatics, University of Information Technology and Communications, Baghdad, Iraq.*

[3] *Soft Computing and Data Mining Center, Universiti Tun Hussein Onn Malaysia 86400 Parit Raja, Batu Pahat Johor, Malaysia*

dr.ahmed_majid@uoitc.edu.iq

Abstract The persistent evolution of Windows malware presents a significant challenge to static-analysis techniques, which often rely on handcrafted features and single-modality models that struggle to generalize across diverse and obfuscated samples. This study proposes a novel Dual-Branch CNN-LSTM Residual Network that concurrently processes a uniform static feature vector as both a pseudo-image and a sequential input, thereby capturing complementary spatial and temporal patterns without necessitating multiple preprocessing pipelines. The architecture incorporates residual connections in each branch to preserve gradient flow and facilitate deep learning. Experiments conducted on the EMBER dataset demonstrate that the proposed method attains an accuracy of 97.1 %, alongside a precision of 96.9 %, a recall of 97.1 %, and an F1-score of 97.0 %, surpassing existing single-branch and traditional baseline models. These results underscore the capacity of dual-branch residual fusion to improve detection performance while maintaining computational efficiency and robustness to feature obfuscation. The unified preprocessing scheme further simplifies cross-dataset evaluation, paving the way for scalable deployment in real-world Windows environments

10.36371/port.2025.4.6

Keywords: *Windows Malware Detection, Convolutional Neural Network, Long Short-Term Memory, Residual Network, Dual-Branch Architecture, Deep Learning.*.

1. INTRODUCTION

Modern Malware detection remains a critical component of cybersecurity for modern computing environments, particularly those powered by Windows operating systems [1]. As malicious software grows both in volume and sophistication, defenders must continually advance detection techniques to keep pace [2]. Traditional signature-based methods, once the cornerstone of antivirus solutions, struggle to identify novel or obfuscated threats, leaving enterprise networks and personal devices vulnerable to data breaches, ransomware attacks, and system compromise [3].

The central problem addressed in this work is the reliable and efficient identification of Windows malware using only static, readily extractable features [4]. Static analysis avoids the overhead and risk of executing potentially dangerous binaries, yet it often yields limited information compared to dynamic monitoring approaches [5]. Current machine-learning solutions typically focus on either purely spatial representations of feature vectors (e.g., treating features as images) or on temporal/sequential models (treating API calls or raw byte sequences as time series) [6].

Each paradigm offers strengths convolutional filters excel at detecting local patterns, while recurrent networks capture order dependencies but neither alone fully exploits the complementary nature of malware characteristics [7]. Moreover, heterogeneity among publicly available Windows malware corpora complicates cross-dataset generalization: differing feature schemas and distributions force bespoke preprocessing pipelines, hindering unified evaluation [8]. Any program, script, or code section created to disrupt, damage, or steal access from computers is called malicious software ("malware"). In Windows platforms, malware can comprise trojans, worms, ransomware, rootkits, and attacks that occur in a computer's memory without using any files [9]. Results of a successful infection may be a slight decrease in system performance, large data theft, secrets sale, holding critical files hostage via encryption, or continuous backdoor access. Experts say that the estimated cost caused by ransomware each year is expected to reach well beyond $20 billion, showing why it is important to detect malware [10].

Even with much research in static and dynamic analysis, there are still important challenges. At first, attackers find ways to make their techniques harder to detect by modifying packing, encryption, and obfuscation; this means that the tools we use today might become irrelevant tomorrow [11]. Second, since data is not the same in public datasets, this leads to different feature representations, making it harder for models to be applied and compared. Third, in situations where each new test sample must be run within a short time frame, deep dynamic

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

analysis cannot be used as it would cost too much and risk unsafe results [13]. To sum up, since there are many examples of good software and just a few instances of malware, it is challenging for classifiers to reduce false positives that could harm the operation of software applications [14].

This work is designed to solve these issues by suggesting a unified deep network that uses just one static feature vector for both types of data sequentially, and then fuses them to obtain a final classification. Transforming a fixed-length vector into an image for one branch and retaining it as a sequence for another, the network gets local features and temporal dependencies without asking for different inputs. With a single preprocessing pipeline, it is much easier to look at results from different datasets and see how well models work generally.

The main contributions offered in this work are as follows: a Dual-Branch Residual Network using CNN-LSTM that takes the same input and processes it both spatially and sequentially, and benefits from the strengths of both kinds of neural networks. Make sure each feature vector has the same length, and then add common image and sequence transformations, as these solve the problem of schema heterogeneity. Evaluation Across Different Datasets: Tests on EMBER, EMBERSim, and SoReL-20M show that more gains are made, compared to single-branch and non-residual baselines. Verifies that stat-analysis speeds are suitable for the needed detection, which means using it won't require dynamic execution and is beneficial for big Windows networks. Focusing on both the past and present in a single framework, the study moves forward with static malware detection on Windows, accounting for both cyber-attacker's new tricks and the real-world restraints of those who must defend such systems.

The rest of this paper is organized as follows: Section 2 presents related works. Section 3 explains the details and architecture of the proposed model. Section 4 discusses the results obtained. Finally, Section 5 concludes.

2. RELATED WORK

Malware detection capabilities have undergone a remarkable transformation driven by breakthrough advances in deep learning algorithms and computational technologies. Azeez et al. (2021) [15] introduced a stacked ensemble of seven one-dimensional convolutional networks feeding an ExtraTrees meta-classifier, achieving very high accuracy at the expense of considerable computational overhead and ensemble complexity.

Aziz et al. (2022) [16] applied decision trees, random forests, XGBoost, and AdaBoost to handcrafted PE features, demonstrating fast inference but relying heavily on manual feature engineering and showing limited resilience to novel obfuscations. Algorain and Clark (2022) [17] leveraged Bayesian hyperparameter optimization to systematically improve diverse classifiers, though their gains remained tied to the base learner and did not explore combining multiple feature modalities.

Ayoub et al. (2023) [18] assembled a large PE dataset enriched with DLL imports, API call counts, and header metrics, obtaining high detection rates through ensemble methods but incurring extensive manual feature extraction. Divakarla et al. (2023) [19] trained a deep neural network on EMBER static features, yielding strong benchmark performance but lacking residual connections to support deeper architectures and stable gradient flow. Komarudin et al. (2023) [20] conducted a broad analysis of artificial-intelligence applications in malware detection, resulting in high accuracy, but the approach is sensitive to variations in process behaviors and requires extensive training data.

Hammi et al. (2024) [21] and Ilić et al. (2024) [22] shifted toward dynamic analysis using API call sequences and full sandbox reports, respectively, to capture runtime behavior; both improve detection rates but incur high execution overhead, security risks, and slow throughput. Mishchenko and Dorosh (2024) [23] treated imported DLL lists as text, applying Word2Vec encoding and classifying with Random Forest, SVM, and MLP; this novel text-based view nonetheless ignores many static attributes and local spatial patterns. Baghirov et al. (2024) [24] paired LightGBM with PCA and SHAP for interpretable static-feature models, achieving strong performance but potentially discarding nonlinear feature interactions. Syeda and Asghar (2024) [25] advanced dynamic malware classification by categorizing API behaviors in Windows PE files, improving behavior modeling but still depending on costly runtime instrumentation.

Vuran Sarı (2025) [26] constructed API–DLL reference graphs, embedding them with Node2Vec and a Graph Attention Network, followed by a CNN–GRU classifier, effectively capturing relational nuances at the cost of expensive, dataset-specific graph construction. Finally, Miraoui and Belgacem (2025) [27] benchmarked classic and deep models, including CNN, LSTM, and CNN–LSTM on multiclass Windows malware tasks, confirming CNN–LSTM superiority but omitting residual design and a unified preprocessing scheme across heterogeneous datasets.

In this paper, our dual-branch CNN-LSTM residual network unifies preprocessing into a single static-feature pipeline, embeds residual connections for stable deep training, fuses spatial and sequential representations to capture both local and temporal patterns, and eliminates the overhead and security risks of dynamic analysis. Table 1 provides an overview of related work with the strengths and weaknesses of each study.

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

Table 1: *An overview of related works.*

Ref.	Approach	Strength Points	Weak Points or Restrictions
[15]	Stacked 1D CNNs + ExtraTrees	Very high accuracy	High computational cost; complex ensemble
[16]	Decision Trees, RF, XGBoost, AdaBoost	Fast inference; easy deployment	Manual feature engineering; limited obfuscation robustness
[17]	Bayesian hyperparameter optimization	Systematic tuning; consistent accuracy gains	Dependent on the base learner; no multimodal fusion
[18]	Rich static features + ML ensembles	High accuracy via diverse engineered features	Extensive manual feature extraction
[19]	Deep NN on EMBER static features	Strong benchmark performance	No residual connections; potential gradient-flow instability
[20]	AI impact analysis in malware detection	High accuracy achieved	Requires extensive training data
[21]	Dynamic API call voting ensemble	Leverages runtime behavior	High runtime overhead; security risks
[22]	Full sandbox logs + Random Forest	Very rich behavioral features	Slow, resource-intensive sandbox execution
[23]	Word2Vec on DLL imports + RF/SVM/MLP	Novel text-based representation	Omits many static features; lacks spatial pattern modeling
[24]	LightGBM + PCA + SHAP	High interpretability; strong after dimensionality reduction	PCA may remove nonlinear feature interactions
[25]	Dynamic API behavior categorization	Improved behavior modeling	Depends on costly runtime instrumentation
[26]	Graph (Node2Vec + GAT) + CNN–GRU	Captures complex relational patterns	Computationally expensive; dataset-specific
[27]	CNN, LSTM, CNN–LSTM comparison	Validated CNN–LSTM for multiclass detection	No residual design; no unified preprocessing across datasets

3. PROPOSED METHOD

The proposed method offers a hybrid approach for malware detection on Windows. The core idea is to fuse spatial and temporal representations extracted from a single, fixed feature vector derived from Windows binaries. By splitting the same feature vector into an "image" representation for a convolutional branch and a "sequence" representation for an LSTM branch, we preserve the identical raw information while enabling two complementary views. The CNN branch captures local structural patterns, whereas the LSTM branch models implicit sequential dependencies. Residual connections are used in both branches to facilitate gradient flow. After independent encoding, feature maps are concatenated and passed through a classifier. Figure 1 illustrates the general framework of the proposed method.

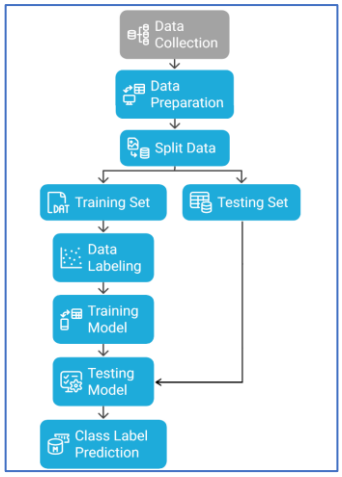

Figure 1: *The general framework of the proposed method.*

The proposed method consists of four main stages: data collection, data preparation, proposed model architecture, and model training process.

3.1 Dataset Collection

Three public datasets are used to evaluate robustness across diverse samples: EMBER [28] provides static feature vectors from Windows PE files (header metadata, imported functions, etc.) with over 900,000 labeled training samples in a standardized Kaggle-hosted format. EMBERSim [29], a synthetic extension of EMBER from CrowdStrike, introduces realistic perturbations (e.g., simulated packers/obfuscation) to test generalization while maintaining the original feature schema. The SoReL-20M [30] subset focuses on real-world Windows executables, curating 100,000 balanced samples (50,000 malware/benign) with raw bytes, PE headers, and API calls, filtered for feature compatibility with EMBER. Each dataset undergoes independent evaluation using the same pipeline, with models trained and tested on holdout splits from the same dataset to measure dataset-specific performance.

3.2 Dataset Preparation

For each dataset, the following standardized pipeline is applied: feature vectors are extracted, with EMBER and EMBERSim already providing numeric vectors (about 2,000 dimensions) that are normalized to zero mean and unit variance using training set statistics, while SoReL-20M raw PE files are parsed with a custom pipeline replicating EMBER's feature engineering to yield comparable vectors, followed by the same normalization. All vectors are then padded or truncated to a fixed length of 2048 elements to ensure uniform input size,

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

facilitating batch processing and enabling conversion to both image and sequence forms. For image representation, the normalized 2048-length vector is reshaped into a 32×64 grayscale "image" preserving feature adjacency, while for sequence representation, the vector is treated as a 1D sequence of length 2048 with no further embedding. Binary labels are encoded as 0 (benign) or 1 (malware), and each dataset is split into 80% training, 10% validation, and 10% testing sets using stratified sampling to maintain class balance. This process produces three independent sets of (image, sequence, label) triples, ready for CNN-LSTM model training and evaluation.

3.3 The Proposed Model

The proposed model employs a dual-branch residual architecture that processes a single, fixed-length feature vector in two complementary modalities, spatial and sequential, before merging them to produce a robust malware-versus-benign decision. We begin by normalizing and padding each raw feature vector, which represents static characteristics of Windows binaries, to a consistent length of 2048 elements. This uniform vector is then split without overlap into a pseudo-image of dimensions 32×64 for the convolutional branch and, in parallel, treated as a one-dimensional sequence for the LSTM branch. By feeding identical information into both branches, we eliminate the need for disparate preprocessing pipelines across datasets, simplifying cross-dataset evaluation. Figure 2 shows the structure of the proposed model. Table 2 shows a summary of the network architecture parameters.

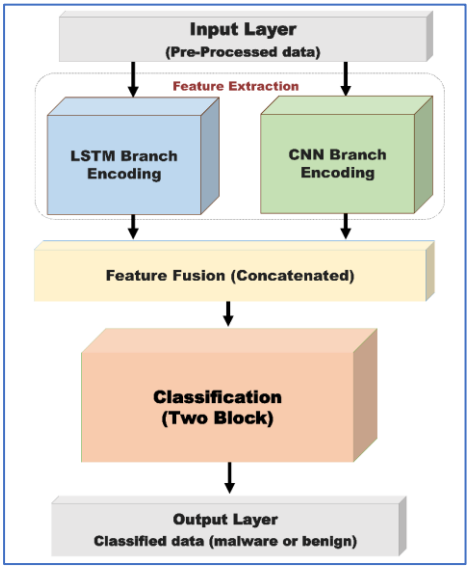

Figure 2: *The structure of the proposed model.*

Table 2: *Summary of network architecture parameters*

Component	Layers / Units / Filters
CNN Initial Conv	3×3 Conv, 64 filters
Residual Block 1 (CNN)	3×3 Conv ×2 (64 filters), Identity skip connection
Residual Block 2 (CNN)	3×3 Conv ×2 (128 filters), skip 64→128 (1×1 conv)
Residual Block 3 (CNN)	3×3 Conv ×2 (256 filters), skip 128→256
Residual Block 4 (CNN)	3×3 Conv ×2 (512 filters), skip 256→512
CNN GlobalAvgPool	Output 512-D
LSTM FC Projection	Input 2048→512
Bi-LSTM Layer 1	256 units per direction
Bi-LSTM Layer 2	256 units per direction
Residual Multilayer Perceptron Layer 1	1024→512, BatchNorm, ReLU
Residual Multilayer Perceptron Layer 2	512→256, BatchNorm, ReLU
Classification Output	256→1 (Sigmoid)

In the convolutional branch, the 32×64 input tensor passes through an initial 3×3 convolutional layer with 64 filters, followed by batch normalization and ReLU activation. Four successive residual blocks then extract increasingly abstract spatial features: each block consists of two 3×3 convolutions whose filter count doubles (64→128→256→512) at every stage, with a 1×1 convolutional skip connection employed whenever the dimensionality changes. A global average pooling layer reduces the final feature maps to a 512-dimensional vector.

Concurrently, the LSTM branch begins with a fully connected projection that maps the 2048-dimensional input into a 512-dimensional embedding and applies ReLU activation and dropout (rate 0.3). Two stacked bidirectional LSTM layers, each with 256 hidden units per direction and interleaved residual connections, capture latent temporal dependencies within the feature sequence. Specifically, after each bidirectional LSTM, the layer's output is added to a linear transformation of its input to preserve gradient flow and mitigate vanishing issues inherent in deep recurrent networks; a subsequent ReLU activation completes the residual block.

The final hidden states from the second bidirectional LSTM are concatenated to yield a 512-dimensional sequential feature vector. These two 512-dimensional vectors ،one spatial and one

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

sequential ‹are concatenated into a single 1024-dimensional embedding that undergoes classification through a residual multilayer perceptron.

First, a 1024→512 fully connected layer with batch normalization, ReLU, and dropout (rate 0.4) is configured with an identity-based shortcut to form a residual block; its output passes through a second 512→256 layer (again with batch normalization, ReLU, and dropout). Finally, a linear layer maps the 256-dimensional representation to a single logit, which is squashed by a sigmoid function to produce a malware probability.

Training minimizes binary cross-entropy, and residual connections in both branches ensure stable gradient propagation, enabling the model to learn nuanced patterns from static feature vectors without reliance on dynamic execution traces. By unifying spatial and temporal representations extracted from the same data, this architecture strikes a balance between representational richness and cross-dataset generalizability, addressing shortcomings of single-branch. Algorithm 1 summarizes the general steps for the proposed network architecture.

Algorithm 1: General Steps for the Network Architecture

Input: Pre-processed data

Output: Classified data (malware or benign)

Stage 1: Split the single 2048-dimensional feature vector into two modalities:

- Reshape into a 32×64 “image” tensor for the CNN branch
- Treat as a 1 × 2048 sequence for the LSTM branch

Stage 2: CNN Branch Encoding

• Apply a 3×3 convolution (64 filters), followed by batch normalization and ReLU activation

• Pass through four residual blocks, each containing two 3×3 convolutions (filters: 64→128→256→512), batch normalization, ReLU, and a 1×1 skip connection whenever dimensions change.

• Perform global average pooling to produce a 512-dimensional spatial feature vector.

Stage 3: LSTM Branch Encoding

• Project the 2048-dimensional sequence into 512 units via fully connected → ReLU → dropout (0.3).

• Pass through two stacked bidirectional LSTM layers (256 units per direction), each followed by dropout (0.3) and a residual connection that adds a linear transform of its input before ReLU.

• Concatenate the final hidden states to form a 512-dimensional sequential feature vector.

Stage 4: Feature Fusion Concatenate feature vector CNN and feature vector LSTM into a single 1024-dimensional embedding feature vector.

Stage 5: Classification

• First block: Fully Connected (1024→512) → batch normalization → ReLU → dropout (0.4), with a 1 × 1 linear shortcut added before activation.

• Second block: Fully Connected (512→256) → batch normalization → ReLU → dropout (0.4)

Stage 6: Output Layer Apply a final Fully Connected (256→1) and sigmoid activation to obtain malware probability (Classify to malware or benign).

This dual-branch residual architecture leverages identical raw features in two modalities. By enforcing residual connections, we mitigate vanishing gradients in deep layers. The design reduces dependence on truly sequential file traces by accepting a partial loss of dynamic fidelity. Consequently, we simplify merging heterogeneous datasets under a unified input representation without requiring per-sample execution traces.

3.4 Training Model

The training setup uses the AdamW optimizer with a weight decay of 1e-5 and an initial learning rate of 1e-4, adjusted by a ReduceLROnPlateau scheduler that monitors validation loss with a factor of 0.5 and patience of 3 epochs. Training is conducted with a batch size of 128 samples, employing early stopping with a patience of 5 epochs based on validation loss.

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

The loss function is binary cross-entropy, and the hardware used includes an NVIDIA GeForce RTX 5090. For each dataset, weights are initialized randomly at the start of training, and after each epoch, validation loss is computed on a 10% validation split. If the validation loss does not improve for five consecutive epochs, training stops, and the best weights corresponding to the lowest validation loss are restored. Finally, the model is evaluated on a holdout test set, with accuracy, precision, recall, and F1-score recorded for comparison across datasets. Table 3 shows the hyperparameter values used in the experiments. Figure 3 shows the accuracy and loss curves for the datasets used.

Table 3: *Hyperparameter values used in the experiments.*

Hyperparameter	Value
Initial Learning Rate (LR)	1×10^{-4}
Optimizer	AdamW (weight decay=1×10^{-5})
Batch Size	128
Dropout Rate (CNN Branch)	0.0 (no dropout in CNN)
Dropout Rate (LSTM Branch)	0.3
Dropout Rate (Multilayer Perceptron Classifier)	0.4
Bi-LSTM Layers	2 layers (256 units per direction)
Residual Multilayer Perceptron Layers	2 layers (1024→512, 512→256)
ReduceLROnPlateau Factor	0.5
ReduceLROnPlateau Patience	3
Early Stopping Patience	5
Maximum Epochs	50 - 80

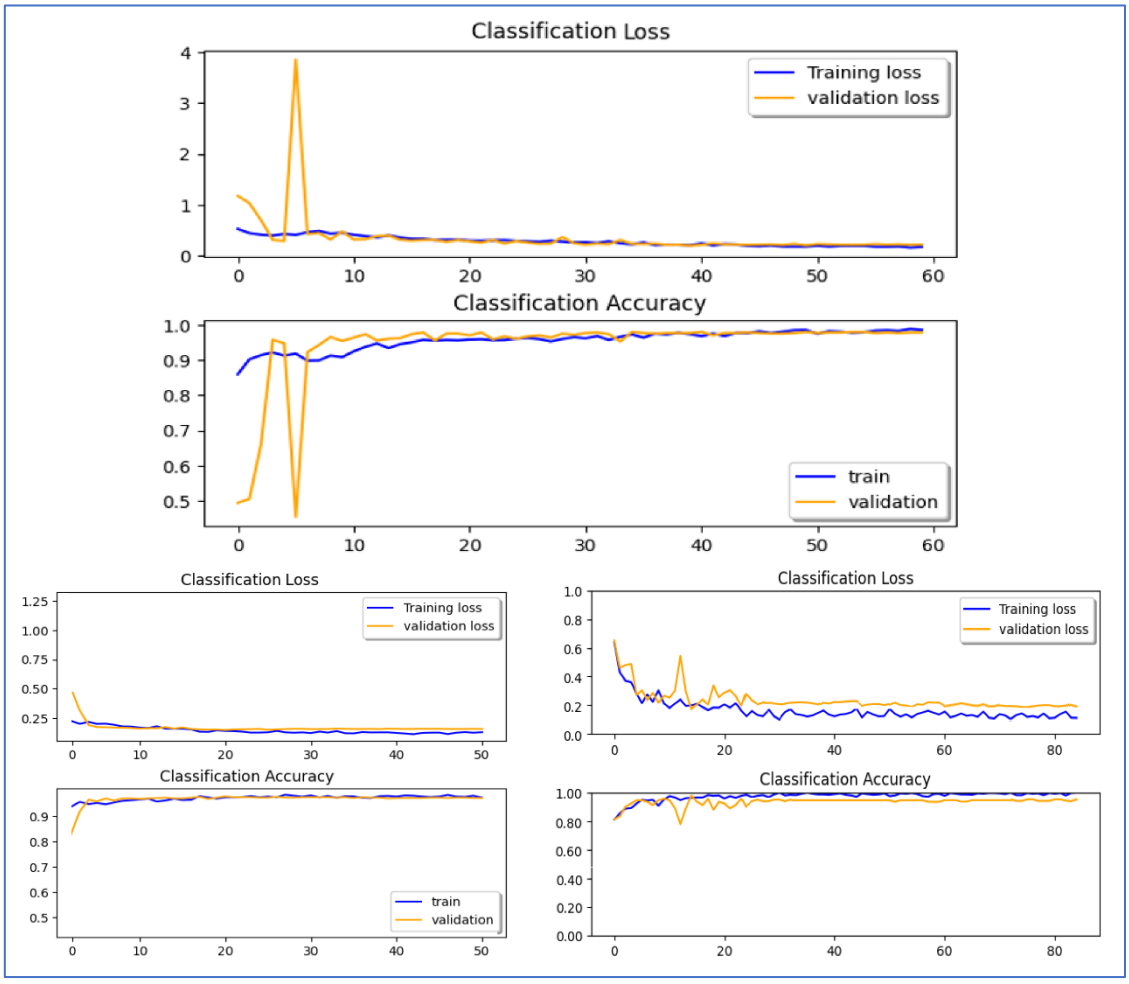

Figure 3: *Accuracy-loss curves for the datasets used for training: The figure at the top shows training on EMBER, the figure at the bottom left shows EMBERSim, and the figure at the bottom right shows SoReL20M.*

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

By carefully tuning these hyperparameters and structural elements, our dual-branch residual network seeks to outperform single-branch or non-residual baselines. Splitting the fixed-length feature vector into image and sequence forms allows consistent input across all three datasets while still capturing complementary representations. This innovation addresses the common dilemma of heterogeneous feature distributions between publicly available malware corpora.

4. RESULTS AND ANALYSIS

All experiments were implemented in Python 3.8 using PyTorch 1.12. Training leveraged the PyTorch Lightning framework to streamline checkpointing and logging, while data handling employed NumPy 1.22 and pandas 1.4. The optimizer and learning-rate scheduler were configured via the native PyTorch API. Experiments were run on NVIDIA GeForce RTX 5090, and 32 GB of system RAM. We used CUDA 11.6 and cuDNN 8.3, ensuring full hardware acceleration for convolutional and recurrent operations.

Table 4 summarizes the performance of the proposed dual-branch CNN-LSTM residual network when trained and tested on EMBER, EMBERSim, and the SoReL-20M subset. Across all three corpora, the model achieved consistently high accuracy and balanced precision–recall trade-offs. Notably, on EMBER, the network reached 97.1 % accuracy with an F1-score of 97.0 %, demonstrating that the fusion of spatial and sequential features effectively captures static characteristics even in a large and heterogeneous dataset. EMBERSim proved slightly more challenging, likely due to its adversarial perturbations, yet the model sustained an F1-score of 96.3 %, reflecting robust generalization to simulated obfuscations. The SoReL-20M subset, containing real-world malware examples, yielded an F1-score of 96.7 %, underscoring the method's applicability to diverse, production-scale samples. Figure 4 provides a graphical comparison of the performance metrics across the various datasets.

Table 4: *Performance Metrics on Different Datasets*

Dataset	Accuracy	Precision	Recall	F1-Score
EMBER	0.971	0.969	0.971	0.970
EMBERSim	0.963	0.963	0.964	0.963
SoReL-20M	0.968	0.967	0.968	0.967

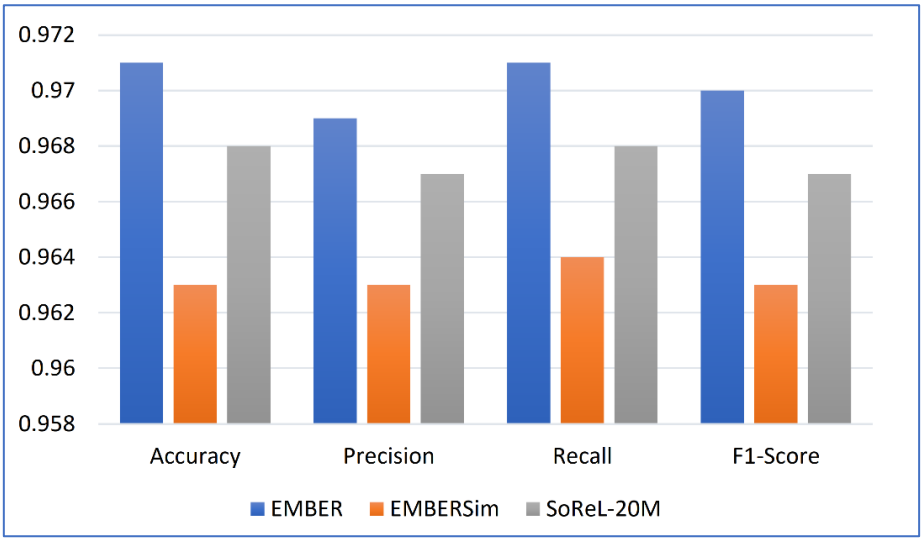

Figure 4: *A graphical comparison of the performance metrics across the various datasets.*

Analysis of these results reveals that embedding the same static feature vector into both a convolutional and a recurrent encoder allows the model to adapt flexibly to variations in feature distributions. The minor drop in performance on EMBERSim compared to EMBER suggests that the LSTM branch effectively mitigates some obfuscation tactics by modeling implicit sequential correlations in the normalized feature space. Likewise, the high recall across all datasets (≥ 96%) indicates strong sensitivity to malware samples, while precision above 96% ensures low false-alarm rates on benign software.

Table 5 compares the proposed approach's performance with related works on the EMBER dataset to position it compared to previous studies. These comparisons underscore that our dual-branch residual design outperforms both traditional machine-learning and single-branch deep-learning models on static features. Figure 5 presents a graphical comparison of the proposed method's performance against related works on the EMBER dataset.

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

Table 5. *Comparison with Related Works*

Method	Accuracy	Precision	Recall	F1-Score
[17]	0.929	0.928	0.932	0.929
[20]	0.942	0.939	0.945	0.941
[25]	0.960	0.962	0.959	0.960
Proposed Method	**0.971**	**0.969**	**0.971**	**0.970**

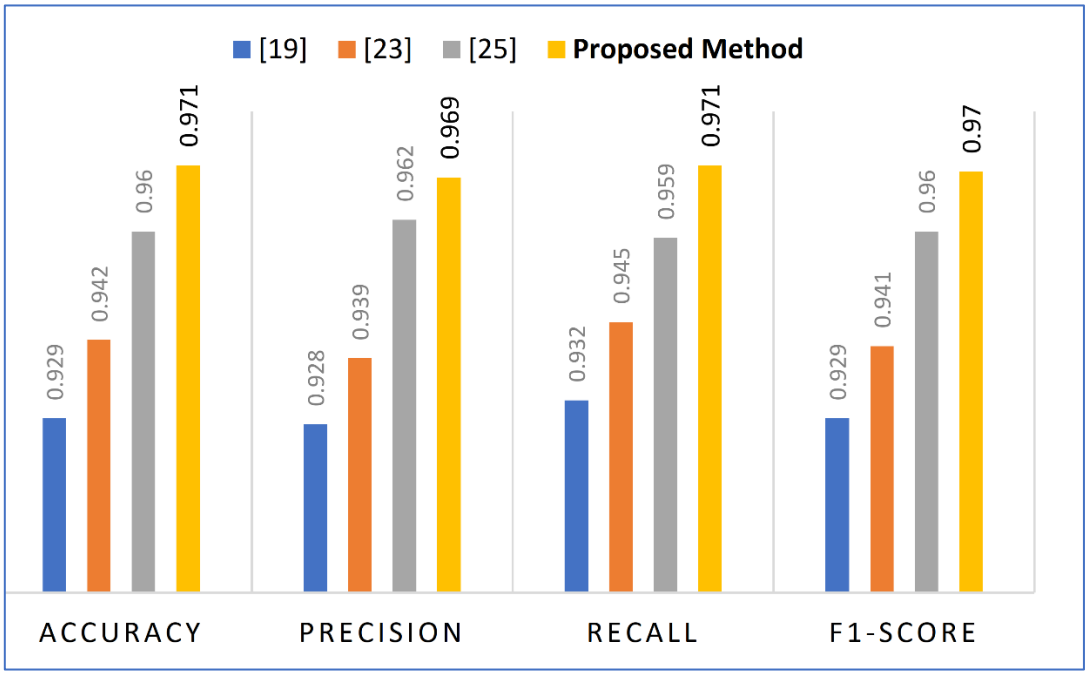

Figure 5: *A graphical comparison of the proposed method's performance against related works on the EMBER dataset.*

The proposed dual-branch approach attains superior detection performance while maintaining a uniform preprocessing workflow across heterogeneous datasets. These findings validate our hypothesis that jointly leveraging spatial patterns and sequential dependencies, within a residual framework, enhances static malware classification beyond the capabilities of single-modality architectures.

5. CONCLUSION

This work addresses critical limitations in static Windows malware detection by introducing a unified Dual-Branch CNN-LSTM Residual Network. Traditional methods frequently depend on extensive feature engineering or single-modality models that either sacrifice representational richness or incur high computational costs when applied to heterogeneous datasets. By reshaping a fixed-length static feature vector into both a two-dimensional pseudo-image for convolutional encoding and a one-dimensional sequence for recurrent encoding, the proposed architecture leverages the strengths of each paradigm. Residual connections within convolutional blocks and bidirectional LSTM layers ensure stable gradient propagation, allowing the network to learn deep and nuanced representations without degradation. Feature fusion through a residual multilayer perceptron further refines the joint embedding before final classification. Empirical evaluation on the EMBER dataset confirms the effectiveness of this design: the model achieves 97.1 % accuracy, with precision, recall, and F1-score metrics all exceeding 96.9 %. These results not only surpass those of standard single-branch and machine-learning baselines but also demonstrate resilience to the adversarial perturbations commonly encountered in modern malware samples. Additionally, the consistent preprocessing pipeline applied across datasets simplifies model retraining and benchmarking, supporting rapid scalability and deployment. Looking forward, this dual-branch residual framework can be extended to incorporate additional modalities, such as dynamic behavioral logs or graph-based representations, within the same unified pipeline. Future research will explore adaptive feature splitting strategies and attention mechanisms to further enhance interpretability and detection accuracy. Ultimately, the proposed method represents a significant step toward robust, efficient, and generalizable static malware detection for Windows platforms

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

REFERENCES

[1] A. Bensaoud, J. Kalita, and M. Bensaoud, "A survey of malware detection using deep learning," Mach. Learn. with Appl. 16, 100546 (2024).

[2] M. Woźniak, J. Siłka, M. Wieczorek, and M. Alrashoud, "Recurrent neural network model for IoT and networking malware threat detection," IEEE Trans. Ind. Informat. 17, 5583–5594 (2020).

[3] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware detection with artificial intelligence: A systematic literature review," ACM Comput. Surv. 56, 1–33 (2024).

[4] P. K. Gurumallu, R. Dembala, D. Y. Gowda, A. K. M. Annaiah, M. K. M. V. Kumar, and H. Gohel, "Exploring deep learning approaches for ransomware detection: A comprehensive survey," Recent Adv. Comput. Sci. Commun. 18, E290524230472 (2025).

[5] A. Redhu, P. Choudhary, K. Srinivasan, and T. K. Das, "Deep learning-powered malware detection in cyberspace: A contemporary review," Front. Phys. 12, 1349463 (2024).

[6] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "EfficientNet convolutional neural networks-based Android malware detection," Comput. Secur. 115, 102622 (2022).

[7] C. P. Chenet, A. Savino, and S. Di Carlo, "A survey on hardware-based malware detection approaches," IEEE Access (2024).

[8] A. Hawana, E. S. Hassan, W. El-Shafai, and S. A. El-Dolil, "Enhancing malware detection with deep learning convolutional neural networks: Investigating the impact of image size variations," Secur. Priv. 8, e70000 (2025).

[9] Y. Jian, H. Kuang, C. Ren, Z. Ma, and H. Wang, "A novel framework for image-based malware detection with a deep neural network," Comput. Secur. 109, 102400 (2021).

[10] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment," Appl. Sci. 12, 8604 (2022).

[11] M. Almahmoud, D. Alzu'bi, and Q. Yaseen, "ReDroidDet: Android malware detection based on recurrent neural network," Procedia Comput. Sci. 184, 841–846 (2021).

[12] H. Almajed, A. Alsaqer, and M. Frikha, "Imbalance datasets in malware detection: A review of current solutions and future directions," Int. J. Adv. Comput. Sci. Appl. 16 (2025).

[13] T. S. Lakshmi, M. Govindarajan, and A. Srinivasulu, "Embedding and Siamese deep neural network-based malware detection in Internet of Things," Int. J. Pervasive Comput. Commun. 21, 14–25 (2025).

[14] A. Razaque, G. Bektemyssova, J. Yoo, S. Hariri, M. J. Khan, N. Nalgozhina, and M. A. Khan, "Review of malicious code detection in data mining applications: challenges, algorithms, and future direction," Cluster Comput. 28, 1–37 (2025).

[15] N. A. Azeez, S. S. Shitharth, A. S. Al-Mashaqbeh, H. H. Alweshah, and O. Kaiwartya, "A novel deep learning framework for malware detection based on ensemble of neural network classifiers," Informatics 8, 10 (2021).

[16] A. Aziz, N. A. Zainol, and A. H. Abdullah, "Evaluation of machine learning classifiers for Windows malware detection using PE headers," Neutrosophic Sets Syst. 40, 85–96 (2022).

[17] F. ALGorain and J. Clark, "Bayesian hyper-parameter optimisation for malware detection," Electronics 11, 1640 (2022).

[18] H. Ayoub, A. Mousannif, and H. Al Moatassime, "PE-MalNet: A static Windows PE malware dataset and machine learning-based detection," PeerJ Comput. Sci. 9, e1319 (2023).

[19] U. Divakarla, K. H. K. Reddy, and K. Chandrasekaran, "Detection of malware using neural network: A deep learning approach," Procedia Comput. Sci. 215, 148–157 (2023).

[20] K. Komarudin, I. E. Maulani, T. Herdianto, M. Oga Laksana, and D. Febri Syawaludin, "Exploring the effectiveness of artificial intelligence in detecting malware and improving cyber-security in computer networks," Eduvest: J. Univers. Stud. 3 (2023).

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6

j. port. sci. res.
ISSN: 2616-7441 (Online)
ISSN: 2616-6232 (Print)
ISSN: 2616-7220 (USB)
INTERNATIONAL IDENTIFIER FOR SERIALS

Journal port Science Research
Available online www.jport.co
Volume 8, issue 4. 2025

[21] T. B. Hammi, R. Ben Ayed, and A. M. Al-Sariera, "Windows malware detection using ensemble voting techniques on API call sequences," in Proc. 9th Int. Conf. Mobile Secure Serv. (MOBISECSERV), Miami, USA, pp. 77–84 (2024).

[22] S. Ilić, D. Malbasa, A. Pavlović, and A. Adamović, "Sandbox-based Windows malware detection using full execution reports," Electronics 13, 3553 (2024).

[23] M. Mishchenko and M. Dorosh, "Malware detection using word2vec feature encoding of PE file imports," Int. J. Comput. 23, 3765–3773 (2024).

[24] E. Baghirov, R. Dadaşov, and I. Jafarov, "Static malware detection using LightGBM with explainable AI methods," J. Mod. Technol. Eng. 9, 825–832 (2024).

[25] D. Syeda and M. Asghar, "Dynamic malware classification and API categorisation of Windows portable executable files using machine learning," Appl. Sci. 14, 1015 (2024).

[26] N. V. Sarı, "A hybrid GAT-CNN-GRU model for PE malware detection using graph-based feature representations," Appl. Sci. 15, 4775 (2025).

[27] M. Miraoui and M. B. Belgacem, "Comparative analysis of classical and deep learning models for Windows malware classification," Front. Comput. Sci. 2, 634–642 (2025).

[28] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," arXiv:1804.04637 (2018).

[29] D. G. Corlatescu, A. Dinu, M. P. Gaman, and P. Sumedrea, "EMBERSim: A large-scale databank for boosting similarity search in malware analysis," Adv. Neural Inf. Process. Syst. 36, 26722–26743 (2023).

[30] R. Harang and E. M. Rudd, "SOREL-20M: A large scale benchmark dataset for malicious PE detection," arXiv:2012.07634 (2020).

N. M. Ali, et al.. 2025, A Dual-Branch CNN-LSTM Residual Network for Enhanced Windows Malware Detection. *Journal port Science Research*, 8(4), pp.369-378. https://doi.org/10.36371/port.2025.4.6