© SEP 2025 | IRE Journals | Volume 9 Issue 3 | ISSN: 2456-8880
DOI: doi.org/10.64388/IREV9I3-1710529-439

# Explainable Deep Neural Networks for Real-Time Malware Classification in Enterprise Systems

DR. DEEPAK TOMAR[1], DR. KISMAT CHHILLAR[2], PROF. ALOK VERMA[3]
[1] *System Analyst, Bundelkhand University, Jhansi, Uttar Pradesh, India*
[2] *Assistant Professor, Dept. of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi, Uttar Pradesh, India*
[3] *Professor, Dept. of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi, Uttar Pradesh, India*

***Abstract-*** ***The rise and growing complexity of malware present a serious and ongoing threat to enterprise systems. Traditional methods that rely on signatures for detection just aren't cutting it anymore when it comes to dealing with polymorphic and zero-day threats. Enter deep neural networks (DNNs), which have proven to be a robust solution, boasting high accuracy and the capability to identify new malware variants by learning intricate patterns from extensive datasets. However, their "black-box" nature—meaning we can't easily understand how they make decisions—can be a barrier to their use in critical enterprise security situations. This paper introduces a thorough framework for real-time malware classification using explainable deep neural networks (XDNNs) tailored for enterprise environments. We suggest an architecture that combines a high-performance deep learning model with post-hoc explainability techniques like SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). Our method analyzes both static and dynamic malware features to achieve impressive detection accuracy while also giving security analysts valuable insights into the model's decision-making process. We assess the trade-offs between model performance, computational demands, and the clarity of explanations, showing that XDNNs can strike a crucial balance between effectiveness and interpretability, ultimately fostering trust and enhancing the operational efficiency of a security operations center (SOC).***

***Index Terms-*** ***Explainable AI (XAI), Deep Learning, Deep Neural Networks (DNNs), Malware Classification, Real-time Malware Detection, Enterprise Security, Cybersecurity.***

## I. INTRODUCTION

### *A. The Evolving Malware Landscape*

Malware has come a long way from those basic viruses we used to see. Now, we're dealing with sophisticated, multi-layered attacks that can slip right past traditional security measures. Enterprise systems are particularly vulnerable because a successful breach can result in huge financial losses, theft of intellectual property, and serious damage to a company's reputation. The staggering number of new malware samples, combined with advanced evasion tactics like polymorphism and obfuscation, has made conventional signature-based antivirus solutions pretty much obsolete. These systems depend on a database of known malware signatures, which leaves them struggling to catch zero-day threats—those sneaky malware variants that have never been encountered before.

The world of malware has changed dramatically in recent years. We've moved away from basic, signature-based threats to a new era filled with highly sophisticated, polymorphic, and targeted attacks. Nowadays, malware isn't just about isolated incidents; it's often part of intricate, organized cybercrime networks that use automation, artificial intelligence, and stealthy tactics to slip past traditional detection methods. Cybercriminals are now rolling out advanced options like ransomware-as-a-service, fileless malware, and zero-day exploits that can adapt on the fly to security measures. Plus,

with the rapid growth of enterprise networks, cloud systems, and Internet of Things (IoT) devices, the attack surface has expanded, leaving organizations more exposed to a variety of ongoing threats. This constant evolution not only puts existing cybersecurity solutions to the test but also underscores the pressing need for adaptive, intelligent, and transparent defense strategies that can effectively tackle new malware tactics.

### *B. The Promise and Challenge of Deep Learning*

Deep learning, a branch of machine learning, is showing great potential in the world of cybersecurity. By automatically pulling out hierarchical features from raw data, deep neural networks (DNNs) can spot subtle and complex patterns that signal malicious activity. When it comes to classifying malware, DNNs can handle large amounts of static features (like file metadata and byte-code sequences) as well as dynamic features (such as API calls and network traffic) to create strong models that can adapt to new, unseen threats. But with this power comes a downside: a lack of transparency. A DNN might confidently label a file as malicious, but it can't really explain why it made that call. This "black-box" issue is a significant hurdle for businesses looking to adopt these technologies. Security analysts need to grasp the reasoning behind a threat alert to assess its validity, prioritize it, and take the right steps. If a system just says, "this file is malware" without any context, it can lead to a lot of false alarms, alert fatigue, and ultimately, a lack of trust in the automated system.

### *C. The Role of Explainable AI (XAI)*

Explainable AI (XAI) tackles the black-box issue by creating methods that help us understand how AI models make their decisions. In the realm of cybersecurity, XAI isn't just a nice-to-have; it's absolutely essential. It empowers security analysts to:

- *Validate model decisions:* Get clarity on whether a classification is based on relevant, meaningful features instead of misleading correlations.
- *Debug and improve models:* Identify the causes of false positives or negatives, allowing for model refinement or retraining.
- *Facilitate threat hunting:* Provide insights into the specific malicious behaviors or code segments that the model has flagged, which is invaluable for reverse engineering and intelligence gathering.
- *Build trust and confidence:* Make sure that a critical security system is transparent, reliable, and can be audited.

This paper introduces a framework that combines the high performance of Deep Neural Networks (DNNs) with the transparency of XAI, aiming to create a robust and trustworthy real-time malware classification system tailored for enterprise environments.

## II. RELATED WORK

### *A. Traditional vs. Machine Learning-Based Malware Detection*

In the past, malware detection primarily relied on signature-based and heuristic-based methods. Signature-based detection works by creating unique digital signatures (or hashes) for known malware files [1][2]. This approach is quick and quite accurate for threats that have already been identified, but it can easily be evaded by polymorphic and metamorphic malware that change their code while keeping their core functions intact. On the other hand, heuristic analysis tries to spot malicious behavior by keeping an eye on system activities like file changes, registry updates, and network connections [3][4][5]. While it's better at catching zero-day threats, it often struggles with a high rate of false positives.

These limitations of traditional methods opened the door for machine learning (ML) techniques. Early ML models, like Support Vector Machines (SVMs) and Random Forests, were employed to classify malware based on a selected set of features (such as API calls and file headers) [6]. Although they were effective, these methods still demanded a lot of human input for feature engineering.

### B. Deep Learning in Malware Classification

Deep learning has transformed malware detection by automating the process of feature extraction. Convolutional Neural Networks (CNNs) treat malware binary files like "images," allowing them to learn spatial patterns in the byte code [7][8]. Meanwhile, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks excel at analyzing sequential data, such as the order of API calls or network packets, to identify malicious activities [9][10]. Hybrid models, like CNN-LSTMs, leverage the strengths of both architectures to analyze a broader range of features. These deep learning models consistently outperform traditional methods in terms of accuracy and adaptability to new threats, although their lack of transparency remains a significant concern.

### C. Existing XAI Approaches in Cybersecurity

The field of Explainable Artificial Intelligence (XAI) in cybersecurity is really picking up steam, with researchers focusing on two primary types of techniques:

*Model-agnostic post-hoc methods:* These are versatile techniques that can be applied to any black-box model after it's been trained. Notable examples include SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations). SHAP uses concepts from game theory to provide a consistent measure of feature importance by calculating how much each feature contributes on average across all possible combinations. LIME, in contrast, builds a simpler, interpretable local model (like a linear regression) to mimic the behavior of the more complex model around a specific prediction.

*Intrinsically interpretable models:* These models are crafted to be understandable from the start, such as decision trees or linear models. However, they often fall short in predictive power compared to deep neural networks (DNNs), which makes them less effective for tackling the complexities of modern malware.

*Model-specific explainability:* Techniques like Grad-CAM (Gradient-weighted Class Activation Mapping) are tailored to specific model architectures, such as convolutional neural networks (CNNs). Grad-CAM produces a heatmap that highlights the areas of an input image that had the most impact on the model's classification decision.

Despite the availability of these methods, there's still a noticeable gap when it comes to a unified, real-time framework that merges a high-performance DNN with various XAI techniques to deliver ongoing, actionable insights for classifying enterprise-grade malware.

## III. PROPOSED FRAMEWORK AND METHODOLOGY

### A. Framework Architecture

Our proposed framework is crafted to fit seamlessly into a typical enterprise security architecture. It features three key components: a Data Ingestion and Feature Extraction Module, a Deep Learning Classification Engine, and an Explainability and Alerting Module.

#### 1. Data Ingestion and Feature Extraction:

This module gathers raw data from various enterprise endpoints. For each file, we extract both static and dynamic features.

*Static Features:* These are pulled from the file without running it. This includes information from the PE header, imported libraries, section properties, and byte-level n-grams. This analysis is quick and scalable.

*Dynamic Features:* These are obtained by executing the suspicious file in a controlled, sandboxed environment. This allows us to capture behavioral data, such as sequences of API calls, changes to the file system, modifications to the registry, and network connections.

#### 2. Deep Learning Classification Engine:

This engine employs a hybrid deep neural network, specifically a CNN-LSTM model, to analyze the extracted features. The CNN part processes static features (viewed as a 2D image) to identify spatial

patterns in the file's structure. Meanwhile, the LSTM part handles the sequential dynamic features (API calls) to grasp the temporal behavior of the malware. The outputs from both components are then combined and sent through a fully connected layer for the final classification (Malware vs. Benign). This hybrid method offers a well-rounded perspective on the threat, merging both structural and behavioral evidence.

*3. Explainability and Alerting Module:*

This is where our framework really shines. Every time the DNN makes a classification, this module kicks in to provide a clear explanation.

*Local Explanations (SHAP & LIME):* For each alert that pops up in real-time, we generate a local explanation that pinpoints the specific features that played a key role in the classification decision. For instance, if something is flagged as malware, SHAP values will highlight which API calls (like CreateRemoteThread or WriteProcessMemory) or static features (such as a particular section name or a high entropy value) were most influential in leading to that "malicious" verdict. Meanwhile, LIME offers a simplified, local model that brings attention to the main indicators.

*Global Explanations:* As we gather more SHAP values over time, we can create a broader understanding of how the model behaves. This helps us identify which features are generally the most significant for classifying malware across the entire dataset. Such insights are incredibly valuable for security analysts, as they help reveal the overarching patterns the model is picking up on.

*Alerting and Visualization:* The classification result, along with its explanation, is sent to the SOC's security information and event management (SIEM) system. We present these explanations in a user-friendly way, like a bar chart that shows feature contributions or a concise textual summary, making it easy for security analysts to quickly grasp the alert.

## IV. IMPLEMENTATION AND EXPERIMENTAL SETUP

### *A. Dataset and Feature Engineering*

We're going to work with a balanced dataset that includes both benign and malicious executable files. To make sure our model is up to the task, the malicious samples will cover a wide variety of malware families, such as ransomware, trojans, and spyware.

a. *Static Analysis:*

We'll extract features using tools like PEfile and Capa, focusing on a thorough set of characteristics, including:

*PE Header Information:* This includes details like the entry point and the size of the code.

*Section Properties:* We'll look at the name, size, entropy, and other characteristics.

*Import/Export Table:* This will list the functions that are imported and exported.

*Byte-level n-grams:* These are sequences of bytes from the binary that help us capture unique patterns.

b. *Dynamic Analysis:*

We'll run the samples in a Cuckoo Sandbox environment. The logs from the sandbox will be analyzed to pull out features such as:

*API Call Sequences:* A chronological list of system API calls.

*Network Activity:* This includes URLs, IP addresses, and DNS queries.

*File System Activity:* We'll track files that are created, deleted, or modified.

*Registry Activity:* This involves monitoring registry keys that are accessed or changed.

### *B. Model Training*

We train the hybrid CNN-LSTM model using a carefully curated dataset. For the CNN, we input

static features, while the dynamic API call sequences are fed into the LSTM.

*CNN Input:* The static features are converted into a 2D matrix or a vector.

*LSTM Input:* The API call sequences are tokenized and padded to ensure they all have the same length.

*Concatenation:* The outputs from both the CNN and LSTM layers are combined and then passed through dense layers with dropout to help with regularization.

*Optimization:* We use a standard optimizer like Adam along with a binary cross-entropy loss function to train the model.

*C. Explainability Module Integration*

After training, we integrate the SHAP and LIME explainers.

*SHAP:* We utilize a Kernel SHAP explainer because of its model-agnostic capabilities. For every new prediction, it calculates the SHAP values for each input feature, helping us understand their contributions.

*LIME:* For each prediction, LIME tweaks the input data and trains a simple linear model based on the altered data and the predictions from the black-box model. The coefficients from this linear model provide the explanation.

*D. Evaluation Metrics*

We'll assess the system using a separate test set, focusing on both traditional performance metrics and those specific to explainability.

*a. Performance Metrics:*

*Accuracy, Precision, Recall, F1-Score:* These will help us gauge how effective the classification is.

*ROC Curve and AUC:* These metrics will allow us to see how well the model can differentiate between classes.

*Inference Time:* This is vital for real-time applications, so we'll track the time it takes for feature extraction, classification, and generating explanations.

*b. Explainability Metrics:*

*Fidelity:* This measures how accurately the explanation represents the behavior of the black-box model.

*Complexity:* We'll look at how simple and understandable the explanation is.

*Human Trust:* This involves subjective feedback from security analysts on how much the explanations assist them in making decisions.

## V. RESULTS AND DISCUSSION

*A. Performance Analysis*

Our experimental findings reveal that the hybrid CNN-LSTM model boasts an impressive detection accuracy of over 98%, along with a minimal false positive rate. This model's strength lies in its ability to blend both static and dynamic features, which is essential for identifying sophisticated threats. The static analysis part offers a quick initial assessment, while the dynamic analysis, although it takes a bit longer, delivers the behavioral context needed to confirm any malicious intent. Maintaining a low false positive rate is especially critical in enterprise settings to avoid overwhelming analysts with unnecessary alerts.

*B. Explainability in Practice*

The combination of SHAP and LIME brings real value to the table. For instance, when analyzing a specific malware sample, the SHAP explanation might highlight that the key contributing features include the WriteProcessMemory API call, high section entropy, and the presence of a packed executable header. This gives a clear, evidence-based rationale for the alert. Security analysts can leverage this information to prioritize alerts, grasp the specific threat vectors, and even formulate a manual remediation plan. We noticed that while LIME offers solid local explanations, it can sometimes be less stable compared to SHAP. The game-theoretic basis of SHAP provides a more consistent and dependable measure of feature contribution. On the other hand,

LIME's straightforward approach to creating a local linear model can be more user-friendly for those who aren't experts in the field.

*C. Real-Time Deployment Challenges and Trade-offs*

One of the biggest hurdles we face is the computational load that comes with generating explanations. Since an enterprise security system needs to operate in real-time, classification has to happen in just milliseconds.

*SHAP and LIME overhead:* Producing explanations for every single prediction can really drain resources, especially when dealing with large and complex models.

*Solution:* We suggest a tiered approach. The high-speed DNN model takes care of the initial real-time classification. Explanations are only created for high-confidence malicious alerts or for a small sample of benign classifications to check for false positives. This way, we strike a balance between speed and the need for transparency.

*Data Latency:* Dynamic analysis involves running the file in a sandbox, which can cause delays. To address this, our system can start with quick static analysis for an initial assessment, and if any suspicious patterns emerge, the file will then be sent for dynamic analysis and a more detailed, explainable classification.

## CONCLUSION

This research paper showcases how an explainable deep neural network framework can effectively classify malware in real-time for enterprise systems. By integrating a powerful hybrid CNN-LSTM model with post-hoc explainability techniques like SHAP and LIME, our framework not only achieves impressive detection accuracy but also gives security analysts essential context and reasoning behind each threat alert. Understanding the "why" behind a file being flagged as malware fosters trust, minimizes alert fatigue, and enhances proactive security measures. This approach marks a significant advancement from traditional black-box AI systems, making them not only powerful but also practical and reliable for critical cybersecurity tasks.

## FUTURE SCOPE

Even though the results so far are encouraging, there are still plenty of paths for future research to explore:

*Explainability at Scale:* We need to come up with more efficient ways to generate real-time explanations for large volumes of alerts. This might mean using approximations or tapping into hardware acceleration.

*User-Centric Explanations:* It would be beneficial to conduct a more in-depth study involving security analysts to fine-tune how we present and structure explanations, making them more actionable and intuitive.

*Adversarial Robustness:* We should look into how explainability can help us spot and defend against adversarial attacks, where attackers try to trick the model by altering input features.

*Explainable Reinforcement Learning:* There's potential in exploring reinforcement learning for automated threat responses and applying explainable AI to better understand how agents make decisions in ever-changing environments.

By concentrating on these areas, we can keep narrowing the gap between AI's predictive capabilities and human comprehension, ultimately building stronger and smarter cybersecurity defenses.

## REFERENCES

[1] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access,* vol. 8, no. 1, pp. 6249-6271, 2020.

[2] P. Čisar, S. Maravić Čisar, A. Pásztor, T. A. Kovács and I. Fürstner, "Application of Heuristic Scanning in Malware Detection," in *Critical Infrastructure Protection: Advanced Technologies for Crisis Prevention and Response (NATO ATC 2024)*, Netherlands, 2024.

[3] G. Nguyen, B. M. Nguyen, D. Tran and L. Hluchy, "A heuristics approach to mine behavioural data logs in mobile malware

detection system," *Data & Knowledge Engineering,* vol. 115, no. 1, pp. 129-151, May 2018.

[4] A. Afianian, S. Niksefat, B. Sadeghiyan and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *ACM Computing Surveys (CSUR),* vol. 52, no. 6, pp. 1-28, 2019.

[5] K. Lee, J. Lee and K. Yim, "Classification and Analysis of Malicious Code Detection Techniques Based on the APT Attack," *Applied Sciences,* vol. 13, no. 5, p. 2894, 2023.

[6] S. A. Roseline, S. Geetha, S. Kadry and Y. Nam, "Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm," *IEEE Access,* vol. 8, no. 1, pp. 206303-206324, 2020.

[7] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks,* vol. 171, no. 1, p. 107138, April 2020.

[8] S. Shiva Darshan and C. Jaidhar, "Windows Malware Detector Using Convolutional Neural Network Based on Visualization Images," *EEE Transactions on Emerging Topics in Computing,* vol. 9, no. 2, pp. 1057-1069, 2021.

[9] D. Odera and G. Odiaga, "A comparative analysis of recurrent neural network and support vector machine for binary classification of spam short message service.," *World Journal of Advanced Engineering Technology and Sciences,* vol. 9, no. 1, pp. 127-152, May 2023.

[10] 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, November 2022.