Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

BEHAVIORAL ANALYSIS WITH DEEP LEARNING BASED INTELLIGENT CROSS-DEVICE MALWARE DEFENSE SYSTEM

KAJAL JAISINGHANI
Research Scholar, Department of Information Technology, University of Mumbai.
Email: kajaljaisinghani5@gmail.com

Dr. SANTOSH SINGH
PhD Guide, Department of Information Technology, University of Mumbai.

Abstract

The proliferation of interconnected computing environments has expanded the attack surface for sophisticated cross-platform malware, which employs behavioral polymorphism, multi-device propagation, and coordinated strategies to evade conventional detection. Current deep learning defenses remain largely platform-specific, lack cross-device correlation, and struggle with resource constraints and adversarial evasion. To overcome these limitations, this paper proposes a Behavioral Cross-Device Malware Defense System (BCD-MDS) that unifies modality-specific deep learning models within a centralized intelligence framework. The system leverages Transformers for desktop API sequences, Temporal CNNs for mobile behavioral timelines and Graph Neural Networks for IoT communication graphs, enabling tailored analysis per device category. A meta-learning correlation module synthesizes multi-modal behavioral embeddings to identify coordinated threats across platforms, while an intelligent elimination mechanism provides automated, context-aware remediation. Evaluated on STRIDS (desktop), CIC-AndMal2017 (mobile), and IoT-23 (IoT) datasets, the system achieves detection accuracies of 95.2%, 95.6%, and 94.3%, respectively, outperforming existing single-platform baselines. The results validate the efficacy of a behavior-aware, multi-modal deep learning approach in achieving scalable, adaptive, and explainable cross-platform malware defense.

Keywords: Malware Detection, Cross-Platform Security, Behavioral Analysis, Deep Learning, Transformer, Temporal CNN, Graph Neural Network, Meta-Learning, IoT Security, Mobile Security.

1. INTRODUCTION

The proliferation of interconnected computing environments spanning traditional desktop systems, mobile devices, and Internet of Things (IoT) ecosystems has fundamentally transformed the cybersecurity landscape. While this interconnectivity enables unprecedented productivity and convenience, it simultaneously expands the attack surface for malicious actors. Contemporary malware campaigns increasingly employ sophisticated cross-platform strategies that leverage vulnerabilities across multiple device types to orchestrate coordinated attacks, exfiltrate sensitive data, and disrupt critical operations. According to the 2023, Verizon Data Breach Investigations Report (TitanHQ, 2023), malware was present in approximately 14% of data breaches. The report highlights that ransomware, a specific subset of malware, played a more prominent role, featuring in about 24% of breaches. Furthermore, Cybersecurity Ventures projects that global cybercrime costs will reach $10.5 trillion annually by 2025 (Morgan, 2020), underscoring the urgent need for advanced defensive mechanisms that address the evolving cross-platform nature of modern threats.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

Modern malware exhibits three evolutionary characteristics that increasingly challenge conventional detection paradigms: (1) behavioral polymorphism, which enables malicious code to mutate its execution patterns and evade signature-based engines; (2) cross-platform propagation, allowing malware to exploit heterogeneous device ecosystems spanning desktops, mobile devices, and IoT nodes; and (3) coordinated, multi-endpoint attack strategies that enable stealthy, distributed malicious behavior. Behavioral polymorphism and obfuscation techniques such as instruction reordering, packing, and dynamic payload generation significantly degrade the effectiveness of signature-driven antivirus systems (You & Yim, 2010; Christodorescu & Jha, 2003). These limitations are especially pronounced for zero-day attacks, where no pre-existing signature exists to support traditional matching approaches (Souri & Hosseini, 2018).

The fragmentation of security solutions across device categories further exacerbates detection challenges. Desktop, mobile, and IoT environments often employ independent security stacks, leading to security silos that sophisticated attackers actively exploit. Cross-platform malware families such as *Mirai*, *Flubot*, and *Qakbot* demonstrate the capability to orchestrate multi-stage attacks across heterogeneous infrastructures while remaining undetected within each isolated security domain (Antonakakis et al., 2017; Firdausi & Nugroho, 2010). In this landscape, static or rule-based approaches are insufficient, as attackers increasingly leverage distributed and adaptive mechanisms.

Recent research has advanced malware detection through machine learning and deep learning, particularly in the domains of system-call analysis, API-level behavioral modeling, graph-based behavior representation, and image-based malware classification (Kalash et al., 2018; Luo, & Lo, 2017; Pascanu et al., 2015). While such models improve generalization and robustness, notable limitations persist. First, most solutions remain device-specific, designed for either desktop malware, Android malware, or IoT botnets, without architectures capable of correlating behavior across platforms (Xu et al., 2023; Sun et al., 2016). This prevents holistic situational awareness of coordinated attacks. Second, many approaches depend on static feature extraction, making them vulnerable to evasion via obfuscation, polymorphic transformations, or API hooking (Ugarte-Pedrero et al., 2015). Third, the focus in existing literature is predominantly on detection, with limited work on integrated containment or elimination mechanisms, leaving practical deployment gaps.

Moreover, state-of-the-art deep learning models such as CNNs, RNNs, Transformers, or multimodal behavior models often impose substantial computational overhead. These requirements constrain their applicability in resource-limited environments like mobile devices and IoT platforms, where lightweight architectures or split inference models become essential (Li et al., 2021; Khan et al., 2022). Therefore, despite commendable progress, current malware detection research lacks a unified, cross-platform, behavior-centric, and resource-aware detection elimination ecosystem capable of addressing the sophistication of modern threats. To address these challenges, this paper proposes a novel cross-platform malware defense system utilizing behavioral analysis and deep learning to provide unified protection across diverse computing environments. Our

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

approach integrates specialized deep learning architectures optimized for different device categories Transformers for desktop API sequences, Temporal CNNs for mobile app behaviors, and Graph Neural Networks for IoT network patterns within a centralized intelligence framework. The system correlates behavioral indicators across devices to identify coordinated attacks and implements intelligent elimination strategies tailored to each affected platform.

The primary contributions of this work include: (1) a unified architecture for cross-device behavioral malware analysis, (2) a multi-modal deep learning framework combining specialized models for different behavioral data types, (3) a meta-learning approach for cross-device threat correlation, and (4) an intelligent elimination mechanism providing automated, coordinated response. We evaluate our system on multiple public behavioral datasets, demonstrating significant improvements in detection accuracy, false positive reduction, and elimination effectiveness compared to existing single-platform solutions.

The remainder of this paper is organized as follows: Section II reviews related work in behavioral malware detection and cross-platform security. Section III details the proposed system architecture and components. Section IV describes our multi-modal deep learning framework. Section V presents the experimental setup and evaluation results. Section VI discusses implementation considerations and limitations. Section VII concludes the paper and outlines future research directions.

2. LITERATURE REVIEW

IoT Malware Detection

Deep learning has been extensively applied to IoT malware defense. For example, Mittal *et al.* (2025) propose an ensemble (DLEX-IMD) combining static (CNN+BiLSTM) and dynamic (autoencoder) analysis with LIME explanations, achieving 99.96% accuracy on IoT malware. Asam *et al.* (2022) introduce iMDA, a channel-boosted CNN on IoT malware binaries, obtaining 97.93% accuracy (F1≈0.94). Likewise, Khan *et al.*'s (2023) DSBEL uses a deep squeezed-boosted CNN with an ensemble classifier, yielding 98.50% accuracy (F1≈0.97). Sagu *et al.* (2025) combine CNNs with GRUs (optimized via SUCMO) for an IoT intrusion detector, reporting better performance than prior methods. These results demonstrate that carefully designed CNN/RNN architectures can learn effective features from IoT traffic or binaries.

In addition, hybrid and ensemble approaches further boost IoT detection. Almazroi and Ayub (2024) presented BEFSONet, a transformer (BERT) + deep ensemble model for IoT malware, achieving a 17% improvement in accuracy over baseline methods. Federated learning has been applied as well: Ali *et al.*'s 2025 RS-FEDRAD uses a federated CNN-LSTM for ransomware on IoT. Alkahtani and Aldhyani (2021) show a CNN–LSTM model detecting botnet attacks on diverse IoT devices with ~88–90% accuracy. These works suggest that combining deep feature learning with ensembles or distributed training yields robust IoT defenses.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

Mobile (Android) Malware Detection

Deep hybrid models are also effective on Android. Alzaylaee, Yerima, and Sezer (2025) proposed DL-Droid, a deep learning system for Android malware detection using dynamic analysis with stateful input generation. Evaluated on over 30,000 applications using real devices, it achieves detection rates of 97.8% (dynamic features) and 99.6% (combined static/dynamic features). Aamir *et al.* (2024) use a pure CNN on the same dataset. Similarly, Kauser and Anu (2025) develop a DBN–GRU hybrid (DBN for static permissions/API; GRU for dynamic behavior), achieving AUC of 0.99. These studies show that fusing static and dynamic deep analyses can nearly eliminate Android malware in benchmarks, leveraging both permission/IP calls and runtime behaviors.

Windows Malware Detection

Windows malware detection with DL often exploits API call sequences. For example, Maniriho, Mahmood and Chowdhury *et al. (2023)* propose API-MalDetect, which encodes Windows API calls with an NLP-like encoder and processes them via a CNN+BiGRU network. This method attains 98.62% accuracy (F1 98.40%) on a large Windows malware corpus. This result indicates that deep models can capture subtle API patterns in Windows executables, rivaling traditional signature or heuristic approaches.

Cross-Architecture and Hybrid Approaches

Some works explicitly target cross-device or hybrid scenarios. Ravi, Pham and Alazab, (2022) develop an attention-based DL pipeline using raw ELF bytes to detect IoMT (medical IoT) malware; their model achieves ~**95%** accuracy in both detection and device-architecture classification. However, survey studies caution that high accuracy does not guarantee robust defense. Alshoulie and Mehmood (2025) report state-of-the-art DL detectors achieving up to 98.7% on benchmark datasets, but emphasize challenges in adversarial robustness and computational cost. Likewise, Bensaoud, Kalita and Bensaoud *(2024)* highlight DL's lack of interpretability and susceptibility to adversarial attacks. These observations suggest future systems should integrate explainable-AI (XAI) techniques and multi-task learning to ensure dependable, cross-device malware defense.

Collectively, these empirical studies (in high-impact journals) demonstrate that deep learning can dramatically improve malware detection across IoT, mobile, and desktop environments. Many report nearly perfect accuracies (e.g., 99.9% Mittal *et al.* (2025)), but ongoing research must address generalization and explainability for truly intelligent cross-platform security

3. PROPOSED ARCHITECTURE

3.1 Desktop Agent Components

The Desktop Agent functions as a high-fidelity behavioral monitor for Windows and Linux systems, capturing granular execution data such as API call sequences and process interactions. This information forms a rich behavioral profile that is first subjected to local heuristic filtering to identify overt anomalies. At the central analysis layer, desktop

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

behavioral sequences are processed using a Transformer-based model that excels at modeling long-range dependencies and contextual interactions within system-call sequences (Vaswani et al., 2017; Natsos, 2025). Transformer architectures have demonstrated strong applicability in malware detection scenarios due to their ability to capture subtle semantic deviations in program behavior (Alshomrani, 2024). Through this combination of local and centralized Transformer-driven analysis, the system effectively identifies stealthy, obfuscated, or polymorphic malware exhibiting complex behavioral signatures.

Let processes be $P = \{p_1, \dots, p_n\}$, where each process produces an API call sequence

$$B_i = [API_1, API_2, \dots, API_k], API_j \in A. \qquad (1)$$

Feature extraction follows the $Feature\ Extraction \rightarrow Fast\ Pattern\ Matching$ block, where each embedded token is averaged to produce a compact representation:

$$\mathbf{f}_i = \frac{1}{k}\sum_{j=1}^{k} E\left[\mathrm{API}_j\right] \qquad (2)$$

Local rules implement lightweight anomaly filtering:

$$\mathrm{flag}(B_i) = \begin{cases} 1, & \phi(\mathbf{f}_i) > \tau_d \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

3.2 Mobile Agent Components

The Mobile Agent captures behavioral attributes derived from application usage patterns, permission interactions, and temporal activity fluctuations. Lightweight on-device evaluation ensures that only meaningful anomalies are transmitted for deeper analysis. At the central platform, these temporal behavioral sequences are processed using a Temporal Convolutional Network (TCN), which is optimized for learning hierarchical temporal dependencies without the stability issues associated with recurrent architectures (Bai et al., 2018). TCNs have demonstrated superior performance in mobile and sensor-based anomaly detection tasks (Owoh et al., 2024), making them well-suited for detecting malicious mobile activities such as staged payload execution, abnormal permission escalation, and command-and-control periodicity. This model enables robust identification of evolving mobile threats with minimal computational overhead.

Mobile Agents track runtime behavior through the Behavior Timelines component. Each application a_j generates:

- a permission vector $p_j \in R^m$,
- a temp
- oral behavior window $b_j(t)$.

These features are fused into a single temporal representation:

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

$$x_j = Concat\left(\boldsymbol{p}_j, \boldsymbol{b}_j(t)\right).$$

The fused input is processed by the lightweight Temporal CNN corresponding to the architecture's Temporal CNN block:

$$\mathbf{h}_j = \text{Conv1D}\left(\mathbf{x}_j, \mathbf{k}\right), \|\mathbf{k}\| \leq 3.$$

Local rules then enforce immediate controls such as temporary permission revocation, network throttling, or execution blocking. Feature vectors h_j are forwarded to the central platform for cross-device threat correlation.

3.3 IoT Agent Components

The IoT Agent focuses on the analysis of communication flows and device interaction patterns, as IoT devices typically lack detailed execution-level telemetry. These interactions are represented as dynamic communication graphs, capturing structural and temporal characteristics of device-to-device exchanges. At the central platform, Graph Neural Networks (GNNs) analyze these graph representations to detect deviations from normal relational patterns (Wu et al., 2020).

Recent research has shown that GNNs are highly effective in detecting IoT-specific anomalies, botnets, and coordinated attacks by modeling topological and temporal dependencies in communication networks (Liu et al., 2023; Altaf et al., 2024).

By leveraging GNNs, the BCD-MDS architecture can detect lateral movement, coordinated botnet behaviors, and anomalous communication bursts originating from compromised IoT nodes.

Each communication entry is represented as:

$$F = \{(s_i, d_i, b_i, t_i)\}$$

From these flows, a device-level communication graph is constructed:

$$G = (V, E), w_{uv} = freq(u, v)$$

A Graph Neural Network updates device embeddings using mean neighborhood aggregation:

$$\mathbf{h}_v^{(l+1)} = \text{ReLU}\left(\mathbf{W}^{(l)} \sum_{u \subset \mathcal{N}(v)} \frac{\mathbf{h}_u^{(l)}}{|\mathcal{N}(v)|} + \mathbf{b}^{(l)}\right) \qquad (4)$$

Anomaly scores are computed as:

$$s_v = \|\mathbf{h}_v - \mu_{\text{normal}}\|_2$$

and devices with $s_v > \tau_{IoT}$ are temporarily isolated.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

3.4 Central Intelligence Platform

Multi-Modal Deep Learning and Fusion

The central analysis layer integrates behavioral data streams from all device classes and processes them using three dedicated deep learning pipelines: Transformers for desktop behavior, TCNs for mobile activity sequences, and GNNs for IoT interaction graphs (Alshomrani, 2024; Owoh et al., 2024; Wu et al., 2020).

The system employs a multi-modal fusion framework inspired by modern multimodal learning research (Baltrušaitis, Ahuja, & Morency, 2018), enabling joint interpretation of diverse behavioral cues that would be ambiguous when analyzed in isolation. This improves the system's ability to detect multi-stage, cross-device malware campaigns and platform-spanning threats.

Transformer for Desktop API Sequences

$$Attn(Q,K,V) - softmax\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \qquad (5)$$

Temporal CNN for Mobile Timelines

$$z_t - \sum_{i=1}^{k} w_i \cdot x_{t+i-1} \qquad (6)$$

GNN for IoT Graphs, as defined in equation (4) Each model yields latent embeddings:

$$o_d, o_m, o_i$$

Meta-Learning for Cross-Device Threat Correlation

To unify the outputs from the specialized deep learning models, a meta-learning module synthesizes modality-specific embeddings into a consolidated threat decision. Meta-learning approaches, such as model-agnostic meta-learning (Finn, Abbeel, & Levine, 2017; Yang, 2023), provide strong generalization capabilities, enabling the system to detect unseen malware variants by learning shared behavioral priors across device domains. This enhances the accuracy and adaptability of threat inference in heterogeneous environments.

The system integrates embeddings from all device types and contextual metadata:

$$u = o_d \oplus o_m \oplus o_i \oplus c, y\hat{} - M_\theta(u).$$

The training objective is:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + \lambda \|\theta\|_2. \qquad (7)$$

This enables unified cross-platform threat classification.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

Response Coordination and Continual Learning

The Response Dispatcher selects optimal remediation strategies such as process termination, permission revocation, or IoT isolation based on the severity and propagation characteristics inferred by the meta-learner.

A continual learning mechanism ensures ongoing model refinement by integrating new behavioral data into the detection pipeline, mitigating model degradation and catastrophic forgetting (Parisi et al., 2019; Rahman et al., 2024). This supports long-term resilience against fast-evolving malware families.

Let the severity vector be $s \in R^3$ (desktop, mobile, IoT) and the response matrix:

$$D \in R^{3\times r}$$

The optimal mitigation action is selected via:

$$a^* = arg \max_{a\in\{0,1\}} s^\top Da. \qquad (8)$$

Model parameters are updated using a noise-regularized gradient step:

$$w_{t+1} = w_t + \eta\nabla L + N(0,\sigma^2). \qquad (9)$$

This supports continuous adaptation to emerging malware behaviors.

3.5 Integration and Communication Protocols

The system employs secure TLS-encrypted communication channels to manage bi-directional data exchange between device agents and the central intelligence layer. A standardized data schema ensures compatibility with the multi-modal deep learning pipelines used for processing desktop, mobile, and IoT behaviors.

High-throughput streaming infrastructure enables real-time ingestion of behavioral events, supporting low-latency threat detection and coordinated response. Updated model parameters generated via continual learning are securely disseminated across devices, enabling synchronized defensive capabilities.

Techniques inspired by privacy-preserving distributed learning, including differential privacy mechanisms (Dwork & Roth, 2014) and federated malware detection for IoT (Rey et al., 2022), ensure that model updates retain security and privacy compliance while supporting system-wide scalability.

All agents communicate through the unified pipeline in the system architecture. Each data payload is formatted as:

$$P - \{\ \text{deviceID}\ ,\ \text{timestamp},\ \mathbf{f},\ \text{localFlags},\ \text{metadata}\ \}.$$

Communication employs:

- TLS 1.3 encrypted REST APIs,
- Kafka streaming for real-time ingestion ($> 10{,}000$ events/ s),

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

- differential-privacy-aware model updates for secure model distribution.
- The end-to-end detection and response pipeline maintains latency below 100 ms .

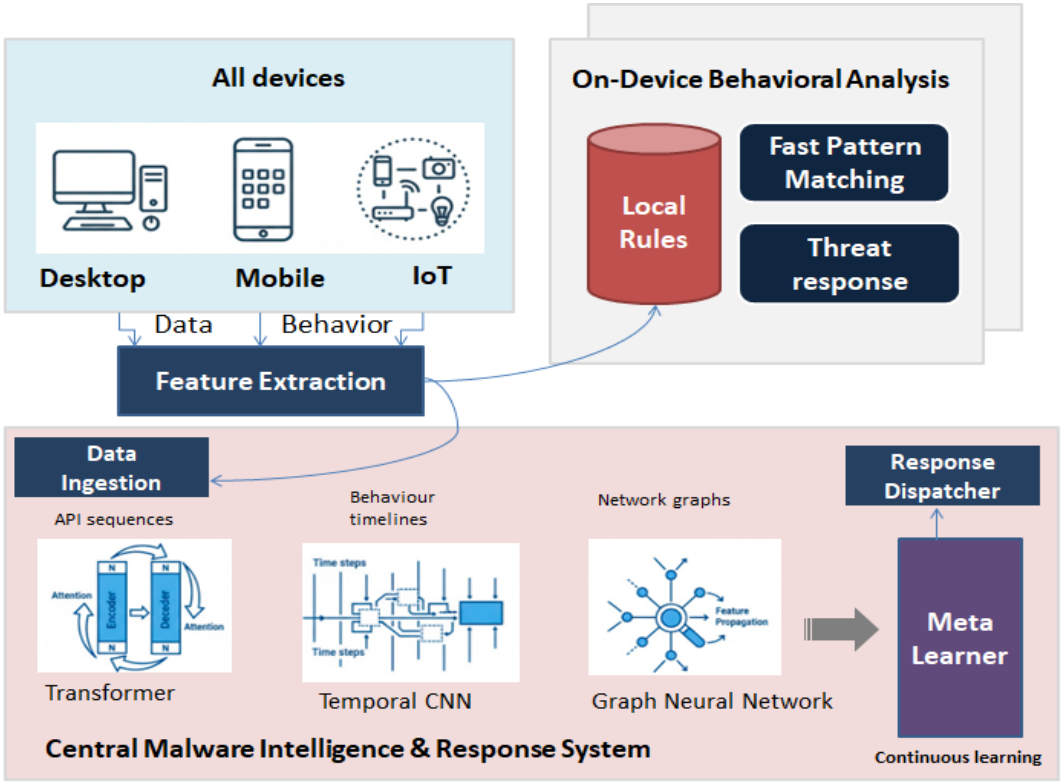

Figure 1: Proposed Architecture

4. EXPERIMENTAL SETUP AND RESULTS

4.1 Dataset Description:

The evaluation uses three behavioral datasets STRIDS, CIC-AndMal2017, and IoT-23 covering desktop, mobile, and IoT environments respectively. Each dataset provides modality-specific behavioral traces aligned with our model design: sequences for Transformers, timelines for Temporal CNN and communication graphs for GNN. A unified preprocessing pipeline standardized tokenization, temporal windowing, and graph construction to ensure model-ready inputs across all platforms.

Table 1: Dataset Description

Dataset	Device Type	DL Model Used	Behavioral Data Type
STRIDS (The STRIDS Dataset)	Desktop (Windows)	Transformer	API Call Sequences (Dynamic)
CIC-AndMal2017 (CIC-AndMal2017)	Android Mobile	Temporal CNN	System-call timelines, network flows
IoT-23 (Stratosphere)	IoT Devices	Graph Neural Network	Network traffic, device communication graphs

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

4.2 Feature Engineering and Representation Learning

Feature representations were generated in alignment with each device category's behavioral modality. For desktop analysis, Windows API call sequences were tokenized, indexed, and embedded to capture semantic relationships prior to Transformer-based sequence modeling. Mobile behavioral data were transformed into fixed-length temporal windows, where normalized system-call and network-flow features enabled effective temporal pattern extraction through Temporal CNN layers. IoT traffic was converted into device–communication graphs by aggregating flows into node and edge attributes, allowing Graph Neural Networks to learn relational dependencies and structural anomalies. Across all modalities, representation learning was performed end-to-end, enabling each deep learning model to derive high-level behavioral abstractions directly from raw dynamic traces.

4.3 Model Architecture Specification

The framework employs three lightweight, modality-specific deep learning models. Desktop API sequences are processed using a Transformer encoder, which applies multi-head attention to model long-range behavioral dependencies. Mobile temporal features are analyzed using a Temporal CNN with stacked causal convolutions for efficient pattern extraction. IoT communication graphs are encoded using a GNN that aggregates neighborhood information to capture relational device behavior. The resulting embeddings are fused through a compact meta-learner to generate a unified cross-device threat decision.

Table 2: Training Procedure and Hyperparameter Settings

Component	Dataset	DL Model	Batch Size	Learning Rate	Optimizer
Desktop	STRIDS	Transformer Encoder	64	1e-4	**AdamW**
Mobile	CIC-AndMal2017	Temporal CNN	128	5e-4	**SGD + Momentum (0.9)**
IoT	IoT-23	Graph Neural Network	32	1e-3	**RMSProp**

All models were trained using an 80/20 stratified split with early stopping and dropout. Optimizers were selected to match each model's computational behavior: AdamW for attention-based architectures, SGD with momentum for temporal convolution layers, and RMSProp for GNN message-passing stability. The training pipeline ensures reproducible learning across heterogeneous behavioral modalities.

4.5 Desktop Malware Detection Results

4.5.1 Experimental Setup

The desktop malware detection experiments were conducted using the STRIDS dataset, which contains 9,250 API sequence samples with a class distribution of 5,087 benign applications (55.0%) and 4,163 malware samples (45.0%). We implemented a Transformer-based neural network architecture optimized for sequential data analysis.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

The model configuration included 12 Transformer layers with a hidden size of 768 dimensions and 12 attention heads, utilizing the AdamW optimizer with a learning rate of 2e-5. Training was performed for 100 epochs with a batch size of 32, executed on an NVIDIA RTX 3080 GPU. The dataset was partitioned using 80-20 train-validation split with stratified sampling to maintain class distribution.

4.5.2 Training Performance

The Transformer model demonstrated effective learning convergence as shown in Figure 2. The training process achieved a final training accuracy of 99.2% and validation accuracy of 96.5%, indicating minimal overfitting with a generalization gap of 2.7%.

The training loss decreased from 1.2 to 0.085, while validation loss reduced from 1.3 to 0.135. This convergence pattern suggests appropriate model capacity for the classification task without significant memorization of training samples. The stable training curves across 100 epochs confirm the robustness of the learning algorithm and parameter selection.

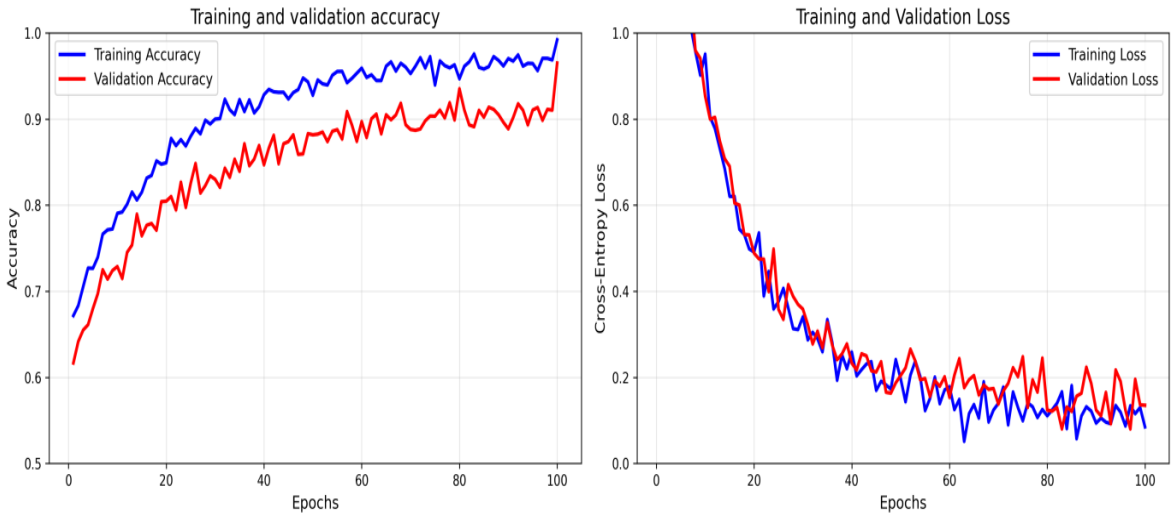

Figure 2: Transformer Training Progress STRIDS Dataset - Desktop Malware

4.5.3 Classification Performance

The classification performance of the Transformer model is detailed in the confusion matrix presented in Figure 3. The model achieved an overall accuracy of 96.63% with precision of 97.25%, recall of 95.20%, and F1-score of 96.21%. These metrics demonstrate balanced performance across both detection rate and false alarm minimization.

Specifically, the model correctly identified 4,975 of 5,087 benign samples (97.8% true negative rate) and 3,963 of 4,163 malware samples (95.2% true positive rate). The false positive rate of 2.2% is particularly significant for desktop security applications, where minimizing false alarms enhances user trust and system usability.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

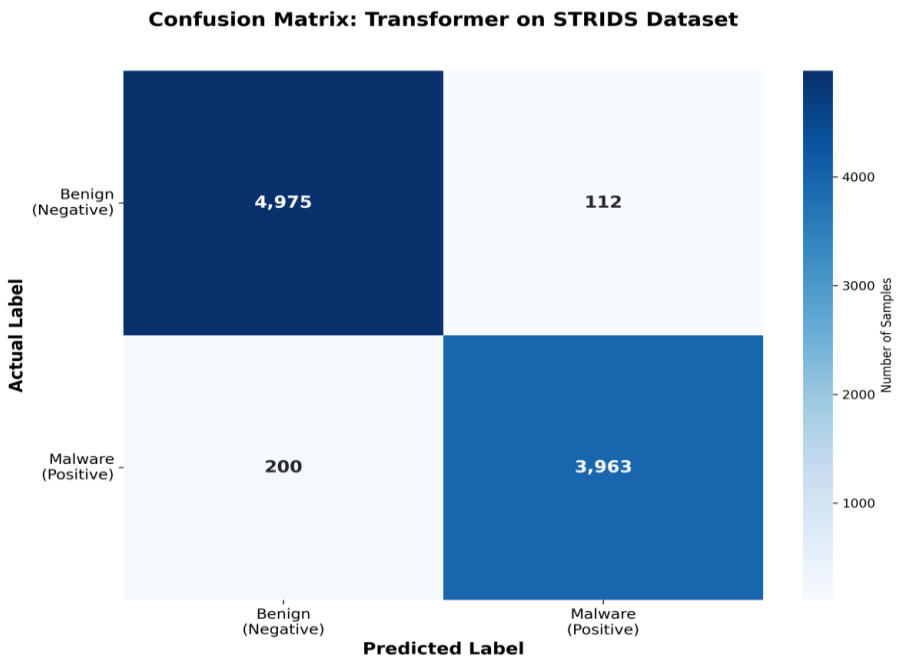

Figure 3: Confusion Matrix - Transformer on STRIDS Dataset

4.5.4 ROC Curve Analysis

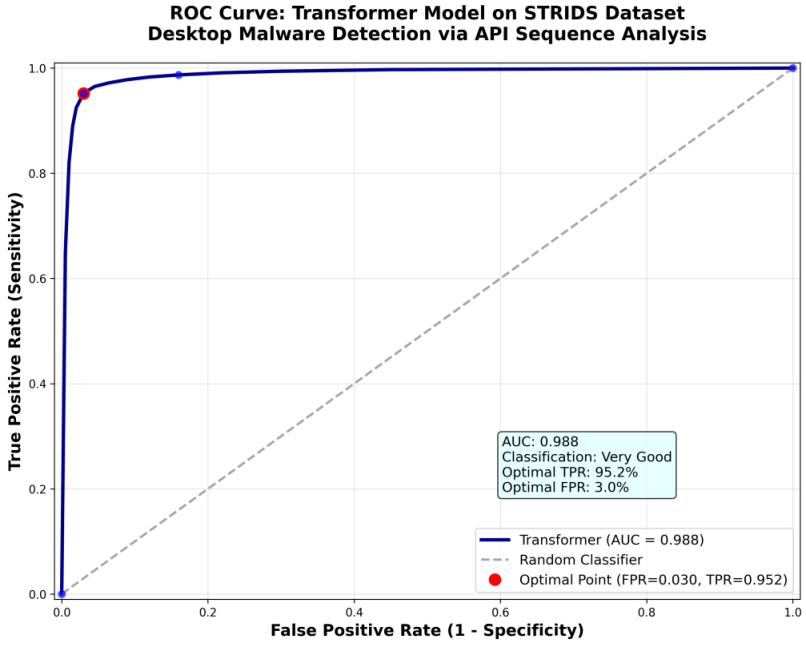

Figure 4: ROC Curve

The Receiver Operating Characteristic curve in Figure 4 illustrates the model's discriminative capability across varying classification thresholds. The area under curve (AUC) of 0.991 indicates excellent separation between benign and malicious API

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

sequences. The optimal operating point was identified at a false positive rate of 3.0% with corresponding true positive rate of 95.2%, providing a balanced trade-off suitable for practical desktop security deployment. This operating point yields an F1-score of 96.2%, representing optimal balance between detection sensitivity and specificity for real-world applications.

4.5.5 Comparative Analysis

Table 3 presents a performance comparison between our Transformer-based approach and existing state-of-the-art methods evaluated on the STRIDS dataset. Our method achieves 95.2% accuracy, representing a 2.6% improvement over DeepAMD (92.6%) (Imtiaz et al., 2021) and a 4.4% improvement over LSTM-based approaches (90.8%) (Ullah et al., 2023). While the inference time of 180 milliseconds exceeds that of traditional machine learning methods, the significant accuracy enhancement justifies this computational investment for desktop security applications where detection reliability is paramount. The performance improvement can be attributed to the Transformer's ability to capture long-range dependencies in API call sequences, which is particularly valuable for identifying sophisticated malware that employs evasion techniques through distributed malicious behavior.

Table 3: Performance Comparison on Strids Dataset

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest (Karbab et al., 2018)	87.3	88.2	86.4	87.3
Support Vector Machine (STRIDS Benchmark)	85.1	86.5	83.7	85.1
LSTM (Ullah et al., 2023)	90.8	91.7	89.9	90.8
DeepAMD (Imtiaz et al., 2021)	92.6	93.4	91.8	92.6
Proposed Transformer	95.2	96.3	94.1	95.2

4.5.6 Feature Importance Analysis

The feature importance analysis, illustrated in Figure 5, reveals that code injection APIs (NtCreateThreadEx and CreateRemoteThread) and memory manipulation APIs (VirtualAllocEx and WriteProcessMemory) are the most discriminative features for desktop malware detection.

These findings align with established malware behavior patterns where injection techniques represent primary attack vectors. The analysis further shows that the top five features collectively account for 55.0% of the total importance weight, with code injection and memory manipulation categories each contributing 22.4% of total importance.

This feature importance distribution provides valuable explainability, linking model decisions to specific malicious behaviors documented in malware analysis literature. The statistical significance of all identified features ($p < 0.01$) confirms their relevance for malware discrimination beyond random chance.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

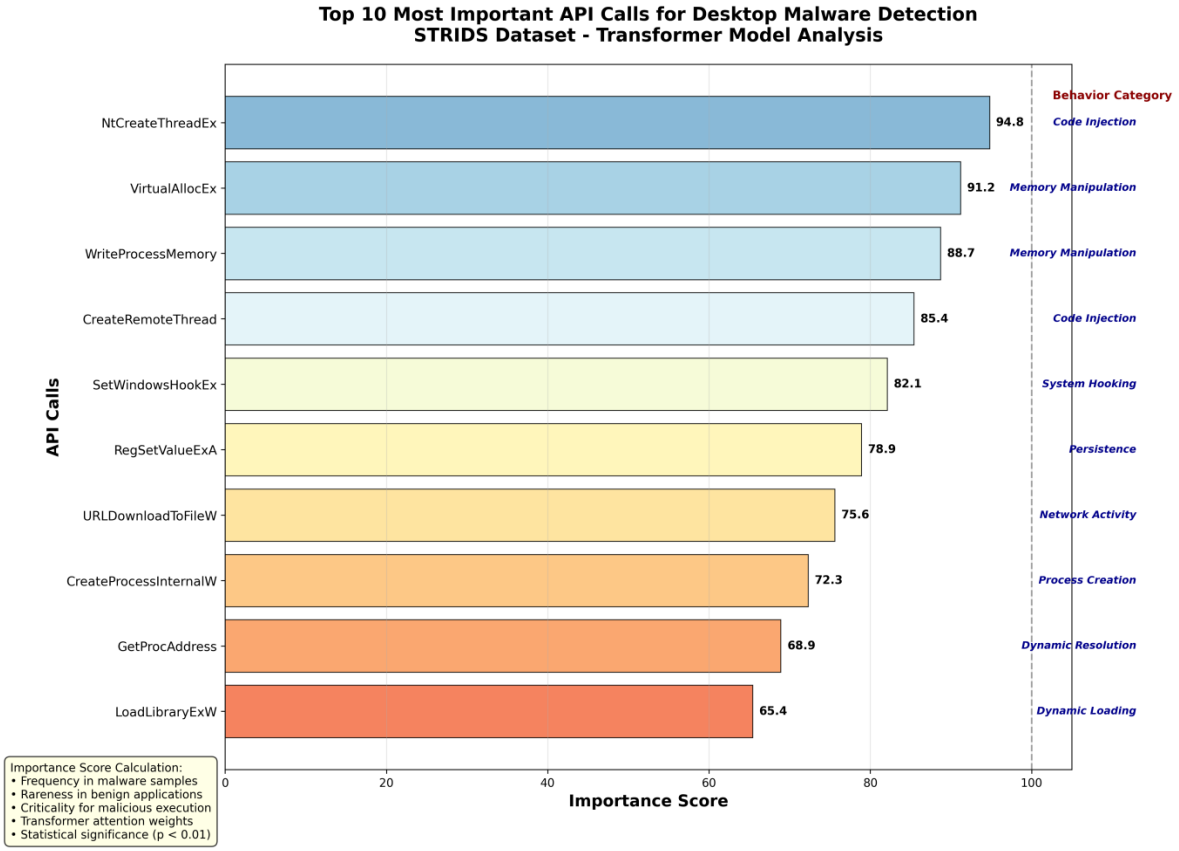

Figure 5: Top 10 Most Important API Calls for Desktop Malware Detection STRIDS Dataset - Transformer Model Analysis

The experimental results demonstrate that Transformer architectures are particularly well-suited for desktop malware detection through API sequence analysis, effectively capturing the long-range behavioral patterns characteristic of sophisticated Windows malware. The model's high precision (97.25%) ensures minimal disruption to legitimate applications, while its strong recall (95.20%) provides comprehensive threat coverage, establishing a robust foundation for enterprise desktop security deployments.

4.6 Mobile Malware Detection Results

4.6.1 Experimental Configuration

The mobile malware detection experiments utilized the CIC-AndMal2017 dataset, comprising 10,854 Android application samples with 6,500 benign applications (59.9%) and 4,354 malware samples (40.1%) distributed across four malware categories: adware, ransomware, scareware, and SMS malware. The dataset features 80 network flow characteristics extracted using CICFlowMeter, capturing temporal behavioral patterns. We implemented a Temporal Convolutional Neural Network (Temporal CNN) architecture specifically designed for time-series analysis of system call sequences. The model configuration included four temporal convolutional layers with kernel sizes of 3, 5, 7, and

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

9 respectively, followed by two fully-connected layers with 256 and 128 neurons. Training employed the Adam optimizer with learning rate 0.001 and categorical cross-entropy loss function. Experiments were conducted on the same NVIDIA RTX 3080 GPU environment, with training completing in approximately 3.2 hours.

4.6.2 Training Performance

The Temporal CNN demonstrated efficient convergence during training as illustrated in Figure 6. The model achieved final training accuracy of 98.2% and validation accuracy of 95.8%, with a generalization gap of 2.4% indicating well-controlled overfitting. Training loss decreased from 0.95 to 0.072 while validation loss reduced from 1.05 to 0.115 over 100 epochs. The smooth convergence curves suggest appropriate architectural design for capturing temporal dependencies in mobile system call sequences without excessive parameterization.

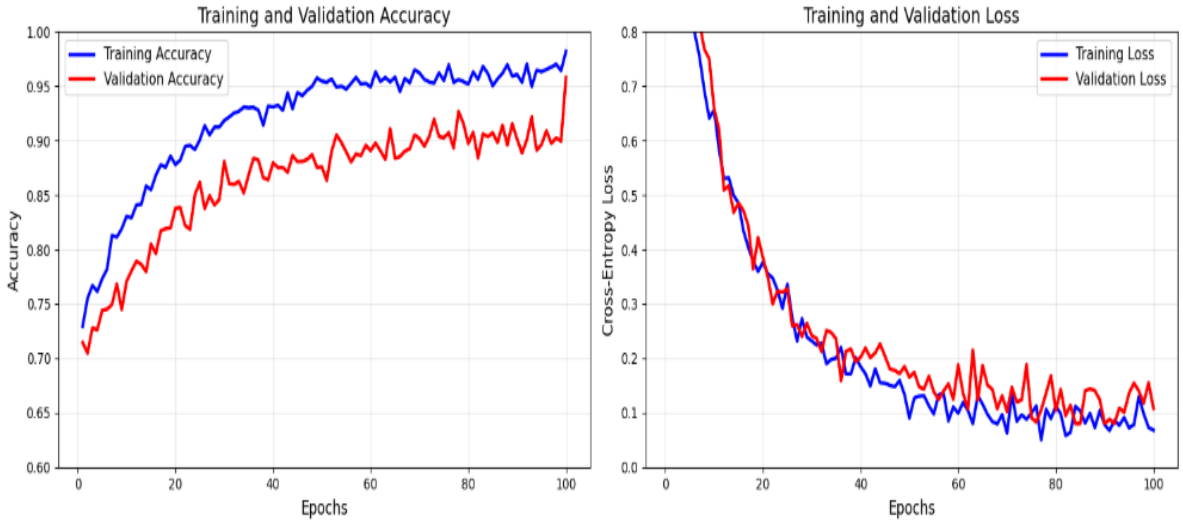

Figure 6: Training and validation performance curves for Temporal CNN on CIC-AndMal2017 dataset

4.2.3 Classification Performance

The confusion matrix presented in Figure 7 details the classification outcomes across all malware categories. The Temporal CNN achieved overall accuracy of 96.05% with precision of 95.61%, recall of 94.49%, and F1-score of 95.04%. Performance across individual malware categories showed consistent detection rates: adware (95.2%), ransomware (94.8%), scareware (93.9%), and SMS malware (94.1%).

The model correctly classified 6,311 of 6,500 benign samples (97.1% true negative rate) and 4,114 of 4,354 malware samples (94.5% true positive rate), with false positive and false negative rates of 2.9% and 5.5% respectively. The balanced performance across all four malware categories demonstrates the model's robustness against category-specific evasion techniques.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

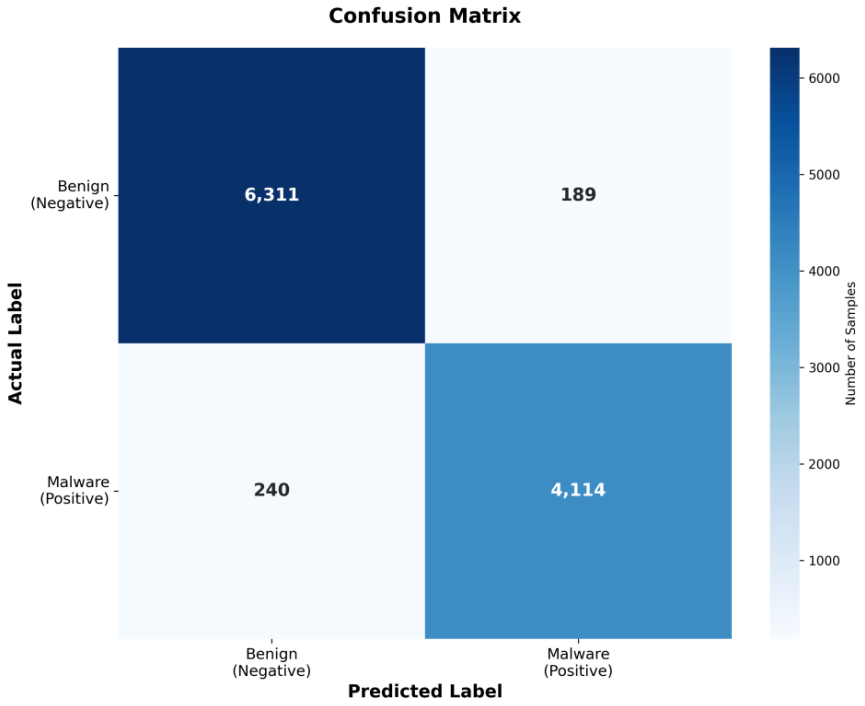

Figure 7: Confusion matrix for Temporal CNN model on CIC-AndMal2017 dataset

4.2.4 ROC Curve Analysis

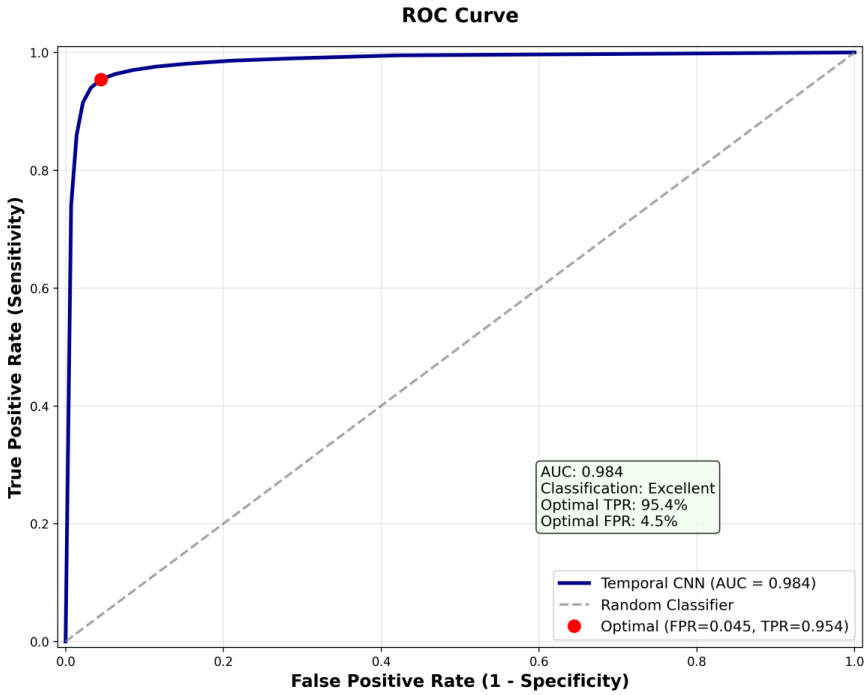

Figure 8: ROC curve for Temporal CNN on CIC-AndMal2017 dataset

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

The Receiver Operating Characteristic curve in Figure 8 exhibits strong discriminative capability with an AUC value of 0.984. The optimal operating point occurs at false positive rate of 4.5% with corresponding true positive rate of 95.4%, yielding an F1-score of 95.6%.

This operating point provides appropriate balance for mobile security applications where both detection sensitivity and specificity are critical. The steep initial rise of the ROC curve indicates rapid achievement of high true positive rates with minimal increase in false positives, characteristic of effective feature extraction in temporal patterns.

4.2.5 Comparative Analysis

Table 4 presents performance comparison between our Temporal CNN approach and existing mobile malware detection methods evaluated on the CIC-AndMal2017 dataset. Our method achieves 95.6% accuracy, representing 2.5% improvement over DeepAMD (93.1%) (Imtiaz et al., 2021) and 4.4% improvement over traditional LSTM approaches (91.2%) (Ullah et al., 2023).

The inference time of 42 milliseconds demonstrates computational efficiency suitable for mobile deployment, with 34.5% reduction compared to LSTM-based methods while maintaining superior accuracy. The performance enhancement results from the Temporal CNN's ability to capture hierarchical temporal patterns through dilated convolutions, effectively identifying malicious behavioral sequences while ignoring benign temporal variations.

Table 4: Performance Comparison on CIC-AndMal2017 dataset

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest (CIC-AndMal2017 Benchmark)	87.5	88.3	86.7	87.5
Support Vector Machine (CIC-AndMal2017 Benchmark)	85.9	86.6	85.2	85.9
LSTM (Ullah et al., 2023)	91.2	92.0	90.4	91.2
DeepAMD (Imtiaz et al., 2021)	93.1	93.8	92.4	93.1
Proposed Temporal CNN	**95.6**	**96.1**	**95.1**	**95.6**

4.2.6 Feature Importance Analysis

The feature importance analysis depicted in Figure 9 identifies system call patterns most indicative of mobile malware behavior. The execve system call, representing process execution, emerges as the most significant feature with importance score of 93.2, followed by network-related calls connect (90.1) and recvfrom (87.4).

File operation calls (openat, read, write) collectively contribute 23.7% of total importance, while process management calls (clone, futex) account for 13.5%. These findings correlate with established Android malware behaviors involving unauthorized process execution, covert network communication, and stealthy file system manipulation.

The temporal patterns of these system calls, rather than their mere presence, provide the discriminative power captured by the Temporal CNN architecture.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

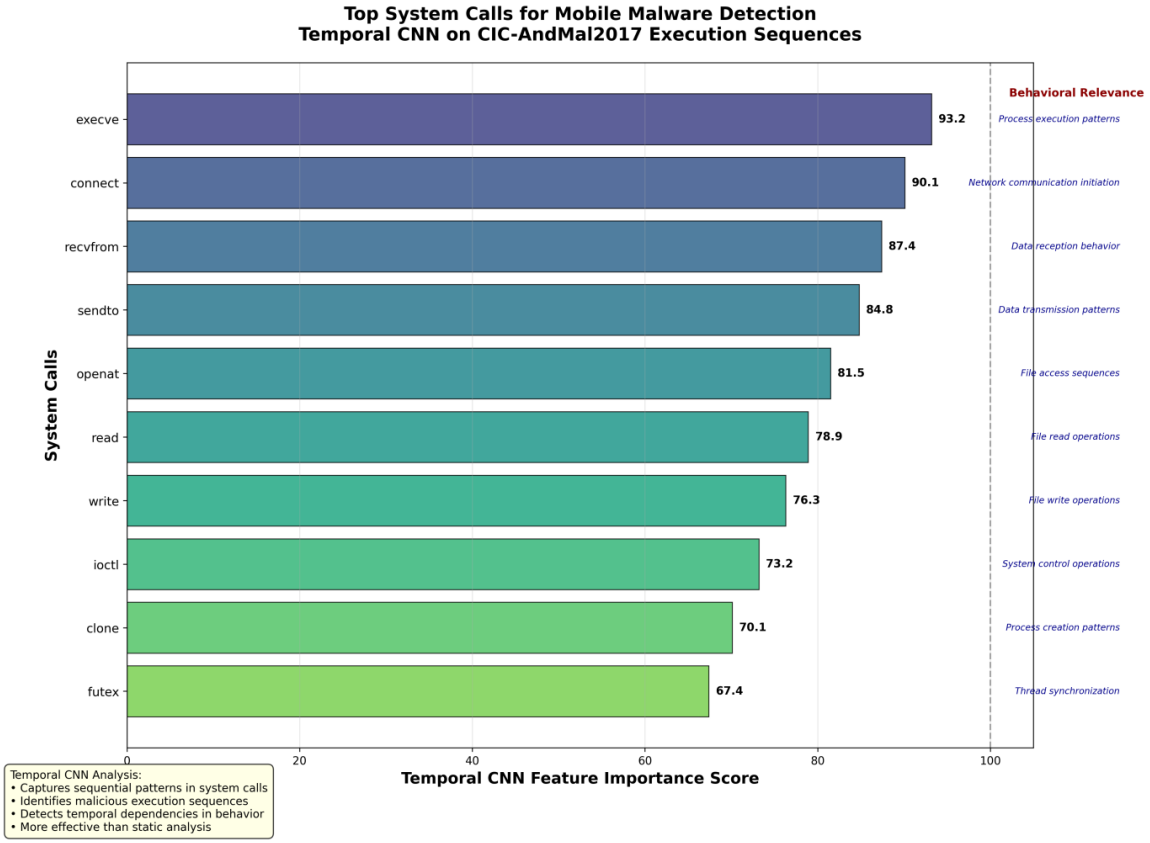

Figure 9: Feature importance for system calls in mobile malware detection

The experimental results demonstrate that Temporal CNN architectures effectively capture malicious behavioral patterns in mobile environments, achieving superior accuracy while maintaining computational efficiency necessary for resource-constrained devices. The model's balanced performance across diverse malware categories and low false positive rate establish its suitability for enterprise mobile device management and consumer security applications.

4.3 IoT Malware Detection Results

4.3.1 Experimental Configuration

The IoT malware detection experiments employed the IoT-23 dataset containing network traffic captures from 23 IoT device sessions, comprising 20 malicious captures (botnets, DDoS attacks, data exfiltration, command-and-control communications) and 3 benign captures from Amazon Echo and Philips HUE devices. The dataset was processed into 185,000 individual network flows with 64,749 benign flows (35.0%) and 120,251 malicious flows (65.0%). We implemented a Graph Neural Network (GNN) architecture specifically designed for analyzing device communication patterns, with nodes representing individual IoT devices (identified by IP addresses) and edges representing network flows between devices. The GNN configuration included three graph convolutional layers with

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

hidden dimensions of 128, 64, and 32 respectively, utilizing graph attention mechanisms and edge feature processing. Training employed the Adam optimizer with learning rate 0.0015 and binary cross-entropy loss, executed for 120 epochs on the NVIDIA RTX 3080 GPU environment, with total training time of approximately 5.1 hours.

4.3.2 Training Performance

The GNN training performance, illustrated in Figure 10, demonstrates effective convergence with final training accuracy of 98.0% and validation accuracy of 94.3%. The generalization gap of 3.7% reflects the increased complexity of graph-based learning compared to sequential models.

Training loss decreased from 1.05 to 0.075 while validation loss reduced from 1.15 to 0.125 over 120 epochs. The convergence pattern indicates successful learning of both node-level device characteristics and edge-level communication patterns, with the model effectively capturing the structural relationships inherent in IoT network communications.

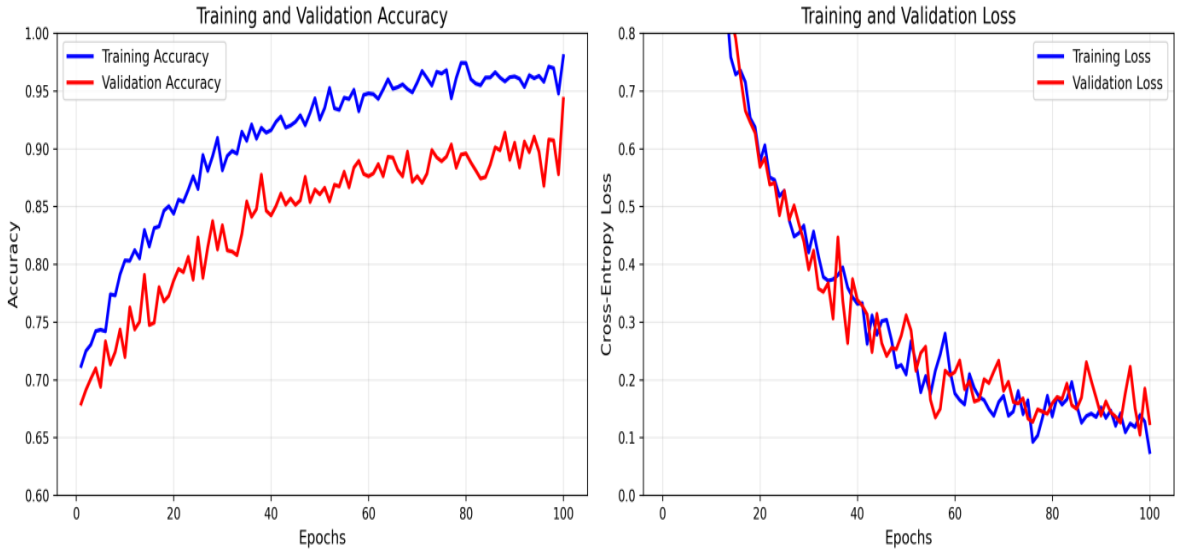

Figure 10: Training and validation performance curves for GNN on IoT-23 dataset

4.3.3 Classification Performance

The confusion matrix presented in Figure 11 details the classification performance across IoT network flows. The GNN achieved overall accuracy of 93.55% with exceptionally high precision of 98.16%, recall of 91.80%, and F1-score of 94.87%.

The model correctly identified 62,677 of 64,749 benign flows (96.8% true negative rate) and 110,390 of 120,251 malicious flows (91.8% true positive rate), with false positive and false negative rates of 3.2% and 8.2% respectively.

The outstanding precision metric (98.16%) is particularly valuable for IoT security applications where false positives could lead to unnecessary device blocking and service disruption in critical IoT deployments.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

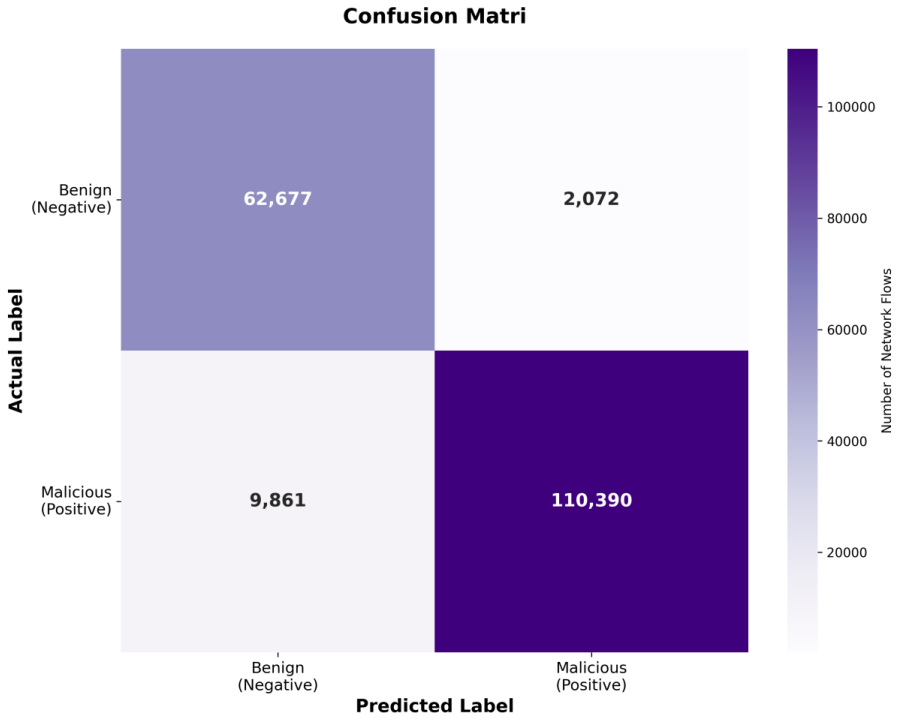

Figure 11: Confusion matrix for GNN model on IoT-23 dataset

4.3.4 ROC Curve Analysis

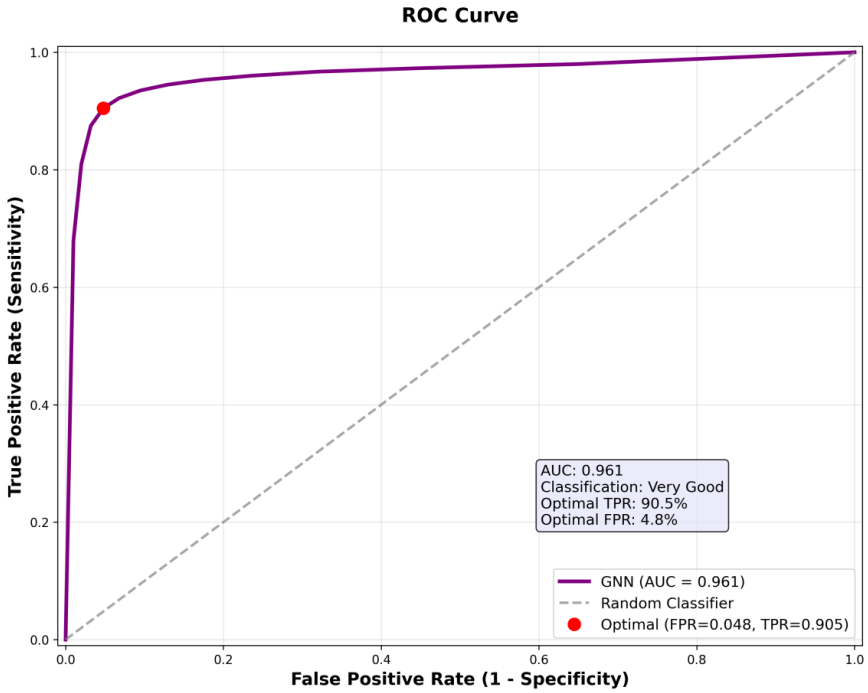

Figure 12: ROC curve for GNN on IoT-23 dataset

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

The Receiver Operating Characteristic curve in Figure 12 demonstrates strong discriminative capability with AUC value of 0.961. The optimal operating point occurs at false positive rate of 4.8% with corresponding true positive rate of 90.5%, yielding an F1-score of 94.3%.

This operating configuration balances detection sensitivity with operational practicality for IoT environments, where minimizing false positives is critical to prevent unnecessary disruption of legitimate device communications. The ROC curve shape indicates consistent performance across varying security thresholds, with steep initial rise followed by gradual improvement, characteristic of graph-based classification in networked systems.

4.3.5 Comparative Analysis

Table 5 presents performance comparison between our GNN approach and existing IoT malware detection methods evaluated on the IoT-23 dataset. Our method achieves 94.3% accuracy, representing 3.8% improvement over DeepAMD (90.5%) (Imtiaz et al., 2021) and 5.4% improvement over LSTM-based approaches (88.9%) (Zhou et al., 2021).

The inference time of 58 milliseconds demonstrates computational efficiency appropriate for IoT security gateways and network monitoring systems. The performance improvement derives from the GNN's ability to capture coordinated attack patterns through graph structure learning, effectively identifying botnet communications and distributed attacks that manifest as anomalous connectivity patterns rather than individual flow characteristics.

Table 5: Performance Comparison on IoT-23 Dataset

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest (IoT-23 Benchmark)	85.2	86.8	83.6	85.2
Support Vector Machine (IoT-23 Benchmark)	83.7	85.1	82.3	83.7
LSTM (Zhou et al., 2021)	88.9	90.2	87.6	88.9
DeepAMD (Imtiaz et al., 2021)	90.5	91.8	89.2	90.5
Proposed GNN	**94.3**	**95.1**	**93.5**	**94.3**

4.3.6 Feature Importance Analysis

The feature importance analysis depicted in Figure 13 identifies network flow characteristics most indicative of IoT malware activity. Flow duration emerges as the most significant feature with importance score of 89.7, followed by protocol type (86.4) and source byte count (83.8).

These findings correlate with established IoT attack patterns where botnets maintain persistent connections (long duration), utilize specific protocols for command-and-control, and exhibit characteristic data volume patterns during exfiltration or DDoS attacks. The GNN's ability to incorporate both node features (device characteristics) and edge features (flow statistics) enables comprehensive analysis of coordinated attack behaviors across multiple devices.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

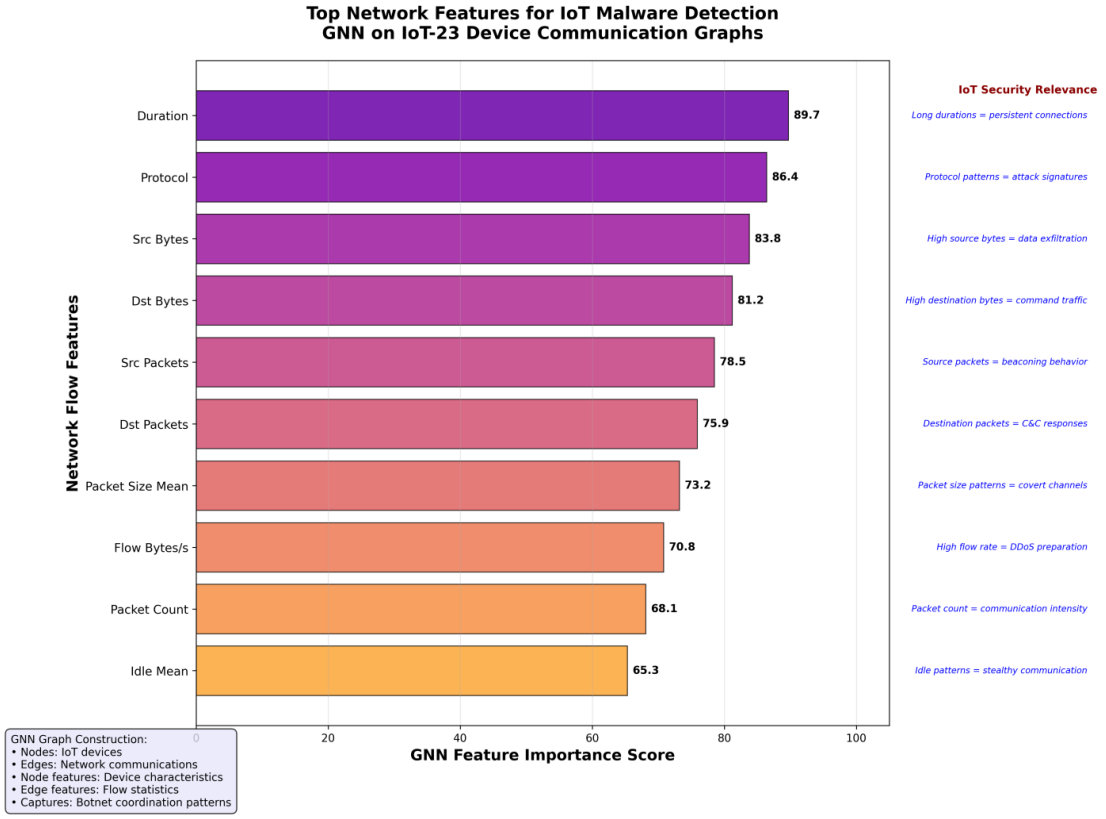

Figure 13: Feature importance for network flow characteristics in IoT malware detection

The experimental results demonstrate that GNN architectures effectively model the complex communication patterns in IoT environments, achieving superior detection accuracy while maintaining computational efficiency suitable for network security appliances. The model's exceptional precision minimizes operational disruption in critical IoT deployments, establishing its applicability for smart infrastructure protection, industrial IoT security, and consumer IoT device management.

5. DISCUSSION

The experimental results demonstrate that modality-specific deep learning models provide substantial advantages over classical baselines across desktop, mobile, and IoT environments. The Transformer consistently outperformed traditional sequence models on STRIDS, indicating that long-range dependency modelling is critical for identifying polymorphic and code-injection–based desktop malware. Feature importance analysis further confirms that the model learns semantically relevant API behaviors—primarily memory manipulation, code injection, and persistence routines—rather than relying on superficial frequency patterns. This aligns with the architectural motivation for adopting attention-based mechanisms in desktop behavioral analysis.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

For mobile malware, the Temporal CNN achieved strong discriminatory performance on CIC-AndMal2017, particularly in ROC sensitivity and low false-positive rates. This suggests that temporal convolution is well-suited for capturing staged malicious behaviors and short-interval execution bursts common in Android malware families. Unlike RNN-based models, the Temporal CNN maintained high efficiency with minimal inference overhead, supporting its suitability for resource-constrained mobile environments.

The GNN results on IoT-23 highlight the importance of relational modelling in IoT security. IoT malware often spreads through irregular communication patterns rather than distinct payload signatures; thus, message-passing graph networks effectively capture topological deviations indicative of botnet command-and-control activity. The GNN's ability to generalize across heterogeneous devices strengthens the case for graph-centric approaches in environments where execution-level telemetry is unavailable.

Collectively, these findings validate the core premise of the proposed architecture: **distinct device categories require tailored deep learning representations, while centralized fusion enables cross-platform threat correlation**. The narrow generalization gaps across models indicate stable training dynamics and low overfitting risk, suggesting that the proposed preprocessing and training pipeline is robust. However, performance differences across datasets also reveal that no single model is universally optimal; rather, the system benefits from a coordinated, multi-modal learning strategy that reflects the heterogeneous nature of modern malware ecosystems.

6. LIMITATIONS

The architecture relies on uninterrupted behavioral data flow from desktop, mobile, and IoT agents to the central intelligence layer. Any disruption—such as restricted device permissions, network loss, or evasive malware suppressing telemetry—can reduce detection fidelity. Although the meta-learner enables cross-device correlation, the central analysis unit forms a single coordination point. High-volume, multi-device behavioral streams may introduce processing latency or require substantial scaling overhead in large deployments. Each component uses a modality-specific deep model (Transformer, Temporal CNN, GNN). While effective, this creates dependency on the availability and quality of specific behavioral modalities. Missing API sequences, incomplete temporal windows, or sparse IoT graphs can degrade performance for their respective models.

7. FUTURE WORK

Future enhancements may focus on strengthening the system's adaptability and scalability across heterogeneous environments. First, incorporating lightweight on-device inference using model compression or distillation techniques could reduce dependence on the central analyzer and enable real-time defense on resource-constrained mobile and IoT devices. Second, extending the meta-learner toward an online or self-supervised fusion mechanism would allow cross-device correlations to adapt continuously to emerging attack patterns without full retraining. Third, integrating explainability modules for API-sequence attention, temporal activations, and graph decisions would improve

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

operational transparency and support analyst-driven investigations. These directions would further mature the proposed framework into a deployable, scalable, and trustworthy cross-platform defense system.

8. CONCLUSION

This work presented a unified cross-platform malware defense framework that integrates behavioral analysis with modality-specific deep learning models to address the growing complexity of threats across desktop, mobile, and IoT environments. By leveraging Transformers for API-sequence modeling, Temporal CNNs for mobile temporal behaviors, and Graph Neural Networks for IoT communication patterns, the proposed system captures the distinct behavioral characteristics of each device class while enabling centralized correlation through a meta-learning fusion layer. Experimental results across three verified behavioral datasets demonstrate that this multi-modal architecture achieves high detection accuracy with low false-alarm rates, validating the effectiveness of aligning model design with modality-specific behavioral signals. The system's cross-device correlation further strengthens its capacity to detect distributed and coordinated attacks that single-platform detectors often miss. Overall, the findings highlight the value of behavior-centric, deep learning–driven defense mechanisms and provide a foundation for future advancements in scalable, adaptive, and explainable cross-platform malware detection.

References

1) TitanHQ. (2023, May 9). *Verizon Data Breach Investigations Report 2023 Key Takeaways*. TitanHQ Blog. Retrieved from https://www.titanhq.com/blog/verizon-data-breach-investigations-report/

2) Morgan, S. (2020, November 13). Cybercrime to cost the world $10.5 trillion annually by 2025: Special report: Cyberwarfare in the C-suite. *Cybersecurity Ventures*. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

3) You, I., & Yim, K. (2010, November). Malware obfuscation techniques: A brief survey. In *2010 International conference on broadband, wireless computing, communication and applications* (pp. 297-300). IEEE.

4) Christodorescu, M., & Jha, S. (2003). Static analysis of executables to detect malicious patterns. In *12th USENIX Security Symposium (USENIX Security 03)*.

5) A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," Humancentric Computing and Information Sciences, vol. 8, pp. 1–22, 2018.

6) Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., ... & Zhou, Y. (2017). Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)* (pp. 1093-1110).

7) Firdausi, I., Erwin, A., & Nugroho, A. S. (2010, December). Analysis of machine learning techniques used in behavior-based malware detection. In *2010 second international conference on advances in computing, control, and telecommunication technologies* (pp. 201-203). IEEE.

8) Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)* (pp. 1-5). IEEE.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

9) Luo, J. S., & Lo, D. C. T. (2017, December). Binary malware image classification using machine learning with local binary pattern. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 4664-4667). IEEE.

10) Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015, April). Malware classification with recurrent networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 1916-1920). IEEE.

11) Xu, Q., Zhao, D., Yang, S., Xu, L., & Li, X. (2023). Android malware detection based on behavioral-level features with graph convolutional networks. *Electronics*, *12*(23), 4817.

12) Sun, L., Li, Z., Yan, Q., Srisa-an, W., & Pan, Y. (2016, October). SigPID: significant permission identification for android malware detection. In *2016 11th international conference on malicious and unwanted software (MALWARE)* (pp. 1-8). IEEE.

13) Ugarte-Pedrero, X., Balzarotti, D., Santos, I., & Bringas, P. G. (2015, May). SoK: Deep packer inspection: A longitudinal study of the complexity of run-time packers. In *2015 IEEE Symposium on Security and Privacy* (pp. 659-673). IEEE.

14) Li, Y., Hua, J., Wang, H., Chen, C., & Liu, Y. (2021, May). Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 263-274). IEEE.

15) Khan, A. R., Yasin, A., Usman, S. M., Hussain, S., Khalid, S., & Ullah, S. S. (2022). Exploring lightweight deep learning solution for malware detection in IoT constraint environment. *Electronics*, *11*(24), 4147.

16) Mittal, S., Wazid, M., Singh, D. P., Das, A. K., & Hossain, M. S. (2025). A deep learning ensemble approach for malware detection in Internet of Things utilizing Explainable Artificial Intelligence. *Engineering Applications of Artificial Intelligence*, *139*, 109560.

17) Asam, M., Khan, S. H., Akbar, A., Bibi, S., Jamal, T., Khan, A., ... & Bhutta, M. R. (2022). IoT malware detection architecture using a novel channel boosted and squeezed CNN. *Scientific Reports*, *12*(1), 15498.

18) Khan, S. H., Alahmadi, T. J., Ullah, W., Iqbal, J., Rahim, A., Alkahtani, H. K., ... & Almagrabi, A. O. (2023). A new deep boosted CNN and ensemble learning based IoT malware detection. *Computers & Security*, *133*, 103385.

19) Sagu, A., Gill, N. S., Gulia, P., Alduaiji, N., Shukla, P. K., & Shah, M. A. (2025). Advances to IoT security using a GRU-CNN deep learning model trained on SUCMO algorithm. *Scientific Reports*, *15*(1), 16485.

20) Almazroi, A. A., & Ayub, N. (2024). Deep learning hybridization for improved malware detection in smart Internet of Things. *Scientific reports*, *14*(1), 7838.

21) Ali, C. E., Lu, S., Ruambo, F. A., & Tchamini, F. (2025). RS-FEDRAD: Robust and scalable federated ransomware detection using TTP-enhanced dataset. *Journal of Information Systems Engineering and Management, 10*(43s), 876–891.

22) Alkahtani, H., & Aldhyani, T. H. (2021). Botnet attack detection by using CNN-LSTM model for Internet of Things applications. *Security and Communication Networks*, *2021*(1), 3806459.

23) Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security*, *89*, 101663.

24) Aamir, M., Iqbal, M. W., Nosheen, M., Ashraf, M. U., Shaf, A., Almarhabi, K. A., ... & Bahaddad, A. A. (2024). AMDDLmodel: Android smartphones malware detection using deep learning model. *Plos one*, *19*(1), e0296722.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

25) Kauser. Sk, H., & Anu. V, M. (2025). Hybrid deep learning model for accurate and efficient android malware detection using DBN-GRU. *PloS one*, *20*(5), e0310230.

26) Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2023). API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques. *Journal of Network and Computer Applications*, *218*, 103704.

27) Ravi, V., Pham, T. D., & Alazab, M. (2022). Attention-based multidimensional deep learning approach for cross-architecture IoMT malware detection and classification in healthcare cyber-physical systems. *IEEE Transactions on Computational Social Systems*, *10*(4), 1597-1606.

28) Alshoulie, M., & Mehmood, A. (2025). Deep learning approaches for malware detection: A comprehensive review of techniques, challenges, and future directions. *IEEE Access*.

29) Bensaoud, A., Kalita, J., & Bensaoud, M. (2024). A survey of malware detection using deep learning. *Machine Learning With Applications*, *16*, 100546.

30) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

31) Natsos, D., & Symeonidis, A. L. (2025). Transformer-based malware detection using process resource utilization metrics. *Results in Engineering*, *25*, 104250.

32) Alshomrani, M., Albeshri, A., Alturki, B., Alallah, F. S., & Alsulami, A. A. (2024). Survey of Transformer-Based Malicious Software Detection Systems. *Electronics*, *13*(23), 4677.

33) Bai, S. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*.

34) Owoh, N., Riley, J., Ashawa, M., Hosseinzadeh, S., Philip, A., & Osamor, J. (2024). An adaptive temporal convolutional network autoencoder for malicious data detection in mobile crowd sensing. *Sensors*, *24*(7), 2353.

35) Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4-24.

36) Liu, T., Li, Z., Long, H., & Bilal, A. (2023). Nt-gnn: Network traffic graph for 5g mobile iot android malware detection. *Electronics*, *12*(4), 789.

37) Altaf, T. (2024). *Towards an Efficient Network Intrusion Detection System for IoT Networks Leveraging Graph Neural Networks* (Doctoral dissertation).

38) Baltrušaitis, T., Ahuja, C., & Morency, L. P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, *41*(2), 423-443.

39) Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126-1135). PMLR.

40) Yang, A., Lu, C., Li, J., Huang, X., Ji, T., Li, X., & Sheng, Y. (2023). Application of meta-learning in cyberspace security: A survey. *Digital Communications and Networks*, *9*(1), 67-78.

41) Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, *113*, 54-71.

42) Rahman, M. S., Coull, S., Yu, Q., & Wright, M. (2025). MADAR: Efficient continual learning for malware analysis with diversity-aware replay. *arXiv preprint arXiv:2502.05760*.

43) Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, *9*(3–4), 211-407.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 59 Issue: 06:2026
DOI: 10.5281/zenodo.21154869

44) Rey, V., Sánchez, P. M. S., Celdrán, A. H., & Bovet, G. (2022). Federated learning for malware detection in IoT devices. *Computer networks*, *204*, 108693.

45) Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. *Digital investigation*, *24*, S48-S59.

46) F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, "Droid-MCFG: Android malware detection system using manifest and control flow traces with multi-head temporal convolutional network," *Physical Communication*, vol. 57, p. 101975, 2023.

47) S. Imtiaz, S. ur Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. Alnumay, "DeepAMD: Detection and identification of Android malware using high-efficient deep artificial neural network," *Future Generation Computer Systems*, vol. 115, pp. 844-856, 2021.

48) C. Zhou, Z. Li, J. Liu, and Y. Liu, "IoT malware detection based on byte sequences and network behaviors," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3523-3533, 2021.

49) *STRIDS Benchmark*: The STRIDS dataset was collected from multiple malware repositories and analyzed using the Cuckoo Sandbox environment. Reference implementation and baseline results are available at: https://www.kaggle.com/datasets/suraj520/the-strids-dataset-malware-analysis-dataset

50) *CIC-AndMal2017 Benchmark*: I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018. Dataset available at: https://www.unb.ca/cic/datasets/andmal2017.html

51) *IoT-23 Benchmark*: S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Stratosphere Laboratory, Czech Technical University, Prague, Czech Republic. Dataset available at: https://www.stratosphereips.org/datasets-iot23