Journal of COMPUTER AND FORENSIC SCIENCES

CFS 2026, Vol. 5, Issue 1, pp. 31–40
https://doi.org/10.5937/jcfs5-67309
Original research paper

A Hybrid Machine Learning Framework for Dynamic Malware Classification Using CNN, LSTM, and Random Forest Models

Karthick Ganapathy[1,2]

[1] Department of Information Technology, SRH University Heidelberg, Germany, Student

[2] Just Access e.V., Germany

Corresponding author: karthicksekarg@gmail.com

Received: May 16, 2026 • Accepted: June 10, 2026 • Published: July 10, 2026

Abstract: Malware continues to evolve through polymorphic, metamorphic, and fileless techniques, reducing the effectiveness of signature-based detection. This study proposes a hybrid dynamic malware classification framework that combines Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and a Random Forest (RF) classifier. Dynamic behavioural traces are transformed into machine-readable features using Term Frequency-Inverse Document Frequency (TF-IDF), Count Vectorization, and Word2Vec embeddings. CNN and LSTM models are used as feature extractors to capture local and sequential behavioural patterns, while RF performs the final classification to improve robustness and interpretability. The reported experimental evaluation shows that the embedding-based hybrid CNN-RF configuration achieved the best performance, reaching 98.21% accuracy. The findings indicate that combining deep representation learning with ensemble classification can improve dynamic malware detection and reduce dependence on static signatures. The proposed framework is therefore a promising approach for adaptive malware classification and future security monitoring applications.

Keywords: dynamic malware analysis; malware classification; CNN; LSTM; Random Forest.

1. INTRODUCTION

Malware remains one of the most persistent and adaptive cybersecurity threats. Modern malicious programs frequently use polymorphism, metamorphism, packing, and fileless execution to avoid conventional detection. Signature-based controls are useful for known malware, but they are often less effective against new, obfuscated, or rapidly modified variants. This creates a need for intelligent classification methods that can learn behavioural patterns rather than relying only on fixed signatures.

Malware analysis is commonly divided into static, dynamic, and hybrid approaches. Static analysis inspects code, file structure, imports, strings, or byte patterns without execution. It is efficient but vulnerable to obfuscation and packing. Dynamic analysis executes the

Citation: K, Ganapathy. A Hybrid Machine Learning Framework for Dynamic Malware Classification Using CNN, LSTM, and Random Forest Models" Journal of Computer a*nd Forensic Sciences*, vol. 5, no. 1, pp. 31–40, 2026, https://doi.org/10.5937/jcfs5-67309.

sample in a controlled environment and observes runtime behaviour such as API calls, file-system modifications, registry access, process creation, and network activity. Prior work has shown that dynamic behaviour is useful for detecting malware variants that may hide their static structure [1], [2].

Recent malware detection research increasingly treats behavioural traces as sequential data. API call sequences, opcode sequences, and sandbox logs can be represented similarly to natural language, where the order and context of tokens carry useful meaning. This makes text-representation methods and deep learning architectures suitable for malware classification. Recurrent models have been used to learn malware behaviour from API call sequences [3], while convolutional and recurrent architectures have been combined to capture both local n-gram patterns and longer sequential dependencies [4], [5].

This study proposes a hybrid framework that combines CNN, LSTM, and RF models. CNN layers extract local spatial patterns from vectorized behavioural sequences, LSTM layers learn sequential dependencies, and RF performs the final classification using high-level representations extracted from the neural models. The approach is motivated by the need to combine deep representation learning with the stability and interpretability of ensemble learning.

This manuscript is derived from the author's Master's thesis work conducted at SRH University Heidelberg, where the original experimental design and implementation of the hybrid CNN, LSTM, and Random Forest-based malware classification framework were developed [6]. The present article refines that thesis work for journal publication by strengthening the methodological description, dataset explanation, and interpretation of the results.

The main contributions of this work are as follows: (i) a hybrid architecture for dynamic malware classification using CNN, LSTM, and RF; (ii) a comparison of TF-IDF, Count Vectorization, and Word2Vec-based representations; (iii) a feature-extraction strategy in which CNN and LSTM models provide high-level features to RF; and (iv) an evaluation showing that embedding-based hybrid configurations can outperform standalone approaches in the reported experimental setting.

1.1. Related work and research gap

Damodaran et al. compared static, dynamic, and hybrid analysis for malware detection, emphasizing that different analysis modes provide complementary information [1]. Dynamic approaches are particularly relevant when behavioural evidence is more informative than code appearance, especially under obfuscation.

Pascanu et al. introduced recurrent neural networks for malware classification using event sequences, showing that sequence modelling can capture behavioural regularities [3]. Kolosnjaji et al. later combined convolutional and recurrent neural layers to classify malware system call sequences, demonstrating the value of hierarchical feature extraction [4]. Zhang et al. proposed a dynamic malware analysis method that combines feature engineering and deep learning over API call sequences, highlighting the importance of both representation design and architecture selection [2].

More recent work has explored API-call-based deep learning and embedding approaches. API-MalDetect uses CNN and BiGRU-based feature extraction over API call sequences and reports strong performance on benchmark datasets [7]. Aggarwal and Di Troia

used dynamically extracted API call embeddings with classifiers including RF and CNN, demonstrating that embeddings can improve malware classification [8]. Bensaoud and Kalita proposed a CNN-LSTM design using opcodes and API calls and reported very high performance on large malware datasets [5].

Despite this progress, several studies focus on either a single neural architecture or direct neural classification. This work addresses that gap by using deep models primarily as feature extractors and RF as the final classifier. The goal is to preserve the pattern-learning strengths of CNN and LSTM while benefiting from RF generalization, reduced overfitting risk, and feature importance analysis [9].

2. MATERIALS AND METHODS

2.1. Research design

The proposed framework performs binary malware classification, distinguishing benign software from malicious software using dynamic behavioural traces. Each sample is represented as a text-like sequence derived from runtime behaviour, such as API calls, system call sequences, or execution logs. These sequences are preprocessed and converted into numerical representations before being passed to machine learning and deep learning models.

The study evaluates three representation families: TF-IDF, Count Vectorization, and Word2Vec embeddings. These representations are used with RF, CNN, and LSTM configurations. In the hybrid design, CNN and LSTM models are trained to learn discriminative feature representations. Instead of relying only on their sigmoid output layers, the penultimate-layer features are extracted and passed to RF for final classification (Figure 1).

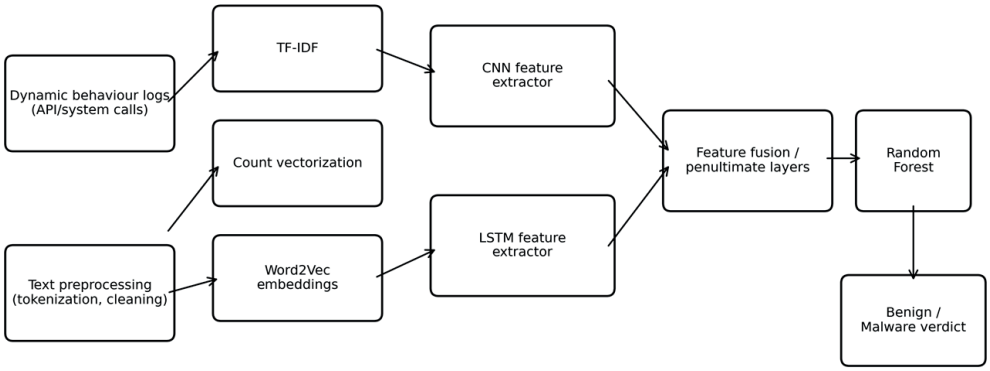

Figure 1. *Overview of the proposed hybrid dynamic malware classification framework.*

2.2. Dataset description

The dataset used in this study was derived from Cuckoo sandbox reports, as documented by Ilić et al. [10]. It was curated to capture the dynamic behaviour of malware and benign

software by analysing system interactions, including API calls, file-system modifications, registry alterations, and network activity.

The original dataset comprised 22,200 labelled samples, including 11,735 benign instances and 10,465 malware samples. Malware samples were sourced from VirusTotal, while benign samples were collected from real-world Windows 10 system files and correspondence documents. This composition provided a broad basis for training and evaluating dynamic malware classification models.

During dataset generation, Ilić et al. [10] observed that 23.78% of the collected sandbox reports lacked API-call data. This observation supports the inclusion of additional behavioural feature sets beyond API-call sequences. Their evaluation showed that classification accuracy improved from 95.56% using only API-call sequences to 99.74% when comprehensive sandbox reports with diverse behavioural features were used [10].

For the present study, a subset of the original dataset was selected to reduce computational overhead while preserving the structural characteristics of the full dataset. The selected subset comprised 622 samples, including 322 benign and 300 malware instances. This subset was used as the benchmark dataset for evaluating the proposed CNN, LSTM, and RF-based hybrid malware classification framework.

The feature extraction process in this research follows the logic of the original dataset methodology by incorporating key behavioural indicators such as API calls, registry modifications, file-system changes, and network activity logs. By integrating behavioural features beyond API-call sequences, the dataset improves the model's ability to learn richer execution patterns and generalize to real-world malware threats.

2.3. Data preprocessing

Dynamic behavioural records were first cleaned and normalized to reduce noise. The preprocessing workflow included tokenization, removal of irrelevant punctuation, normalization of spacing, and conversion of behavioural logs into sequences suitable for vectorization. Where applicable, duplicated or empty records were removed to prevent bias during model training.

Labels were encoded numerically for binary classification. The dataset was divided into training and testing subsets to evaluate generalization. The source experiment used a 60:40 train-test split and compared multiple feature extraction and model combinations. Hyperparameter tuning was then applied to improve model stability and reduce overfitting.

2.4. Feature extraction

Feature extraction transforms raw behavioural text into structured numerical vectors. The study compares sparse frequency-based representations and dense embedding-based representations.

TF-IDF: Term Frequency-Inverse Document Frequency assigns higher weight to terms that are frequent in a sample but less common across the corpus. It reduces the influ-

ence of common tokens and highlights more informative behavioural indicators. The representation can be expressed as:

$$TF\text{-}IDF(t,d,D) = tf(t,d) \times \log(N / df(t,D)) \quad (1)$$

where t is a term, d is a document or behavioural sequence, D is the corpus, N is the number of documents, and df(t,D) is the document frequency of term t.

Count Vectorization: Count Vectorization converts each behavioural record into a sparse vector of raw token counts. It is simple, interpretable, and effective for baseline classifiers such as RF. However, it does not capture semantic similarity or token position.

Word2Vec embeddings: Word2Vec represents tokens as dense vectors in a continuous space, allowing semantically similar behaviours to be positioned closer together [11]. In malware analysis, this helps represent related API calls or behaviour tokens more flexibly than sparse vectors. Embedding-based representations were especially useful for the CNN-based feature extraction pipeline.

2.5. Model architecture

The hybrid model consists of three main learning components: CNN, LSTM, and RF. CNN and LSTM learn high-level features from behavioural sequences, while RF performs the final classification decision (Table 1).

Table 1. *Model components used in the proposed hybrid framework.*

Component	Configuration / role	Purpose in the framework
CNN	1D convolution with 128 filters, kernel size 3; GlobalMaxPooling1D; dense layers; dropout; sigmoid output during neural training	Captures local n-gram-like behavioural patterns and produces high-level spatial features.
LSTM	Single LSTM layer with 128 units; batch normalization; dense layers; dropout; sigmoid output during neural training	Models temporal dependencies and long-range relationships in behavioural sequences.
Random Forest	Final ensemble classifier trained on extracted neural features	Improves robustness, reduces overfitting risk, and supports feature-importance-based interpretation.

CNN feature extractor: The CNN model is designed to detect localized behavioural patterns from vectorized input. A 1D convolutional layer captures short token combinations, followed by pooling and dense layers that compress the learned representation. The output from the penultimate dense layer is extracted as a learned feature vector for RF classification.

LSTM feature extractor: The LSTM model captures sequential behaviour by maintaining memory across token positions. This is useful when malware actions are staged across time, for example, when a sample first performs reconnaissance, then modifies files, then initiates network communication. The final hidden representation is passed to RF.

Random Forest classifier: RF is used as the final classifier because it combines multiple decision trees and can improve generalization through bagging [9]. In this framework, RF receives high-level CNN or LSTM features rather than raw behavioural tokens.

2.6. Training and evaluation

The models were trained and evaluated using binary labels. Neural models used dropout to reduce overfitting and batch normalization in the LSTM pipeline to stabilize training. The experimental process compared representation methods and model combinations, then selected the best-performing hybrid configuration.

Performance was evaluated primarily using accuracy, with additional attention to false positives, false negatives, and macro-average classification behaviour. For security use cases, false negatives are especially critical because undetected malware can lead to compromise, while false positives can increase analyst workload. A robust malware classifier should therefore balance accuracy with operational reliability.

3. RESULTS

3.1. Experimental result summary

The experimental results reported in this section are based on the implementation and evaluation pipeline originally developed in the author's Master's thesis [6]. The evaluation compared feature-representation strategies, including TF-IDF, Count Vectorization, and Word2Vec embeddings, together with CNN, LSTM, RF, and hybrid deep-feature-extraction configurations.

The results are interpreted at both representation and model levels. Sparse representations such as TF-IDF and Count Vectorization provided useful baseline behaviour, while Word2Vec embeddings offered a denser representation of behavioural relationships between runtime events. The strongest reported configuration was the Word2Vec-based CNN-RF hybrid model, which achieved 98.21% accuracy on the selected Cuckoo sandbox subset.

The reported evaluation indicates that the hybrid strategy improved classification performance compared with standalone methods. Among the tested representation methods, Word2Vec-based embeddings paired with CNN feature extraction and RF classification achieved the best reported accuracy of 98.21%. This suggests that dense representations are more suitable than sparse token-count approaches when the model must learn semantic and contextual relationships between behavioural events.

The results also support the use of deep feature extraction before ensemble classification. CNN layers captured local behavioural patterns, while RF improved the final decision

boundary using the learned feature vectors. LSTM-based features were useful for sequential behaviour modelling, especially where malware actions appear in meaningful temporal order. However, the reported best-performing configuration was embedding-based CNN feature extraction with RF.

Table 2. *Performance comparison of the experimental model configurations.*

To show the different experiments conducted in the thesis-derived evaluation, the main classification results are summarized below. All reported values are percentages; where later hyperparameter tuning was reported, the improvement is noted in the final column.

Feature representation	Model configuration	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Macro F1 (%)	Result note
Count Vectorization	Random Forest	97.14	98.26	94.96	96.58	97.06	Strong frequency-based RF baseline.
TF-IDF	Random Forest	98.57	99.15	97.48	98.31	98.54	Best standalone RF baseline.
TF-IDF	CNN + Random Forest	80.80	81.11	73.74	77.25	78.50	Lower performance with sparse weighted features.
Count Vectorization	CNN + Random Forest	90.76	91.09	86.79	88.89	90.00	Improved CNN feature extraction compared with TF-IDF-CNN-RF.
TF-IDF	LSTM + Random Forest	90.63	93.33	84.85	88.89	90.49	Sequential extraction improved over TF-IDF-CNN-RF.
Count Vectorization	LSTM + Random Forest	90.18	95.29	81.82	88.04	90.41	After tuning, accuracy increased to 92.41%.
Word embeddings	LSTM + Random Forest	88.50	92.00	80.00	85.00	86.50	After tuning, accuracy increased to 91.07%.
Word embeddings	CNN + Random Forest	98.21	97.98	95.96	96.94	97.79	Best overall hybrid configuration.

The comparison shows that traditional RF baselines performed strongly with TF-IDF and Count Vectorization, while hybrid deep-feature extraction produced mixed results depending on the representation. The Word embedding-CNN-RF configuration achieved the strongest overall result with 98.21% accuracy, supporting its selection as the final hybrid configuration (Table 2).

The strongest result of the study is the reported 98.21% accuracy from the embedding-based hybrid model. In a malware analysis context, this is significant because it demonstrates that behavioural sequence representation and deep feature extraction can

improve detection beyond simple static signatures. The results are consistent with recent work showing the value of API call embeddings, CNN-LSTM structures, and hybrid deep learning approaches for malware classification [5], [7], [8].

4. DISCUSSION

The proposed framework is effective because it combines complementary model strengths. TF-IDF and Count Vectorization provide interpretable frequency-based baselines, while Word2Vec embeddings capture behavioural similarity between tokens. CNNs are useful for extracting local sequence patterns, LSTMs capture temporal dependencies, and RF improves final classification stability.

From a digital forensics perspective, dynamic malware classification is valuable because it can support evidence-driven analysis. Behavioural indicators such as API calls, file operations, registry changes, and network activity can help investigators understand what a sample attempted to do. A model that learns from these behaviours may provide better investigative value than a purely static signature match.

The use of RF as the final classifier can also support interpretability. While deep neural networks often behave as black-box models, RF can provide feature importance values. When RF is trained on extracted neural features, analysts can identify which learned behavioural dimensions contributed most strongly to a classification decision. This is important for cybersecurity operations because analysts need evidence, not only a prediction.

However, the approach also has limitations. Dynamic analysis can be resource-intensive because samples must be executed in controlled environments. Malware may detect sandbox conditions and modify its behaviour. The performance of the model also depends on the quality and diversity of the dataset. If the training data does not include modern or evasive malware families, the model may not generalize well to real-world threats.

Future work should validate the model on public benchmark datasets and include a detailed confusion matrix, precision, recall, F1-score, ROC-AUC, and runtime overhead. Additional explainability methods, such as SHAP values or feature attribution over behavioural tokens, could make the framework more useful for SOC analysts and forensic investigators.

5. CONCLUSIONS

This study presented a hybrid machine learning framework for dynamic malware classification using CNN, LSTM, and RF models. Dynamic behavioural traces were transformed using TF-IDF, Count Vectorization, and Word2Vec embeddings. CNN and LSTM models were used to extract high-level behavioural features, and RF was used as the final classifier.

The reported results show that the Word2Vec-based CNN-RF hybrid configuration achieved the best performance, reaching 98.21% accuracy. This supports the conclusion that dense embedding representations and deep feature extraction can improve malware classification when combined with ensemble learning. The framework provides a practi-

cal foundation for adaptive malware detection, digital forensics support, and future SOC integration.

Future research should extend the evaluation to larger datasets, multi-class malware family classification, real-time sandbox pipelines, and explainable AI methods that provide analyst-friendly evidence for each classification decision.

FUNDING:

This research received no external funding.

ACKNOWLEDGMENTS:

The author gratefully acknowledges SRH University Heidelberg for academic support during the development of this research work. The author also thanks Just Access e.V. for providing practical cybersecurity experience that helped strengthen the applied perspective of this study.

AUTHOR CONTRIBUTIONS:

Karthick Ganapathy: conceptualization, methodology, implementation, analysis, writing-original draft, and writing-review and editing.

INSTITUTIONAL REVIEW BOARD STATEMENT:

Not applicable. This study did not involve human participants, animals, cell lines, or plants.

INFORMED CONSENT STATEMENT:

Not applicable. This study did not involve human participants.

DATA AVAILABILITY STATEMENT:

The experimental dataset and implementation details can be made available by the author upon reasonable request, subject to malware-handling safety and institutional restrictions.

CONFLICTS OF INTEREST:

The author declares no conflict of interest.

REFERENCES

[1] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, :A comparison of static, dynamic, and hybrid analysis for malware detection," Journal of Computer Virology and Hacking Techniques, vol. 13, pp. 1–12, 2017.

[2] Z. Zhang, P. Qi, and W. Wang, "Dynamic malware analysis with feature engineering and feature learning," in Proc. AAAI Conference on Artificial Intelligence, vol. 34, no. 1, 2020, pp. 1210–1217.

[3] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1916–1920.

[4] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in Australasian Joint Conference on Artificial Intelligence, Lecture Notes in Computer Science, vol. 9992, 2016, pp. 137–149.

[5] A. Bensaoud and J. Kalita, "CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls," Knowledge-Based Systems, Art. no. 111543, 2024, doi: 10.1016/j.knosys.2024.111543.

[6] K. Ganapathy, "A Hybrid Machine Learning Framework for Dynamic Malware Classification Using CNN, LSTM, and Random Forest Models," Master's thesis, Department of Information Technology, SRH University Heidelberg, Heidelberg, Germany, 2025.

[7] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "API-MalDetect: Automated malware detection framework for Windows based on API calls and deep learning techniques," Journal of Network and Computer Applications, vol. 218, Art. no. 103704, 2023, doi: 10.1016/j.jnca.2023.103704.

[8] S. Aggarwal and F. Di Troia, "Malware classification using dynamically extracted API call embeddings," Applied Sciences, vol. 14, no. 13, Art. no. 5731, 2024, doi: 10.3390/app14135731.

[9] L. Breiman, "Random forests," Machine Learning, vol. 45, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[10] S. Ilić, M. Gnjatović, I. Tot, B. Jovanović, N. Maček, and M. Gavrilović Božović, "Going beyond API Calls in Dynamic Malware Analysis: A Novel Dataset," Electronics, vol. 13, no. 17, Art. no. 3553, 2024, doi: 10.3390/electronics13173553.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv:1301.3781, 2013.