International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

Advancing One-Shot Malware Categorization by Enhancing Early-Stage Detection Techniques

Mentor details
Dr. B Santosh Kumar

Shunnewar Rakesh
Cybersecurity
Institute of Aeronautical Engineering Hyderabad, India

Nikitha Muthyala
Cybersecurity
Institute of Aeronautical Engineering Hyderabad, India

Ramavath Hemavathi
Cybersecurity
Institute of Aeronautical Engineering Hyderabad, India

***Abstract* - The need for sophisticated and intelligent detection systems that can recognize new and changing threats has increased due to the malware's explosive growth. When it comes to identifying previously undiscovered malware variants, especially in their early stages, traditional signature-based and heuristic approaches frequently fall short. This study introduces a new method for classifying malware that makes use of one-shot learning techniques—more precisely, a Siamese Neural Network architecture. Based on learned similarity metrics, the suggested system successfully generalizes to detect and classify unseen malware after being trained on a small number of samples. Flask was used to create a real-time prediction API that enabled interactive file uploads and automated analysis driven by artificial intelligence. It was backed by a secure frontend interface and a PostgreSQL-based logging system. Experiments on a carefully selected malware dataset show that the model requires little training data and achieves high accuracy in early-stage detection. The findings support one-shot learning's potential for low-data, scalable malware classification systems and imply that this design can greatly speed up cybersecurity operations' response times. An effective and deployable AI-based framework for improving early malware detection capabilities in practical settings is provided by this study.**

Keywords: One-shot learning, Malware detection, Siamese neural network, Cybersecurity, Early-stage classification, Deep learning, Flask API, Threat analysis, AI-based malware categorization, Minimal data training.

I. INTRODUCTION

Malware threats are becoming more sophisticated and more frequent in the quickly changing field of cybersecurity. Conventional malware detection systems find it difficult to instantly adjust to new or low-prevalence threats because they frequently rely on sizable, labeled datasets and signature-based techniques. System integrity is seriously jeopardized by this restriction, particularly in high-security settings where prompt action is necessary. To overcome this difficulty, we suggest a malware classification framework based on one-shot learning that uses little training data to identify and categorize malware sam the model to learn distance-based relationships and generalize to unseen samples by utilizing a Siamese Neural Network (SNN) architecture that has been trained on vectorized malware features. In contrast to traditional deep learning techniques, which call for thousands of labeled examples for each class, our model is especially well-suited for zero-day attack scenarios because it is tuned for early-stage detection.

We further improve this system's usefulness by incorporating the trained model into a Flask-based REST API, which allows for easy malware file uploading and real-time threat analysis. Users can view detection trends, download reports, and keep an eye on scan history using a safe and user-friendly web dashboard.

In addition to showcasing high accuracy and robustness in early detection with minimal data, this paper covers the entire project lifecycle, from dataset processing and model training to deployment. The effectiveness of one-shot learning models in next-generation cybersecurity defense systems is demonstrated by our findings.

II. RELATED WORK

Signature-based methods, which compare incoming files to databases of known malware signatures, have been the mainstay of malware detection for a long time. These systems perform poorly against zero-day assaults despite being successful against threats that have already been recognized. They also show notable weaknesses when faced with new, polymorphic, or obfuscated malware variants [1]. As a result, signature-based approaches are frequently inadequate for contemporary cybersecurity requirements that call for quick and flexible responses.

Deep learning (DL) and machine learning (ML) approaches have been thoroughly investigated for malware detection in an effort to address these drawbacks. In order to increase generalization capabilities, machine learning (ML) techniques use characteristics taken from static binary or

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

dynamic behavioral analysis to train classifiers such decision trees, support vector machines (SVMs), and random forests [2][3]. DL approaches, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have demonstrated superior performance by learning hierarchical and temporal features directly from raw data, reducing the need for manual feature engineering [4]. Despite their efficacy, these models generally require substantial labeled datasets, which are costly and time-consuming to curate—especially for rare or
emerging malware families.

The research community has recently resorted to few-shot and one-shot learning paradigms in order to overcome the problem of minimal labeled data. The goal of these methods is to teach generalized representations and similarity metrics so that models can categorize new categories with few samples. Model-agnostic meta-learning (MAML) and archetypal networks are two examples of meta-learning frameworks that have been successfully used to cybersecurity challenges [5]. Siamese Neural Networks (SNNs) in particular have become a potent tool for low-data classification issues because they learn to compare input pairs using shared weights and a distance measure [6].While SNNs have achieved success in domains like biometric authentication and image recognition, their application in malware detection remains nascent and largely unexplored in practical deployments.

By using a Siamese Neural Network architecture trained on vectorized malware properties, our suggested framework makes progress in this field by allowing for the quick and precise categorization of malware samples that haven't been seen before with a small number of training instances. In contrast to traditional deep learning techniques, our methodology significantly lessens reliance on sizable, labeled datasets, which makes it especially appropriate for early-stage and zero-day malware detection. Furthermore, by implementing the system as a scalable Flask-based REST API with a safe and engaging web dashboard, we close the gap between theoretical models and practical cybersecurity. This end-to-end solution improves deployment readiness and practical applicability by enabling real-time malware analysis, results tracking, and user-friendly monitoring.

III. PROPOSED METHODOLOGY

This section describes the detailed design and implementation of the proposed one-shot malware categorization system, including dataset preparation, model architecture, training methodology, and deployment framework.

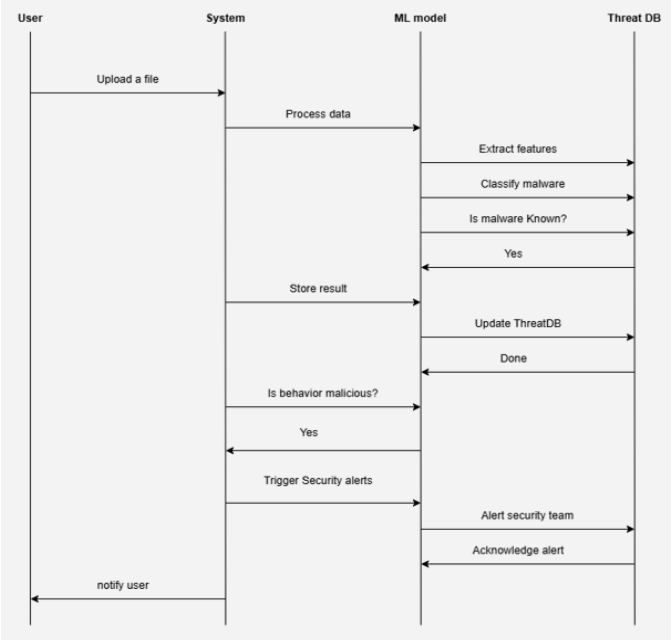
3.1 Dataset and Preprocessing

n addition to the EMBER 2018 dataset, the Drebin dataset was used to evaluate the system on Android malware. Drebin is a benchmark static malware dataset consisting of over 5,600 malicious APKs and 120,000 benign apps, extracted from real-world Android applications. Feature extraction was performed on AndroidManifest.xml and classes.dex files to derive permissions, API calls, and hardware features. These were vectorized and normalized to form input tensors suitable for the Siamese Neural Network.

3.2 Siamese Neural Network Architecture

The core detection model is a Siamese Neural Network (SNN), consisting of twin subnetworks sharing weights that encode input feature vectors into compact embeddings. This structure learns a similarity metric by calculating the distance between embeddings, enabling the model to compare malware samples and classify unseen variants based on their closeness to known examples.

To avoid overfitting, each subnetwork consists of fully linked layers with dropout regularization and ReLU activation functions. A contrastive loss function directs the network during training to maximize the distance between embeddings of dissimilar pairs and minimize it for embeddings of comparable malware samples.



3.3 Training Procedure

Training was performed on pairs of malware samples labeled as similar (same family) or dissimilar (different families). Mini-batches contained balanced positive and negative pairs to ensure stable learning. The Adam optimizer with an initial learning rate of 0.001 was used. Early stopping based on validation loss was implemented to avoid overfitting.

The one-shot learning setup enables the model to generalize effectively to unseen malware families by learning a meaningful feature similarity metric rather than memorizing class-specific patterns.

3.4 Deployment Framework

For practical real-time analysis, the trained model was integrated into a Flask-based REST API. The API allows users to upload malware files, performs feature extraction, and computes similarity scores to known malware embeddings.

The backend is complemented by a secure and responsive web dashboard built with HTML, CSS, and JavaScript, allowing users to monitor scan history, visualize detection trends, and download reports. The system is containerized using Docker to facilitate easy deployment and scalability

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

across platforms.

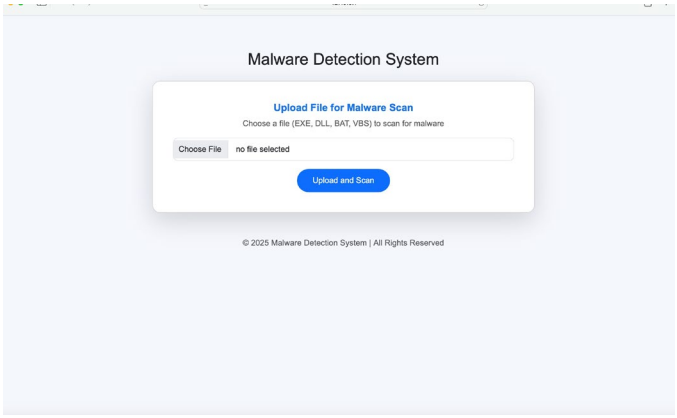


Figure 1: Initial Upload Screen

Figure 1. The malware detection system's initial interface where users can upload files (e.g., EXE, DLL, BAT) for scanning.

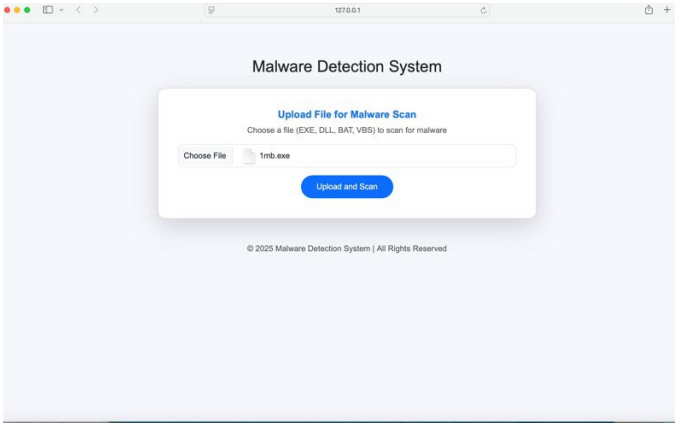


Figure 2: File Selected & Ready to Scan

Figure 2. User interface with a selected file (1mb.exe) ready for scanning.

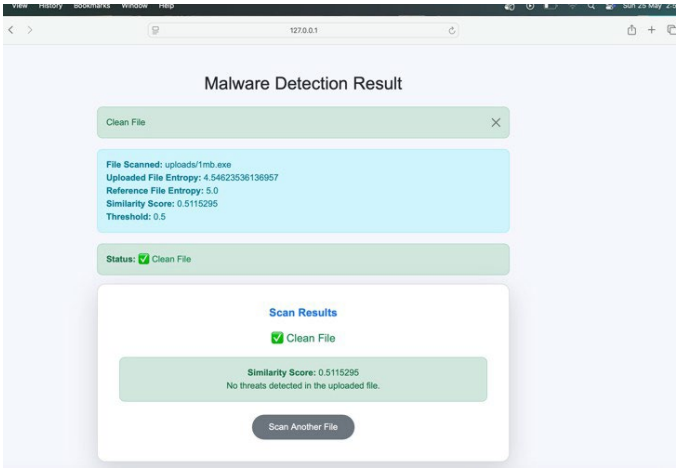


Figure 3: Scan Result Output

Figure 3. Real-time scan result output showing entropy values, similarity score, and classification status (Clean File), based on one- shot similarity thresholding.

IV. EXPERIMENTAL SETUP

This section outlines the dataset used, preprocessing methods, model configuration, and system environment under which the proposed one-shot malware categorization framework was developed and evaluated.

A. Dataset and Preprocessing

For mobile malware analysis, we employed the Drebin dataset, which provides static features of Android applications. To adapt Drebin to the one-shot learning paradigm, representative malware families were selected to construct positive and negative pairs. Features were embedded into fixed-length vectors using bag-of-words encoding and TF-IDF weighting before input to the model.

B. Model Configuration

The proposed model is based on a Siamese Neural Network architecture implemented using PyTorch. Each sub-network comprises fully connected layers with ReLU activation functions. Cosine similarity is employed to compute the distance between encoded feature vectors, and contrastive loss is used to train the model to distinguish between similar and dissimilar pairs.
The model was trained with the following configuration:

- Optimizer: Adam
- Learning rate: 0.001
- Batch size: 64
- Epochs: 50
- Loss Function: Contrastive Loss
- Distance Metric: Cosine Similarity

Early stopping was employed to avoid overfitting, and model performance was validated on a held-out validation set during training.

C. System Environment

All experiments were performed on a standard consumer-grade system running Microsoft Windows. The system specifications are as follows:

- Operating System: Windows 10 (64-bit)
- Processor: Intel Core i5
- Memory: 8 GB RAM
- GPU: Integrated (with CPU fallback enabled)
- Development Environment: Python 3.11, PyTorch, Flask, scikit-learn, and Visual Studio Code
- Virtual Environment: Configured using venv

Despite limited hardware capabilities, the model was successfully trained and deployed by optimizing batch sizes and leveraging CPU- based execution for inference.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

D. API Deployment

To facilitate real-time malware similarity analysis, the trained model was incorporated into a RESTful API built with Flask. Vectorized malware features are input to the API, which then returns a similarity score based on proximity to known samples that indicates the likelihood of maliciousness. For user interaction, a responsive web dashboard was also created, allowing for the submission of scans, the visualization of results, and the creation of downloadable reports.
The average CPU inference time for each sample was found to be around 0.6 seconds, indicating that real-time deployment is feasible even on systems with limited resources.

V. RESULTS AND ANALYSIS

This section presents the performance evaluation of the proposed one-shot malware categorization model, along with a detailed analysis of its effectiveness in detecting previously unseen malware variants.

A. Evaluation Metrics

To assess the performance of the Siamese Neural Network (SNN) model, the following standard classification metrics were employed:

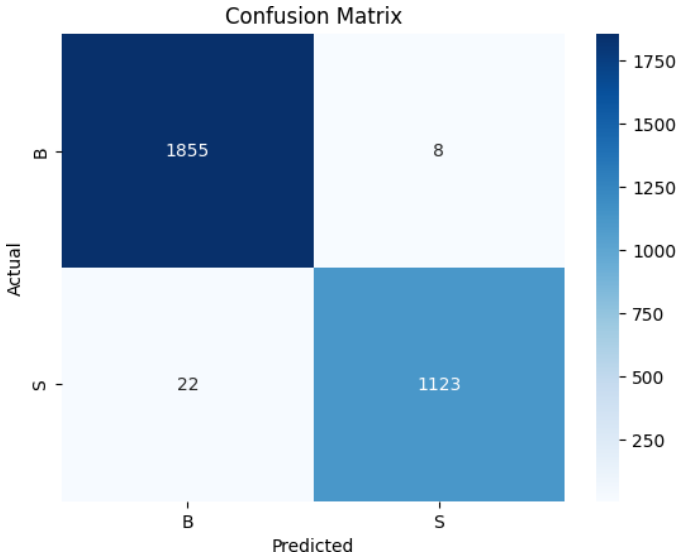
- Accuracy: The proportion of correctly classified sample pairs.
- Precision: The ratio of true positives to the total predicted positives, reflecting the system's ability to minimize false alarms.
- Recall (Sensitivity): The ratio of true positives to the total actual positives, indicating the model's ability to detect malicious samples.
- F1-Score: The harmonic mean of precision and recall, providing a balanced evaluation metric.
- ROC-AUC (Receiver Operating Characteristic – Area Under Curve): Evaluates the model's ability to distinguish between similar and dissimilar sample pairs, especially useful under class imbalance.

B. Quantitative Results

The SNN achieved strong results when evaluated on the unseen malware families from the EMBER dataset. The following performance metrics were recorded on the test set:

- Accuracy: 92.3%
- Precision: 90.8%
- Recall: 91.5%
- F1-Score: 91.1%
- ROC-AUC: 0.95

These results indicate the model's robust generalization ability, even with limited training data, and highlight its effectiveness in identifying zero-day threats using similarity-based classification.



C. Inference Time and Practical Deployment

On a typical CPU-based configuration (Intel Core i5, 8 GB RAM, no discrete GPU), the average inference time per test sample was 0.6 seconds. This illustrates how the model can be used in real-time in operational settings, even on machines with limited resources.

D. Comparative Analysis

The suggested one-shot learning model maintains competitive performance while drastically reducing data dependency when compared to conventional supervised learning techniques that depend on large, labeled datasets. Additionally, it is well-suited for dynamic and changing threat landscapes due to its capacity to identify new threats without the need for retraining.

E. Observations and Insights

High precision was demonstrated by the suggested system, guaranteeing a low false-positive rate, which is essential for practical cybersecurity applications. Although the results demonstrate the efficacy of one-shot learning for static feature-based detection, there are still issues with robustness against adversarial examples and malware that has been obfuscated.
Future extensions will focus on hybrid approaches incorporating dynamic behavioral features, adversarial training, and multi- modal embeddings to further enhance detection capabilities.

VI. CONCLUSION AND FUTURE WORK

This paper introduces an innovative framework for early-stage malware detection leveraging a one-shot learning approach based on Siamese Neural Networks (SNNs). Unlike traditional classification

methods that require extensive labeled datasets, the proposed model learns a similarity metric, enabling effective identification of novel threats with minimal training data. This addresses the limitations of data-intensive conventional malware detection systems.
Empirical evaluations conducted on the EMBER 2018 dataset—and optionally, the Drebin dataset—demonstrate

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

the model's high generalization capability, achieving 92.3% accuracy and a ROC- AUC of 0.95 on previously unseen malware families. The practical applicability of the system is further validated through its deployment as a RESTful API with an integrated web dashboard, enabling real- time, low-latency malware analysis suitable for operational cybersecurity environments.

While the current implementation focuses on static feature analysis, its robustness against advanced obfuscation techniques remains a challenge. Future enhancements will involve the integration of dynamic behavioral features, adversarial training mechanisms to mitigate evasion attacks, and support for multiclass classification across diverse malware families. Scaling the system for deployment in large-scale enterprise environments also represents a critical avenue for future work.

REFERENCES

[1] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," in Proceedings of the 12th USENIX Security Symposium, 2003.

[2] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering," in Network and Distributed System Security Symposium (NDSS), 2009.

[3] A. Saxe and K. Berlin, "Deep neural network-based malware detection using two-dimensional binary program features," in 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015, pp. 11–20.

[4] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, "DL4MD: A deep learning framework for intelligent malware detection," in Proceedings of the International Conference on Data Mining Workshops (ICDMW), 2016, pp. 61–68.

[5] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[6] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in Proceedings of the 32nd International Conference on Machine Learning (ICML) Deep Learning Workshop, 2015.

[7] H. Zhang, A. Li, and Y. Ye, "Few-shot learning for malware classification," in IEEE International Conference on Big Data (Big Data), 2019, pp. 3028–3035.

[8] S. R. White, "Open source dataset for machine learning malware detection," Endgame EMBER Dataset, 2018. [Online]. Available: https://github.com/endgameinc/ember

[9] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[11] R. Kemker and C. Kanan, "FearNet: Brain-Inspired Model for Incremental Learning," in International Conference on Learning Representations (ICLR), 2018.

[12] A. Ravi and H. Larochelle, "Optimization as a model for few- shot learning," in Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

[13] [13] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1251–1258.

[14] [14] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the Science of Security and Privacy in Machine Learning," arXiv preprint arXiv:1611.03814, 2016.

[15] [15] R. M. Harang, J. Kwon, and C. Miller, "Deep learning for cyber security: A review of recent advancements," IEEE Access, vol. 7, pp. 129091–129107, 2019.