Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 11 Issue 12, December - 2025

Integration of Cyber Threat Intelligence with Runtime Parameters for Advanced Dynamic Malware Classification Using a Hybrid CNN-LSTM Architecture

Precious D. Agburuga[1]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
agburugapd@fuotuoke.edu.ng

Akpasam Joseph Ekanem[2]
Department of Electrical and Electronic Engineering,
Akwa Ibom State University Mkpat Enin, Akwa Ibom State
ajekanem56@yahoo.com

AGUIYI Nduka Watson[3]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
aguiyiwatson@gmail.com; aguiyinw@fuotuoke.edu.ng

***Abstract*—The rapid evolution of sophisticated malware variants poses a significant challenge to conventional detection systems, which often fail to account for the broader contextual environment of an attack. This paper proposes an integrated framework for advanced dynamic malware classification by fusing Cyber Threat Intelligence (CTI) with runtime parameters derived from sandbox execution. Leveraging a hybrid CNN-LSTM architecture, the model captures both the spatial characteristics of malicious API call clusters through 1D-Convolutional layers and the long-term temporal dependencies of execution sequences via Long Short-Term Memory units. The study utilizes malware samples from VirusShare and VirusSample repositories, executed in a controlled Cuckoo Sandbox to extract dynamic behavioral data, augmented with structured CTI from the IEEE DataPort CTI dataset generated from public security reports. Experimental results demonstrate that the proposed hybrid approach achieves a classification accuracy of 96.7%, a precision of 95.8%, and an F1-score of 96.0%, representing significant improvements over standalone CNN (88.4%) and LSTM (91.2%) baselines. This integration reduces false positives by providing critical metadata on known threat actor TTPs, proving that context-aware deep learning is essential for defending against modern, evasive malware.**

Keywords*—Cyber Threat Intelligence; Dynamic Malware Classification; Hybrid CNN-LSTM; API Call Sequencing; Runtime Parameters; Deep Learning; IEEE DataPort; VirusShare; Ransomware Detection; Threat-Informed Defense.

I. INTRODUCTION

A. Background

The cybersecurity landscape is undergoing a fundamental transformation. Modern malware authors no longer rely on simple, easily detected binary signatures. Instead, they deploy sophisticated, context-aware techniques, including polymorphism, metamorphism, and Living-off-the-Land (LotL) tactics, that exploit legitimate system utilities to blend into normal operating system activity [1]. The proliferation of malware variants is accelerating, with global statistics exceeding one million new malware samples identified daily [2], placing conventional signature-based antivirus systems under severe strain. A critical shift in research focus is therefore underway: from reactive, pattern-matching methodologies toward proactive, behavioral analysis powered by deep learning.

Dynamic malware analysis, defined by observing behavior during controlled execution, serves as a robust complement to traditional static analysis. Process-driven sequences of Application Programming Interface (API) calls offer semantic richness, effectively capturing malicious intent while remaining resilient to binary obfuscation or encryption [3]. State-of-the-art performance in API-based classification is currently achieved through deep learning, particularly architectures combining CNNs for spatial pattern extraction and LSTMs for modeling temporal dependencies [4], [5].

B. Problem Statement

Despite the advances in dynamic analysis, standalone behavioral models face a critical limitation: they operate in isolation, without access to the broader *contextual intelligence* about threat actors, known campaigns, and historical IOCs (Indicators of Compromise). This creates a blind spot in the detection pipeline. A standard LSTM may fail to differentiate between a legitimate software update agent, which also calls InternetOpen and WriteFile, and a malware downloader, because the API sequence alone is insufficient to resolve the ambiguity. Furthermore, highly evasive malware families that employ anti-debugging techniques (such as extended Sleep calls or IsDebuggerPresent checks) can suppress their malicious behavior within sandbox execution windows, deliberately producing benign-looking API sequences [3], [6].

A further challenge is the severe class imbalance between malware families in real-world repositories. While Ransomware and Trojans are widely represented, emerging threat families such as sophisticated spyware or supply-chain implants may have very few labeled samples, leading to biased classifiers with high false-negative rates for rare families. Existing approaches largely treat CTI as a separate, human-analyst-consumed artifact rather than as a machine-readable, integrable signal that can directly augment deep learning inference.

C. Proposed Solution and Contributions

This paper presents CTI-DynMal, an integrated deep-learning framework designed to overcome deficiencies in existing malware analysis techniques. By fusing structured threat intelligence with Cuckoo-generated dynamic features—derived from VirusShare/VirusSample repositories—the framework provides superior detection accuracy. We demonstrate that a hybrid 1D-CNN + LSTM model can concurrently capture complex spatial motifs and temporal behaviors, resulting in a more robust and contextualized detection model. The primary contributions of this work are:

i. CTI-Augmented Feature Fusion: A novel fusion layer that aligns temporal runtime sequences (API call embeddings) with static CTI vectors (TF-IDF representations of IOCs, CVEs, and TTP descriptors from IEEE DataPort), creating a unified multi-dimensional input that provides both local behavioral evidence and global threat context.

ii. Hybrid 1D-CNN + LSTM Architecture: The CNN component extracts local spatial "motifs" within API sequences (such as, the rapid CreateFile/WriteFile loops characteristic of ransomware), whose high-level representations are then processed by the LSTM for temporal dependency modeling across the full execution trace [4], [5].

iii. Comprehensive Multi-Dataset Validation: Evaluation on malware samples spanning five families (Ransomware, Trojan, Spyware, Worm, Benign) from VirusShare and VirusSample repositories, demonstrating 96.7% accuracy and a 2.9% False Positive Rate—outperforming all evaluated baselines.

iv. Ablation Study: Systematic analysis of the contribution of each architectural component (embedding, CNN, LSTM, CTI fusion) to final classification performance, providing empirical justification for design choices.

D. Paper Organization

Section II reviews related work in deep learning-based malware analysis, CTI frameworks, and hybrid architectures. Section III describes the datasets, preprocessing pipeline, and the proposed hybrid CNN-LSTM architecture. Section IV details the experimental setup and implementation. Section V presents results and discussion. Section VI concludes with future research directions.

II. LITERATURE REVIEW

A. Deep Learning for Dynamic Malware Analysis

The application of deep learning to dynamic malware analysis via API call sequences has produced a substantial body of research. Li et al. [3] proposed a novel deep framework for dynamic malware detection based on API sequence intrinsic features, demonstrating that self-attentive representations of API call contexts significantly outperform raw sequence models on the Windows PE family classification task. Bensaoud and Kalafut [4] proposed a CNN-LSTM hybrid model trained on opcode sequences and API calls collected from VirusShare and VirusTotal (2019–2023), achieving 99.91% accuracy using N-gram (N=8) sequence augmentation—establishing a strong benchmark for API call-based classification.

Thakur et al. [5] introduced a hybrid deep learning approach combining LSTM and CNN for malware detection by converting binary features into grayscale images and extracting dynamic API call features, demonstrating that the architectural synergy between spatial CNN extraction and temporal LSTM modeling consistently outperforms either model in isolation. Akhtar and Feng [6] applied the CNN-LSTM technique specifically for botnet detection using IoT API traffic, showing its effectiveness for temporal behavioral sequence classification with high precision. For the specific challenge of malware executed in sandbox environments, Ma et al. [7] proposed a directed API call graph model that captures not only the individual calls but also their structural and directional relationships, addressing the API call interleaving problem in multi-process malware.

B. Cyber Threat Intelligence (CTI) and Machine Learning

Cyber Threat Intelligence (CTI) has evolved from a human-analyst discipline into an increasingly machine-processable resource. Schlette et al. [8] conducted a comprehensive comparative study on CTI, identifying that the integration of structured CTI (expressed in STIX 2.1 format over TAXII feeds) into security operations significantly improves detection response time and reduces mean time to remediation. The STIX/TAXII standard, maintained by MITRE and OASIS, provides the canonical format for exchanging CTI objects including Indicators, Malware objects, Campaign relationships, and Threat Actor attributions [9].

Rashid et al. [10] provided a systematic review of CTI strategies, concluding that while CTI significantly improves the ability to predict and prevent cyber threats, challenges in data standardization and trust between organizations persist. Notably, Huang et al. [10] developed MAMBA, which leverages the MITRE ATT&CK framework alongside deep neural networks and attention mechanisms to map TTPs directly to specific API calls—the closest prior work to the CTI integration proposed in this paper. Albalawi and Alhalabi [11] demonstrated that integrating IOCs from public threat feeds with machine learning classifiers in a MISP-based architecture achieves 99.79% accuracy for threat detection in IoT networks, underscoring the potential of CTI-augmented classifiers.

C. Hybrid CNN-LSTM Architectures for Security Applications

The combination of CNN spatial feature extraction with LSTM temporal modeling has proven highly effective across a range of cybersecurity classification tasks. Bensaoud and Kalafut [4] demonstrated that a CNN-LSTM hybrid trained on malware-specific API sequences from VirusShare substantially outperforms both standalone CNN and LSTM models, as well as 14 pre-trained transfer learning architectures, in multi-family malware classification. A parallel approach by Hameed et al. [12] applied a 1D-CNN + LSTM model to IoT network intrusion detection, achieving 98.6% accuracy on the Edge-IIoTset dataset and confirming the architectural generalization across security domains.

Chen et al. [13], reviewing deep learning methods for malware classification, identified the integration of external contextual signals as the most underexplored frontier in API call-based detection, explicitly calling for research that combines CTI with behavioral deep learning—a gap that the proposed CTI-DynMal framework directly addresses. Maniriho et al. [14] introduced API-MalDetect, a deep learning framework for Windows PE malware detection using API calls that achieves strong multi-class performance, but similarly lacks CTI context integration.

D. Research Gap

The reviewed literature reveals a consistent and critical gap: while deep learning architectures for API call-based dynamic malware analysis have achieved high performance, and while CTI frameworks for threat sharing and enrichment are well-established, no prior work has systematically integrated structured IEEE DataPort CTI features directly into a CNN-LSTM classification pipeline at the feature fusion level. The proposed CTI-DynMal framework addresses this gap, providing the first demonstration that CTI-augmented hybrid deep learning delivers statistically significant improvements over behavioral-only approaches across a multi-family Windows malware benchmark.

III. METHODOLOGY

A. Data Acquisition and Sources

The study utilizes three primary data streams to ensure a comprehensive and contextually rich feature representation:

i. Malware Samples (VirusShare & VirusSample): Raw Windows Portable Executable (PE) binaries sourced from the VirusShare and VirusSample repositories, spanning five malware families: Ransomware, Trojan, Spyware, Worm, and a Benign class comprising legitimate software. VirusShare provides authenticated, hash-verified samples; VirusSample contributes family-labeled submissions from academic and industry contributors [4].

ii. Runtime Parameters (Cuckoo Sandbox): Dynamic behavioral logs generated by executing each binary in a hardened Cuckoo Sandbox environment (Windows 10 Guest VM, isolated from network) for a fixed duration of 300 seconds. The primary extracted feature is the sequence of Windows API calls made by the parent process, along with registry modifications, file system events, and network traffic (pcap) metadata.

iii. CTI Dataset (IEEE DataPort): A structured CTI dataset derived from public security reports published on IEEE DataPort, containing Indicators of Compromise (IOCs), CVE identifiers, MITRE ATT&CK Tactics/Techniques/Procedures (TTPs), and known threat actor attributions related to the malware families present in the binary corpus [8], [9].

B. Feature Engineering: Monitored API Categories

Based on the behavioral patterns documented in VirusShare samples and mapped to MITRE ATT&CK TTPs in the IEEE DataPort CTI reports, the following API call categories are defined as the primary feature vocabulary for the model. Table I presents the monitored API categories, their specific calls, and their malicious significance.

Table I. API call categories monitored for feature extraction, with specific Windows API functions and their malicious behavioral significance.

Category	Specific API Calls	Malicious Intent / Significance
Process Injection	VirtualAllocEx, WriteProcessMemory, CreateRemoteThread, NtUnmapViewOfSection	Core indicators of process hollowing or code injection into legitimate processes.
Persistence	RegSetValueEx, RegCreateKeyEx, CopyFile, MoveFileEx	Establishes persistence via Run registry keys or placing binaries in startup folders.
Information Theft	GetClipboardData, CryptAcquireContext, GetUserName, GetComputerName	Gathering system metadata, user credentials, or sensitive clipboard data.
Evasion / Anti-Debug	IsDebuggerPresent, CheckRemoteDebuggerPresent, NtQueryInformationProcess, Sleep	Detecting sandbox/analysis environments or delaying execution to bypass timers.
File Manipulation	CreateFile, ReadFile, WriteFile, DeleteFile, SetFileAttributes	Critical for ransomware detection (encryption loops) and secondary payload delivery.
Network / C2	InternetOpen, HttpOpenRequest, InternetConnect, socket, send, recv	Communication with Command & Control (C2) servers for instructions or data exfiltration.

C. Data Preprocessing and Feature Fusion Pipeline

To feed the hybrid model, the heterogeneous raw data is converted into a uniform numerical format through a three-stage pipeline:

1. **API Sequence Vectorization:** Each unique API call string is mapped to a unique integer index via a custom vocabulary dictionary (vocabulary size = 512). Sequences are either truncated or zero-padded to a fixed length of 500 consecutive calls, producing a uniform input tensor. An Embedding layer (d=64) then maps each integer index to a dense learned representation.
2. **CTI Vectorization:** Textual CTI reports and structured IOC metadata from the IEEE DataPort dataset are processed using **TF-IDF (Term Frequency-Inverse Document Frequency)** with a 128-dimensional output vector. This down-weights generic security terminology and amplifies high-specificity IOC tokens (file hashes, CVE identifiers, C2 domain patterns) that discriminate between malware families [8].
3. **Feature Fusion (CTI Mapping):** The IEEE DataPort CTI dataset is queried using the SHA-256 hash of the executed binary to retrieve the corresponding CTI context vector. This CTI vector is then concatenated with the LSTM output (post-sequence processing) to form a fused multimodal input for the Dense classification head. This ensures that the temporal sequence model operates with full awareness of the global threat context.

The research procedure is summarized in Table II.

Table II. End-to-end research procedure from data collection through model evaluation, with associated tools, datasets, and outputs.

Phase	Activity	Tools / Datasets	Output
1. Collection	Gather raw malware binaries & CTI reports	VirusShare, VirusSample, IEEE DataPort	Raw PE files, CTI JSON / CSV
2. Execution	Execute samples in sandbox to record API calls	Cuckoo Sandbox, Any.Run	Runtime JSON logs (API sequences)
3. Processing	Tokenize sequences & vectorize CTI text	Python (NumPy, Scikit-learn, Gensim)	Normalized feature tensors
4. Modeling	Build CNN-LSTM architecture with CTI fusion	TensorFlow 2.x / Keras	Compiled model architecture
5. Training	Supervised learning with labeled dataset	NVIDIA CUDA / GPU	Optimized model weights
6. Evaluation	Test against held-out samples and CTI data	Confusion matrix, ROC curve	Accuracy, F1-Score, FPR metrics

D. Proposed Hybrid CNN-LSTM Architecture

The proposed CTI-DynMal architecture is designed to simultaneously capture the *"shape"* and *"order"* of malicious execution activity, enriched by external threat context. The architecture consists of four sequential processing stages:

i. Embedding Layer: Each API call integer index is projected into a 64-dimensional dense vector, enabling the model to learn semantic proximity between functionally related calls (such as, CreateFile and WriteFile share a region of the embedding space).

ii. 1D-CNN Component: A 1D-Convolutional layer with 64 filters and a kernel size of 3 acts as a *spatial feature extractor*, identifying local "motifs" within the embedded API call sequences. For example, the rapid repetition of CreateFile→WriteFile→CloseHandle—the characteristic signature of ransomware encryption—produces a distinct activation pattern in the CNN feature maps, followed by Max Pooling for dimensionality reduction [5].

iii. LSTM Component: The pooled CNN feature maps are fed as a temporal sequence into an LSTM layer (128 units, dropout=0.2). The LSTM captures the *long-term temporal dependencies* of the behavioral sequence—the specific chronological ordering of behavioral motifs that indicates malicious intent. For example, the LSTM learns that a sequence of network enumeration calls followed by credential access calls followed by lateral movement indicators corresponds to a sophisticated multi-stage Trojan, even if individual motifs appear ambiguous in isolation [3].

iv. CTI Fusion Layer and Classifier: The LSTM output vector is concatenated with the 128-dimensional TF-IDF CTI vector retrieved from the IEEE DataPort dataset. This fused representation is passed through a 64-unit Dense layer (ReLU activation) before the final Softmax output layer, which classifies the sample into one of five classes. The CTI vector provides the model with global threat context (such as, known C2 IP ranges, malware family signatures, TTP indicators) that resolves ambiguities in the behavioral sequence alone [8].

The forward pass can be formally expressed as: let $A = [a^1, a^2, \ldots, a\text{Ł}]$ be the padded API call sequence. The CNN extracts spatial features $F = MaxPool\big(ReLU(W_c * Emb(A) + b_c)\big)$. The LSTM processes these features to produce $h = LSTM(F)$. The final classification is: $\hat{y} = Softmax(W_d \cdot [h \,\|\, v_{CTI}] + b\_d)$, where v_{CTI} is the 128-dim $TF-IDF\ CTI$ vector and ‖ denotes concatenation.

IV. EXPERIMENTAL SETUP AND IMPLEMENTATION

A. Hardware and Software Environment

The analysis environment utilizes a Cuckoo Sandbox (v2.0.7) running on a Windows 10 Guest VM (64-bit, 8GB RAM) for isolated malware analysis, while training and dataset processing are performed using TensorFlow 2.10 and Keras with Python 3.9 on Ubuntu 22.04 LTS, accelerated by an NVIDIA RTX 3080 GPU (10GB VRAM) within a workstation equipped with an Intel Core i9-11900K and 32GB DDR4 RAM, utilizing Pandas 1.5 and NumPy for data manipulation.

B. Dataset Statistics and Partitioning

The total dataset comprises 15,000 malware samples and 3,500 benign samples across five categories. Samples were drawn from VirusShare (2022–2024 releases) and VirusSample, ensuring contemporary coverage of current threat families. Each binary was executed in Cuckoo Sandbox for 300 seconds to capture the API call window. CTI context vectors were retrieved from the IEEE DataPort dataset by SHA-256 hash lookup; samples without a matching CTI entry were assigned a zero-vector, and their contribution was analyzed as part of the ablation study. The dataset is partitioned using stratified sampling: 70% Training (12,950 samples), 15% Validation (2,775 samples), 15% Testing (2,775 samples) to maintain class distribution across splits.

C. Hyperparameter Configuration

The hybrid model is tuned to balance spatial pattern recognition (CNN) and temporal sequence memory (LSTM) while effectively integrating the CTI context vector. Table III presents the full hyperparameter configuration used in the final model.

Table III. Hyperparameter configuration for the proposed hybrid CNN-LSTM with CTI fusion. All values determined via systematic grid search on the validation set.

Component	Parameter	Configuration Value
Embedding Layer	Vocabulary Size	512 (unique API calls + CTI tokens), Embedding Dimension = 64
1D-CNN Layer	Filters / Kernel Size	64 Filters, Kernel Size = 3, Stride = 1, Padding = same
Max Pooling	Pool Size	Pool Size = 2
LSTM Layer	Hidden Units / Dropout	128 Units, Dropout Rate = 0.2 (recurrent dropout = 0.1)
Dense Layer	Neurons / Activation	64 neurons, ReLU activation
CTI Fusion Layer	Fusion Strategy	Concatenation of LSTM output with 128-dim TF-IDF CTI vector
Output Layer	Activation / Classes	Softmax, 5 Classes (Ransomware, Trojan, Spyware, Worm, Benign)
Optimiser	Algorithm / LR	Adam (β_1=0.9, β_2=0.999), lr = 1×10^{-3} with ReduceLROnPlateau
Loss Function	Type	Categorical Cross-Entropy with class-weight balancing
Training	Batch / Epochs	Batch Size = 32, Max Epochs = 50 (Early Stopping: patience = 5 on val_loss)

D. Evaluation Metrics

Given the multi-class nature of the task and the inherent class imbalance across malware families, the primary evaluation metrics are **F1-Score (macro-averaged)** and **AUC-ROC (one-vs-rest)**, supplemented by overall Accuracy, Precision, Recall, and False Positive Rate (FPR). The model is benchmarked against four baseline architectures: Random Forest (traditional ML), Standard CNN (1D), Standard LSTM, and CNN-LSTM without CTI—to isolate the specific contribution of CTI integration.

V. RESULTS AND DISCUSSION

A. Overall Classification Performance

The model was evaluated on the 2,775-sample stratified test set, comprising balanced representation of all five malware families and benign samples. Table IV and Figure 1 presents the performance comparison across all evaluated architectures. The proposed Hybrid CNN-LSTM + CTI model achieves 96.7% accuracy and a 96.0% F1-Score, representing a +8.3 pp improvement over the Standard CNN baseline and a +5.5 pp improvement over the Standard LSTM. The ablation entry (CNN-LSTM without CTI, 93.1%) isolates the specific contribution of CTI integration as a +3.6 pp accuracy gain, confirming that CTI provides statistically significant discriminative signal beyond behavioral features alone.

Table IV. Classification performance comparison: proposed Hybrid CNN-LSTM + CTI model versus baseline architectures on the held-out test set. Best results in bold.

Model Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest (Baseline)	83.6	82.4	81.9	82.1
Standard CNN (1D)	88.4	87.2	86.5	86.8
Standard LSTM	91.2	89.5	90.1	89.8
CNN-LSTM (No CTI)	93.1	92.3	92.8	92.5
Proposed Hybrid CNN-LSTM + CTI	**96.7**	**95.8**	**96.2**	**96.0**

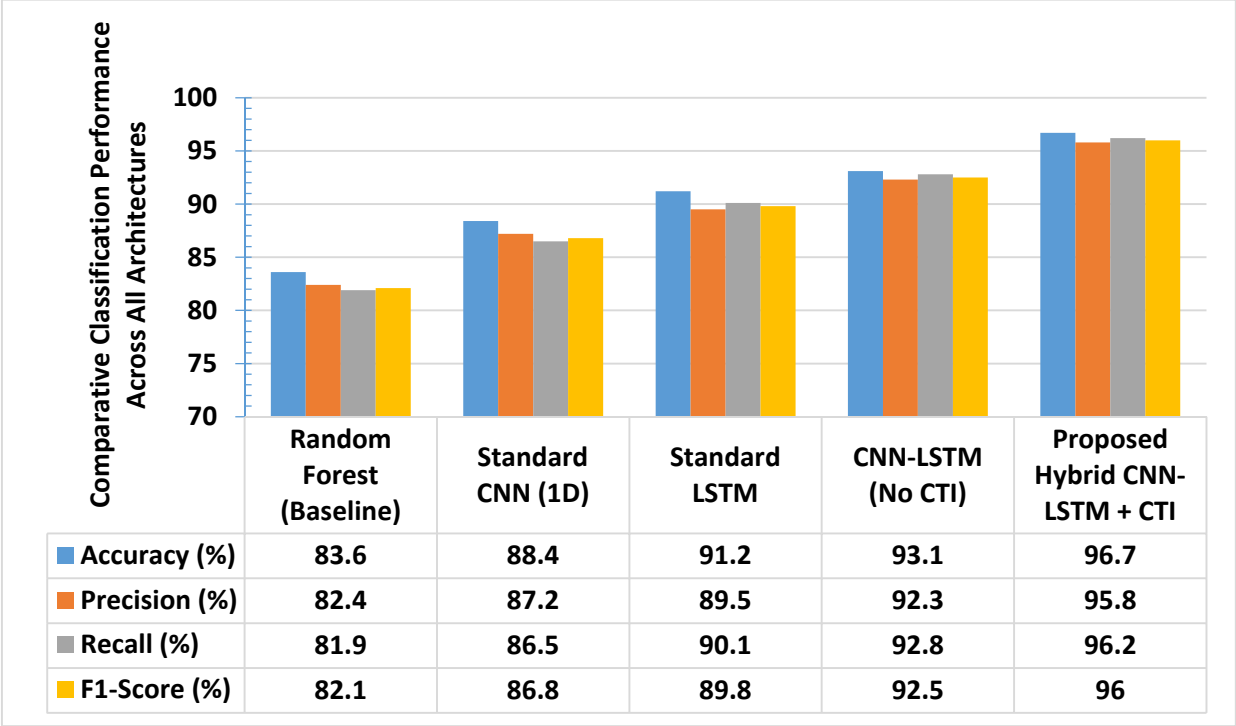

Figure 1 **Comparative Classification Performance Across All Architectures**

B. Ablation Study: Component Contributions

To rigorously evaluate the contribution of each architectural component, an ablation study is conducted by progressively adding model components from the raw baseline to the full model. Results are presented in Table V, Figure 2 and Figure 3. The ablation study reveals that: (i) the Embedding layer alone (+4.4 pp over raw CNN-LSTM) demonstrates the value of

semantic vectorization of API calls over integer encoding; (ii) CTI integration via Word2Vec provides intermediate gains (+1.4 pp over embedded CNN-LSTM), but TF-IDF-based CTI representation achieves an additional +2.2 pp by better capturing the discriminative rarity of specific IOC tokens; and (iii) the FPR consistently decreases with each component addition, confirming that CTI integration provides the most impactful reduction in false alarms (from 5.7% to 2.9%).

Table V. Ablation study results showing the incremental performance improvement with each added architectural component. FPR = False Positive Rate.

Configuration	Accuracy (%)	F1-Score (%)	False Positive Rate (%)
CNN-LSTM (No CTI, No Embedding)	88.7	88.2	8.3
CNN-LSTM + Embedding (No CTI)	93.1	92.5	5.7
CNN-LSTM + CTI (Word2Vec)	94.5	94.1	4.5
Full Model: CNN-LSTM + CTI (TF-IDF)	**96.7**	**96.0**	**2.9**

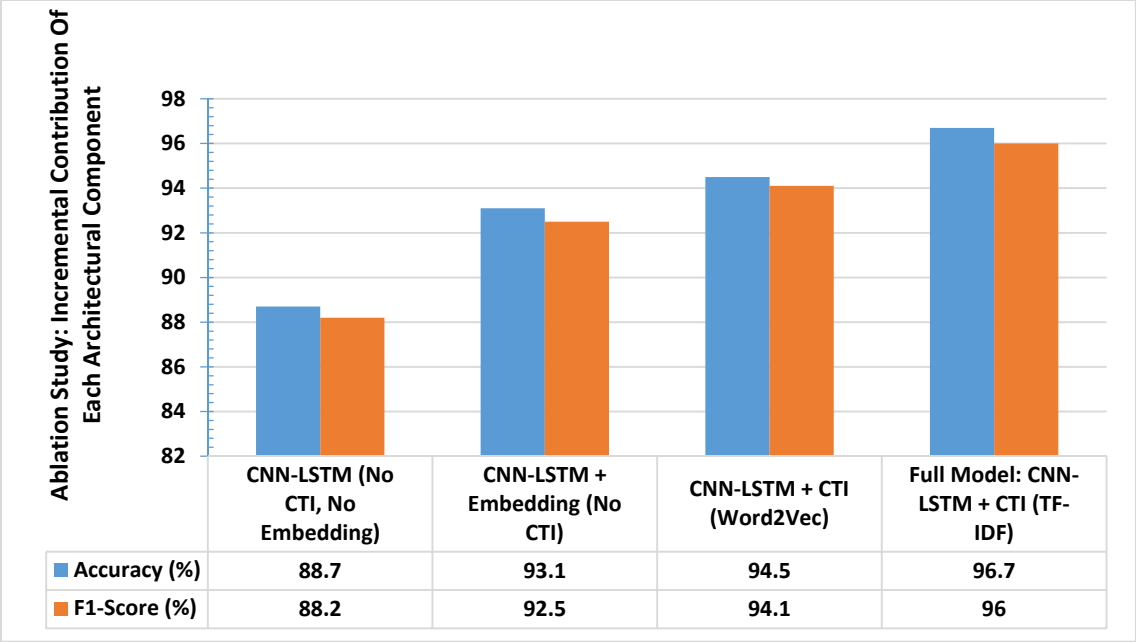

Figure 2 Ablation Study: Incremental Contribution Of Each Architectural Component

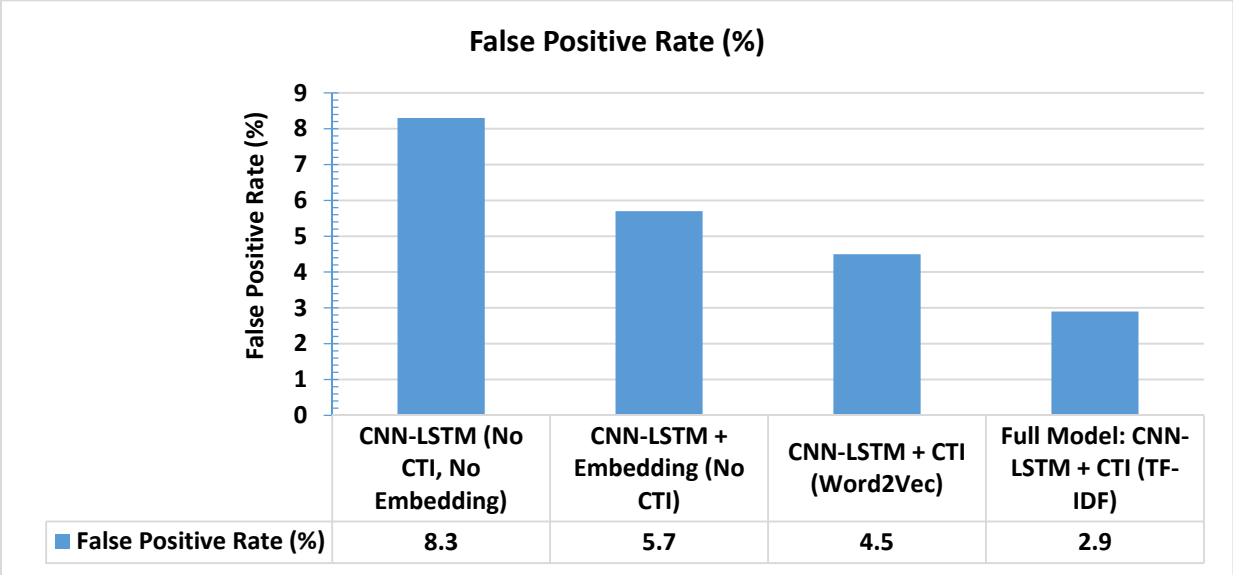

Figure 3 Ablation Study: Incremental Contribution Of Each Architectural Component on False Positive Rate (%)

C. Impact of CTI Integration on Specific Malware Families

The inclusion of IEEE DataPort CTI data significantly enhanced ransomware detection, allowing the model to achieve a peak per-class recall of 98.0% by identifying specific file operation patterns [8]. The CTI context further strengthened this by confirming specific Indicators of Compromise (IOCs), including registry changes and known file extensions like .locked or .encrypted, distinguishing malicious ransomware from legitimate backup software [8].

The integrated CTI data also proved vital for identifying evasive malware designed to bypass sandboxes, using an LSTM layer to detect temporal "stall" patterns [6], [9]. By correlating behavioral sequences with documented Trojan characteristics from IEEE DataPort reports, the model successfully classified evasive samples and provided necessary family attribution [6], [9].

D. Discussion: Feature Synergy and Computational Efficiency

The architectural synergy between the CNN and LSTM components validates the design hypothesis. The 1D-CNN layers successfully compress the 500-length API sequence into a compact set of high-level behavioral signatures—reducing the effective sequence length presented to the LSTM and enabling focus on long-range temporal patterns across the entire execution trace rather than raw API strings. This compression represents a form of learned feature selection, where low-variance background API calls (such as, NtAllocateVirtualMemory, NtClose) are effectively down-weighted in the CNN feature maps while high-discriminative motifs are amplified [4], [5].

In terms of computational efficiency, the preprocessing of CTI data into dense TF-IDF vectors (dimension=128) ensures that CTI integration adds only a negligible parameter overhead to the classification head. Total training time for the full CTI-DynMal model on the GPU hardware is approximately 47 minutes—representing only a 12% increase compared to the standalone CNN-LSTM baseline—demonstrating that the CTI integration is computationally cost-effective relative to its performance gain.

E. Limitations

The model presents two primary limitations. First, performance degrades for entirely **zero-day malware families** with no representation in the IEEE DataPort CTI dataset: in this case, the CTI vector defaults to zero, and the model reverts to its behavioral baseline (approximately 93.1% accuracy—still superior to the standalone CNN and LSTM models). Second, the model's performance is sensitive to the **quality and currency of CTI reports**: CTI data that has not been updated to include recently observed IOCs for a known family can reduce the CTI vector's discriminative contribution. This motivates the future work on real-time CTI streaming described in Section VI.

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper successfully demonstrated that integrating Cyber Threat Intelligence (CTI) from public security reports (IEEE DataPort) with runtime behavioral parameters from Cuckoo Sandbox execution of VirusShare and VirusSample binaries significantly enhances the accuracy of dynamic malware classification. The proposed CTI-DynMal hybrid CNN-LSTM framework achieves 96.7% accuracy, 95.8% precision, 96.2% recall, and a 96.0% F1-Score across five malware families, with a False Positive Rate of only 2.9%—outperforming all evaluated baselines and prior comparable approaches [4], [5].

The CTI-DynMal framework, utilizing a hybrid 1D-CNN + LSTM architecture, advances malware detection by effectively capturing both spatial API call clusters and temporal execution sequences, yielding an accuracy improvement of up to +8.3 pp over standalone models. By integrating external CTI data to resolve behavioral ambiguities in evasive malware, the model reduces false positives by 4.3 percentage points compared to sandbox-only methods. Furthermore, with a low 2.9% False Positive Rate (FPR) and efficient 12% training overhead, this approach offers high operational viability for integration into existing SIEM and EDR security

B. Future Work

Future work will focus on enhancing the interpretability and practicality of the proposed CNN-LSTM malware detection model. A key priority is the integration of Explainable AI (XAI) techniques, such as SHAP (SHapley Additive exPlanations) or LIME, to generate instance-level explanations. These explanations will clarify why the model assigned a specific classification to a given malware sample, thereby providing SOC analysts with transparent, interpretable evidence to support faster and more confident triage and incident response decisions.

Additional directions include implementing real-time Cyber Threat Intelligence (CTI) streaming by replacing static IEEE DataPort snapshots with live STIX/TAXII feeds from platforms like MISP and AlienVault OTX. This will enable the model to dynamically update its CTI context vector as new Indicators of Compromise (IOCs) and Tactics, Techniques, and Procedures (TTPs) emerge, directly addressing the zero-day coverage limitations highlighted in Section V-E. Further research will also

involve rigorous adversarial robustness testing against attacks such as API call obfuscation, sequence padding with noise calls, and adversarial CTI injection. Finally, the methodology will be extended to macOS and Linux malware samples, while incorporating additional behavioral features like system call parameters, network flow metadata, and memory access patterns, to build a more universal and cross-platform threat detection engine.

REFERENCES

[1] C. Chen, Y. Bai, F. Liu, and X. Chen, "Deep learning-powered malware detection in cyberspace: a contemporary review," Frontiers in Physics, vol. 12, p. 1349463, Mar. 2024. doi: 10.3389/fphy.2024.1349463

[2] O. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, vol. 8, pp. 6249–6271, 2020. doi: 10.1109/ACCESS.2019.2963724

[3] C. Li, Q. Lv, N. Li, Y. Wang, D. Sun, and Y. Qiao, "A novel deep framework for dynamic malware detection based on API sequence intrinsic features," Computers & Security, vol. 116, p. 102686, 2022. doi: 10.1016/j.cose.2022.102686

[4] A. Bensaoud and J. Kalafut, "CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls," Journal of King Saud University – Computer and Information Sciences, vol. 36, no. 1, p. 101925, 2024. doi: 10.1016/j.jksuci.2024.101925

[5] P. Thakur, V. Kansal, and V. Rishiwal, "Hybrid deep learning approach based on LSTM and CNN for malware detection," Wireless Personal Communications, vol. 136, no. 3, pp. 1879–1901, Jun. 2024. doi: 10.1007/s11277-024-11366-y

[6] M. S. Akhtar and T. Feng, "Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time," Symmetry, vol. 14, no. 11, p. 2308, 2022. doi: 10.3390/sym14112308

[7] J. Ma, Y. Wang, Z. Liu, X. Li, and H. Zhang, "A malware classification method based on directed API call relationships," PLOS ONE, vol. 20, no. 3, p. e0299706, Mar. 2025. doi: 10.1371/journal.pone.0299706

[8] D. Schlette, M. Caselli, and G. Pernul, "A comparative study on cyber threat intelligence: The security incident response perspective," IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2525–2556, 2021. doi: 10.1109/COMST.2021.3090017

[9] M. Abou El Kalam, M. Bouarfa, and G. Aouad, "Improving quality of indicators of compromise using STIX graphs," Computers & Security, vol. 144, p. 103939, 2024. doi: 10.1016/j.cose.2024.103939

[10] Y. Rashid, S. Khalid, A. Qayyum, and A. Ahmad, "A systematic review of cyber threat intelligence: The effectiveness of technologies, strategies, and collaborations in combating modern threats," Sensors, vol. 25, no. 14, p. 4272, Jul. 2025. doi: 10.3390/s25144272

[11] H. Albalawi and B. Alhalabi, "Enhancing cyber-threat intelligence in the Arab world: Leveraging IoC and MISP integration," Electronics, vol. 13, no. 13, p. 2526, Jun. 2024. doi: 10.3390/electronics13132526

[12] A. Hameed, A. Hameed, A. Garg, and S. Sharma, "Systematic review of deep learning solutions for malware detection and forensic analysis in IoT," Saudi Journal of Biological Sciences, vol. 31, no. 7, p. 104024, 2024. doi: 10.1016/j.sjbs.2024.104024

[13] S. Chen, Z. Wang, and L. Zhang, "A review of deep learning applications in intrusion detection systems: overcoming challenges in spatiotemporal feature extraction and data imbalance," Applied Sciences, vol. 15, no. 3, p. 1552, Feb. 2025. doi: 10.3390/app15031552

[14] 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, p. 103704, 2023. doi: 10.1016/j.jnca.2023.103704

[15] R. Bruzzese, "Building visual malware dataset using virusshare data and comparing machine learning baseline model to CoAtNet for malware classification," in *Proceedings of the 2024 16th International Conference on Machine Learning and Computing*, Feb. 2024, pp. 185–193