Iraqi Journal for Computers and Informatics
Available online: https://ijci.uoitc.edu.iq/index.php/ijci
Volume 52, issue 1. 2026

Print ISSN: 2313-190X, Online ISSN: 2520-4912

Graph Neural Networks for Malware Classification: Comparing Graph-Structured and Sequence-Based Representations

Meer Twana Qadir [a], Saman Mirza Abdullah [b]

[a] *Department of Software Engineering, Koya University,Koya, Kurdistan Region, Iraq*

[b] *Department of Software Engineering, Koya University,Koya, Kurdistan Region, Iraq*

A R T I C L E I N F O

Keywords:

Malware, Dynamic API call sequences,Portable Excutable Files(PE Files), Graph Convolutional Networks (GCN),Graph Attention Networks (GAT)

A B S T R A C T

Malware detection is one of the most important cybersecurity issues because the traditional signature-based methods cannot resist polymorphic threats and obfuscated ones. This paper explores the dynamic API call sequences as behavioral characteristics and contrasts the two representation methods, integer-based feature encoding into the traditional machine learning models and graph-based models using Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT). Unlike prior studies, this paper conducts a systematic head-to-head comparison of these approachs and introduces a newly collected balanced dataset of 2,000 malware and 2,000 benign sampels, for this paper Two datasets were employed, one of which was a large public dataset with 42,797 malware and 1,079 benign samples, and the other was a novel developed dataset consisting of 2,000 malware and 2,000 benign samples that were collected according to this research by means of sandboxed execution. To facilitate 10-fold cross-validation, API calls were pre-encoded into fixed length sequences of integers and call graphs directed to allow fair evaluation. The findings indicate that ensemble and tree-based models achieved competitive results (≈92% on the public dataset and ≈90% on the novel dataset), but the graph-based ones were more accurate with GCN coming to 98.76% and GAT at 98.33%. Because graph neural networks can capture relational dependence and contextual patterns in API call behavior, they generate a richer representation and stronger categorization than integer encodings also the best feature of graph-based models is that they learn not only features but also the connectivity of API calls, which gives much richer and more accurate representation than integer-only encodings. Unlike prior studies, this work conducts a systematic head-to-head comparison of these approaches and introduces a newly collected balanced dataset of 2,000 malware and 2,000 benign samples.

1. INTRODUCTION

The increased growth of computer and Internet technologies has been accompanied by a similarly accelerated growth in malicious software (malware). Malware is any program that is deliberately meant to hinder, destroy, or obtain unauthorized access to computer systems[1]. Typical examples are viruses, worms, Trojans, ransomware, spyware and rootkits[2]. These malicious codes interfere with confidentiality, integrity and availability of data, which in most cases propagate through networks, removable devices or even in disguise as legitimate software[3]. In the last ten years, malware has become more sophisticated, with the adoption of more sophisticated evasion methods like code obfuscation, polymorphism, and encryption, and thus, has become more difficult to detect and analyze[4]. Signature refers to a sequence of bytes derived from unique data within a binary[5]. Modern technology's linked application ecosystem has many hazardous apps. Malicious applications, including viruses, are constantly generated and distributed to steal data from computers and networks, while firewalls and antivirus programs evolve to protect legitimate users and apps[6]. To

E-mail address:
meer.twana.qadir@gmail.com [a]
saman.mirza@koyauniversity.org [b]

Corresponding* : Saman Mirza Abdullah

Received 12 October 2025, Accepted 17 November 2025

DOI: 10.25195/ijci.v52i1680

address these limitations, a specification-based approach, fundamentally a behavior-oriented strategy, has been created. Researchers are using data mining and machine learning techniques, achieving outstanding results in malware detection and classification with enhanced accuracy ratings[7].Malware detection often involves two categories of features: static and dynamic. Static features refer to the inherent characteristics of the malware file, including its byte sequences, strings, and code structure, whereas dynamic features are obtained by monitoring the virus's activity during execution in a controlled environment[8].Static analysis is rapid but can be circumvented by obfuscation methods, whereas dynamic analysis provides superior detection accuracy but necessitates time and resources to run malware[9].The majority of the techniques used in malware detection and classification employ behavioral characteristics rather than structural features sequence of API calls can yield significant insights about the behavioral characteristics of malware[10].

The majority of researchers employed API calls to examine behavior based malware[11]. A key advantage of classifying malware based on a malicious program's system call behavior is the expedited attribution of its source and enhanced comprehension of its impacts[12]. Recognizing the category of malware associated with a malicious application provides administrators of an affected device with insights into resolution techniques for the attack[13]. Despite these joint efforts, hackers have escalated their utilization of malware, resulting in money losses, privacy violations, and service disruptions. Given the emergence of more sophisticated malware code, heuristic and signature-based detection systems are being criticized for their efficacy in identifying portal-like malware.

Contemporary and evolving malware frequently exploits authorized tools, particularly API calls designed for facilitating communication between programs and operating systems.Portable Executable (PE) files to address existing gaps. The analysis will focus on the invocation patterns of API calls, examining their sequence, frequency, and contextual features in both benign and malicious products; the classification modeling of constructs utilizing standard machine-learning techniques and advanced graph-based neural networks; the efficacy of the detection system; the resources linked to the proposed system's differentiation between benign and malicious products; and the comparative performance of both traditional methodologies and the proposed approach. The aims will focus on enhancing malware detection capabilities. Numerous studies have employed standard machine learning methodologies for malware detection and classification, with specific emphasis on API sequences, For this paper A comparison between two different approaches for malware detection, Firstly converting the API class to Integer and applying traditional Machine Learning algorithms and another by creating pattern graph and applying Graph Nural Network (GNN) techniques like Graph Convolutional Network (GCN) and Graph Attention Network (GAT).

The present work also highlights how graph-based encodings are more successful since they can encode the attribute features as well as the relational structure of execution traces, whereas integer-based encodings are unable to do so. Thereby, the rest of this paper is organized in the following way: Section 2 will conduct a review of related work, Section 3 will describe the proposed methodology, Section 4 will provide the experimental results and comparisons, and finally Section 5 will draw a conclusion regarding further research directions.

1.1 Problemstament and Limitations

Although there is a current development in malware detection, malicious messages can still be used to spread malicious code, rob valuable information, or gain unauthorized access to valuable resources of the systems. The difference between legitimate and malicious API patterns tends to overlap to develop a high rate of false positives, and the enduring development of malware only weakens the efficiency of the conventional detection techniques. Moreover, the majority of current methodologies are not able to record the contextual order of API calls, and the lack of high quality, labeled data sets remains a significant issue to successful malware analysis.

1.2 Contribution

Main contributions to this work are First,defining a balanced set of dynamic API sequences calls in order to make fair comparison between malware and benign samples. Second, performing a systematic analysis of the capabilities of integer-encoded machine learning models and graph-based neural architectures, which includes a rigorous comparison of them. Third, introduceing a comprehensive analysis of both publicly and recently built datasets with the same experimental conditions in place and make sure that the results are reliable and reproducible.

2. RELATED WORKS AND BACKGROUND

The quick improvement in computer and Internet technologies is simultaneously associated with an enormous rise in malicious software (malware). Malware, including viruses, Trojans, and worms, evolved rapidly and emerged as the most significant threat to internet. Malicious software executes harmful actions on the computer system. Typically, malware takes control of computer systems through modifying or disturbing the standard execution flow of processes[14]. As per AV-TEST research only at 2024 approximately 450,000 new malware variants were created, with more than 76 percent targeting Windows operating systems. Additionally, more than 43 million 0-day malware instances were documented in between 2020-2024[15]. Software and malware are collections of data, instructions, and functions designed for specific goals, occasionally distributed in an executable binary format[16]. Anti-malware providers typically apply signature-based approaches to recognize known threats, along with algorithmic detection techniques[17]. Malware, also known as malicious software is a heterogeneous collection of harmful programs that are maliciously designed to penetrate, interrupt, or gain unauthorized access to computer networks and systems[18]. Malicious entities manifest in various forms, each with distinct payload attributes and transmission methods, including viruses, worms, trojans, ransomware, spyware, adware,

rootkits, keyloggers, fileless malware, and botnets. One of the first forms of malicious software is known as a virus, and it is possible for viruses to attach itself to legitimate executables or documents. They are able to replicate themselves and propagate through removable storage devices, email attachments, and file-sharing networks when the infected file is executed. In current viruses, the payloads that are used to infect computers frequently cause files to become corrupted, deactivate antivirus applications, or create further infestations[19]. Worms on the other hand spread independently in networks and do not require human intervention. They use the weaknesses of network protocols or unpatched software to spread as quickly as possible, using bandwidth and sometimes adding auxiliary code like ransomware or spyware[4]. The purpose of Trojans is to trick victims into executing them by seeming to be programs that are either innocuous or helpful. After they have been installed, they will either elevate privileges, deactivate security measures, or develop backdoors, all of which will allow them to achieve further compromise. In order to provide an example, banking trojans are able to record financial transactions, whilst Remote Access Trojans (RATs) give attackers the ability to exercise complete control over the computers that they have infected[20].Spyware is able to monitor user actions and collect browsing history, user logins, financial information and in some cases web camera or microphone feeds. Spyware used in commercial aspects has increasingly been used as a weapon of surveillance. Adware, even though it might be considered less harmful, propagates invasive advertisements, slows down the working of the system, and at times, also acts as a vehicle to more malicious software[21].Rootkits operate on a lower level as subsystem level and are embedded in system kernels, boot loaders or firmware to hide the existence of malicious processes and provide persistent existence. They are especially hard to eliminate due to their ability to evade detection. Keyloggers represent a more narrower methodology, where the keystrokes are captured to steal sensitive data like online banking details, passwords or encryption keys[22].Equally, botnets form armies of infected computers under the command of the attacker. They are typically used to initiate massive Distributed Denial-of-Service (DDoS) attacks, spam, harvest data, or mine cryptocurrency to cause massive economic and security harm[23], [24].In addition to these conventional types, new malware is incorporating sophisticated and hybrid types more and more. Polymorphic malware is able to change its code signature dynamically with each infection making it virtually useless to detect these infections by signature. Metamorphic malware also rewrites its internal code logic without losing functionality and therefore completely avoiding traditional pattern-matching defenses . Blended threats are those that employ more than one method to propagate, including worms that drop ransomware payloads, and to be as destructive as possible. These developments highlight the flexibility and complexity of the modern cyber threats[25] .Of all the types, malware in the form of Portable Executable (PE) has become one of the most serious issues. PE format is the Windows executable standard (.exe, .dll) and, therefore, an all-the-more appealing target to enemies[26]. PE malware modifies the PE header and injects malicious code into common sections including (.text or .data). Some of them use section padding, code caves, or even develop completely new sections to hide payloads. Import Address Table (IAT) hooking, runtime packing, and control flow obfuscation techniques are used to ensure the malware evades the process of a static analysis and to make it stay operable even after the repeated cycles of packing or encryption[27]. Since the windows systems are designed to trust PE files, attackers use this trust to avoid the defenses so that the malware can seem legitimate when loaded and run[28].PE-based malware can only be detected using advanced methods that are not just a signature scanner. These are behavioral analysis of suspicious API call sequences, graph based structural anomaly detection and machine learning classifiers that have been trained to detect malicious execution patterns. In recent research, it is noted that dynamic monitoring, memory forensics, and AI-driven systems can be integrated and used to mitigate zero-day PE threats before they spread. With the increasing complexity of these attacks, it is still critical to keep researching PE file detection, as these files are the main path of the modern, highly evasive families of malware[29].

2.1 Malware Analysis

Malware analysis is an important field in the field of cybersecurity and it is usually categorized into two major methodologies, which include static analysis and dynamic analysis[30]. In the static analysis, malicious code is analyzed without executing the code, and methods like disassembly, decomplication and binary inspection are used to find the code structures, embedded strings, control-flow diagrams, and possible signs of compromise[31]. This methodology proves to be especially effective when it comes to the detection of familiar malware families and the detection of simple obfuscation techniques. But the former cannot work with polymorphic malware or with heavily packed and encrypted binaries. These kinds of variants may hide their actual functionality behind purely-static examination, and this greatly diminishes the usefulness of purely-static detection plans[32]. Dynamic analysis differs and it entails malware being executed in a controlled environment, usually on virtualized sandboxes or emulation infrastructure, in order to monitor how it behaves under execution. Through the execution of the program, the analysts can obtain behavioral artifacts like file system modifications, alterations in registry, network traffic patterns, and attempts to inject processes and use of the system resources[14]. A sequence of Application Programming Interface (API) calls made by the malware is one of the most enlightening products of the dynamic analysis[13]. These API calls are the most common method by which malware communicates with the underlying operating system and with external resources and hence they can provide clues about the intent of its operation. As an example, API calls involving file access, memory management and network access may reveal stealthy malicious behavior which cannot be detected by static analysis[33]. Figure 1 shows the broad outline of malware analysis, comparing malware analysis using both the static and dynamic methods. In this design, the surface-level but efficient inspection is offered by the static analysis whereas a deeper look at the activities in the run time is possible with the help of dynamic analysis[34]. The figure underscores the importance of dynamic analysis to capture data at the execution-level, where API call monitoring comes out as an especially powerful tool to profile malware behavior. The approach is based on the observation directly, With the help of systematic extraction of dynamic API

call sequences allows to create the behavioral representations that can be consumed by classical machine learning algorithms as well as by more graph-based neural models[35]. It makes a direct link between the backgrounds of malware analysis and the subject of our study. However, there are no limitations to dynamic analysis. Malware writers take action to develop anti-analysis methods to circumvent detection, e.g. by deferring the execution of malicious code, detecting virtualized or sandboxed systems, or by requiring certain user actions before dangerous activities can be executed. These methods can be used to greatly diminish the visibility of malicious actions in automated settings that can result in incomplete behavioral traces. In addition, dynamic analysis comes with computational overhead and scaling issues, as running thousands of samples with dynamic analysis consumes a lot of infrastructure and time. These shortcomings encourage the development of smarter methods of detection. In recent studies, understanding the machine learning approach and, more precisely, the models that could analyze the sequence of API calls as a structured data, have become more popular[36]. These technical innovations counteract the increasing complexity of evasion strategies used by contemporary virus developers. The ongoing advancement in malware creation requires constant enhancement of analytical techniques to provide effective identification of new cyber threats[37]. In particular, graph-based learning approaches have shown promise in modeling the relationships and dependencies among API calls, thereby addressing some of the shortcomings of traditional

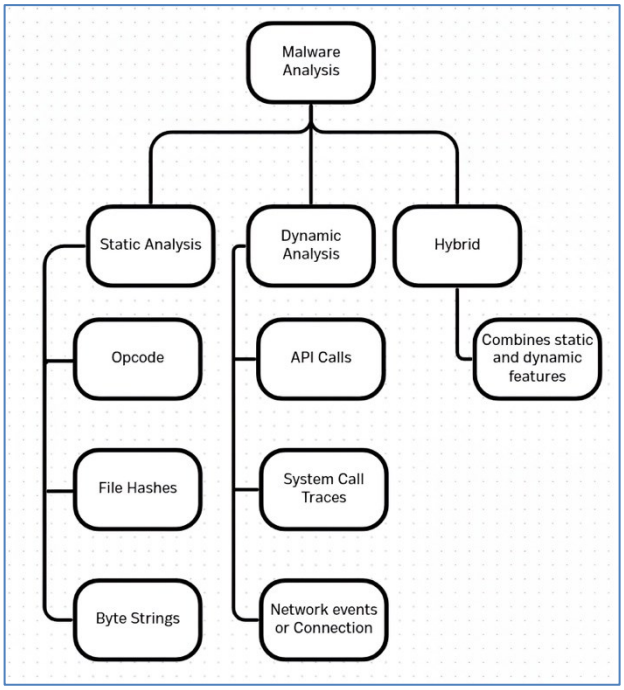

Fig 1*:malware analysis types*

sequence-based methods.

2.2 Literature Review

In 2015 a novel research had been done using DNA sequence alignment algorithms like MSA (Multiple Sequence Alignment) and LSC (Longest Common Subsequence) having a dataset which had 25,807 malware data having the accuracy of (99.8%)[38].in 2016 fuzzy hashing was employed for classifying a dataset of 2600 malware's which achieved the accuracy of (96.33%)[39].in 2017 a new research had been done using four classifiers (SVM, KNN, Random Forest, Multilayer Perceptron) having a dataset which contained 23,080 samples archiving the accuracy of (99.06%)[40] in 2019 researches used (Minkowski Distance, Cosine Similarity, Longest Common Subsequence) having a dataset of 78,568 which had 61,354 malicious API and 17,214 benign API gaining the accuracy of (95%)[41].in 2020 another research has been done using serval classifiers like SVM ,Navie Bayes , Random Forest having a dataset consist of 94 API calls That have been monitored, and using PCA (principle component analysis) as a dimension reduction tool studying both API-CFM(API Call Frequency mining) and API-CTMM (API Call transition matrix mining) , API-CTM accuracy was 76.16% and API-CTMM accuracy was 95.23%[42].In 2021, a research addresses the difficulty of analyzing API call sequences for malware detection and categorization. It advocates employing Natural Language Processing (NLP) techniques such as word embedding and Latent Dirichlet Allocation to enhance virus detection. The research employs two datasets comprising over 13,000 samples. The findings indicate that NLP enhances malware detection accuracy; however, further data is required for comprehensive analysis[43]. In 2022, a study employed TF-IDF (Term Frequency-Inverse Document Frequency) and Word2Vec for the statistical and contextual weighting of API bigrams during feature extraction. It utilized Ant-Colony Optimization (ACO) for feature transformation to generate a new behaviorally weighted matrix and implemented a Multi-

Layer Perceptron (MLP) with three layers (sizes 50, 100, 50) and a tanh activation function for classification. The research used the Brazilian-Malware Dataset for malware and the benign Dataset, which had 50,221 malware samples and 21,145 benign samples achieving an accuracy of 99.7%, sensitivity of 66.4%, and specificity of 99.92%[44]. In 2023, a temporal process graph (TPG) was employed to model the inter-process behavior for detecting API sequences associated with malware, utilizing a dataset comprising 14,657 samples. The results indicate that this method demonstrates a 3.47% enhancement in precision and a 4.78% enhancement in recall compared to Node2Vec when using a basic k-NN model[45]. Another research 2023, research achieved an F2 score of 76.21%, Sensitivity of 91.34%, and Specificity of 99.27% utilizing Random Forest Classifier (RFC) and LeNN as classifiers for detecting polymorphic malware through a sequence of API calls, employing Bitdefender's Cyber Threat Intelligence Lab dataset[46]. In 2024, a paper identified the shortcomings of current dynamic malware detection techniques that predominantly depend on API call sequences, neglecting the relevant run-time parameters. It achieved an accuracy of 93.9% and an enhancement in F1-score ranging from 1.2% to 6.5% compared to state-of-the-art methods that incorporate both API calls and run-time parameters, applying two public datasets[47].further more detail of other papers have been shown in table 1.

***Table I**:Related Work*

Ref	Year	Classification Techniques	Dataset detail	Accuracy
[48]	2017	CNN	2000 malware, 500 benign	93%
[49]	2018	MLP	41,265 malware, 10920 benign	98.6%
[50]	2022	ANN	22120 malware, 22187 benign	97.31%
[36]	2022	CNN,LSTM,GCN	9443 malware ,9185 benign	Accuracy 1: 92% Accuracy 2: 93% Accuracy 3: 80%
[37]	2023	ANN	3000 malware and 2000 benign	97.91%
[51]	2023	Logistic regression	First dataset 1079 benign files and 1079 malicious. Second dataset 42797 malware and 1079 benign	Accuracy data set 1: 83% Accuracy dataset 2 : 98%
[52]	2024	BiLSTM, BiGRU	First dataset 1285 benign files and 1285 malicious. Second dataset 42797 malware and 1079 benign	Over 90% for both models
[35]	2025	GCN,GIN,GAT,SE	512 Malware ,319 benign	Accuracy 1: 83% Accuracy 2: 82% Accuracy 3: 83% Accuracy 4: 80%
[53]	2025	GAT	1000 benign , 1000 malware	99%
[54]	2025	DGNN	Not specified	97%
[55]	2025	CNN	13422 malwares ,8634 benign	98.36%

As its clear the focus area of malware detection using AI based models had focused on text and numerical based approaches mostly till before 2020 and after that from 2020 to 2025 the classification approach had been shifted and more focused on graph based approach malware detection and the most focused techniques in graph based approaches are Graph Neural Network techniques (GNNs), in this paper two types of GNN are focused on and compared to integer based approaches to see which one gives a better result.

3. METHODOLOGY

3.1 Research Design

The design of this research as shown is figure 2,involves a comparative experimental design, which will be organized to compare the difference in performance between the approaches of detecting malware based on integers and based on graphs. The research is based on a systematic procedure where data collection and preprocessing is followed by the encoding of API call sequences in two different forms i.e. integer features and graph representations. The representations are compared against the set of models that represent them under the same experimental conditions. The reliability and reproducibility of the results are guaranteed by ten-fold cross-validation. The design enables objective comparisons and demonstrates the impact of relational graph learning as compared to flat integer encoding on the malware classification accuracy.

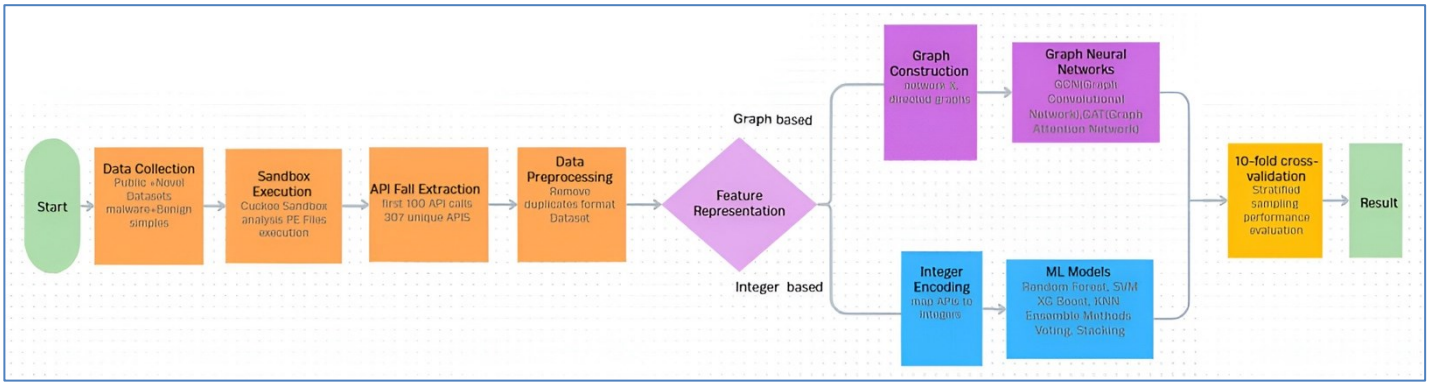

***Fig 2**: Methodology Processes*

3.2. Dataset

For this Research two datasets will be used first dataset contributes to an IEEE research paper investigation of malware detection and classification with Deep Learning techniques. It has 42,797 malware API call sequences and 1,079 benign API call sequences. Each API call sequence consists of the initial 100 unique consecutive API calls linked to the parent process, derived from the calls sections of Cuckoo Sandbox reports[56]. Because public dataset is highly skewed and could bias results, which is why the balanced dataset was created for this research which contains 2000 benign and 2000 malware samples.

3.3. Data Collection

Due to lack of public domain PE dynamic malware analysis dataset for training and evaluating a dataset was developed malware samples were collected from malware bazaar [57]and virus share [58] and benign from portablepps.com[59]. and windows 7 active directory.at the beginning for the data collection an environment must be setup in order to make Cuckoo sandbox to work for that prepose you need a server and a client, Client to run PE files and server is to analyze the PE files. For that a Server environment using Virtual Box as a virtual device had been used to install Ubuntu server 16.04 version on it and, and a windows 7 was used as client pc. It required around 6 months of continues and daily working to run around 5000 files and end with 4000 Cuckoo Json Report files.

3.4. Data Preprocessing

After Data collection two supplementary preprocessing procedures were execute firstly it was determined the initial 100 non-consecutive repeated API queries to prevent tracking loops and also, it's time consuming and to avoid complexity. Secondly In malware detection jobs, it is crucial to identify malicious patterns right away; therefore, the sequences were retrieved just from the parent process. Finaly generated a dataset in which the initial column comprises the MD5 hash of the sample. Next to it the column indicated the sample label, with 0 representing benign and 1 denoting malware. The subsequent 100 columns comprise ordinal categorical values denoting the API call sequence of the sample

3.5. Integer based approach

First step of integer-based approach was compiling a list of distinct API calls, taking into account all samples, and thereafter assigned

each API call name a unique integer identifier corresponding to its index in the list A total of 307 unique API calls were detected. Then it has been directly fed into Machine learning algorithms, testing various machine learning models of malware classification over dynamic API call sequence as represented by 100 numeric features per sample. Individual models were optimized Random Forest (200 trees, max depth 15), SVM with RBF (C=10), XGBoost (200 estimators, max depth 6), Gradient Boosting (200 estimators, max depth 5) logistic regression (max iter 1000) and KNN (7 neighbors, distance-weighted). As hyperparameters, the number of neighbors in case of KNN, the SVM parameters, the depth of the tree, and the learning rates of the ensemble models were varied, yet the specified setting performed the best. With regard to ensemble methods, a soft Voting classifier was employed by first summing outputs of Random Forest, SVM, XGBoost, Gradient Boosting, and Logistic Regression, with each model contributing based on a particular weight; the class with the highest weighted probability was selected as the final prediction. A Stacking ensemble was also used where base model outputs were used as features in a Logistic Regression meta learner. The performance of all models was estimated using 10-fold stratified cross-validation, and the criteria of accuracy, precision, recall, F1-score and TPR and FPR classification as well as ROC-AUC curves and confusion matrices, were also calculated, providing a comparative assessment of individual and combined classifiers as shown in table 2.

***Table II**:Integer based parameters*

Model	Key Parameters / Configuration
Random Forest	n_estimators=200, max_depth=15, min_samples_split=5, min_samples_leaf=2, random_state=42
SVM (RBF Kernel)	kernel='rbf', C=10, gamma='scale', probability=True, random_state=42
XGBoost	n_estimators=200, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, eval_metric='logloss', random_state=42
Gradient Boosting	n_estimators=200, learning_rate=0.1, max_depth=5, random_state=42
Logistic Regression	C=1.0, max_iter=1000, random_state=42
K-Nearest Neighbors	n_neighbors=7, weights='distance', metric='euclidean'
Voting Ensemble	Soft voting, weights= [3,2,3,2,1] (RF, SVM, XGB, GB, LR]
Stacking Ensemble	Base estimators: RF, SVM, XGB, GB; final estimator: Logistic Regression; cv=5

3.6. Graph Based Approch

The paper proposes a graph-based malware detection which incorporates two Graph Nural Network (GNN) techniques, Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT) on dynamic API call sequences. First step was creating directed graphs of the API calls call sequences of every malware and benign example in our dataset. The graph was produced using NetworkX with nodes representing distinct API calls and directed edges representing the time sequence within a sequence of consecutive API calls. The graphs were visually plotted using a spring layout to define the edges and saved as PNG graphics files with the edges colored as gray and nodes in the color of sky blue as shown in figure 3. Further, the graphs were converted into the PyTorch Geometric Data, which is used in the graph-based neural networks. The unique identity of nodes was encoded in the form of one-hot vectors, and edges were encoded in edge index tensor representing the source-target node pairs. Last, the generated graphs were automatically saved in (.pt) format, which could be loaded efficiently in graph neural networks. This pipeline enables visual examination of the protocol of API calls as well as direct use of the graphs as a component of the graph-based malware classification models. Two neural architecture permutations are implemented (a GCN classifier composed of two convolutional layers (128 hidden channels, 0.3 dropout, batch normalization, and Xavier initialization) and a GAT classifier with multi-head attention (4 heads in the first layer, 128 hidden channels, and 0.3 dropout)). Both models have employed 10-fold stratified cross-validation, an Adam optimizer (learning rate 0.005, weight decay 5e-4), and trained over 15 epochs with a batch size of 256, with extensive evaluation in the form of accuracy, AUC, precision, recall, F1-score, ROC curves and confusion matrices, and thus represent a very good method to detect malware using the behavioral pattern analysis based in API call sequences further detail shown in table 3 .

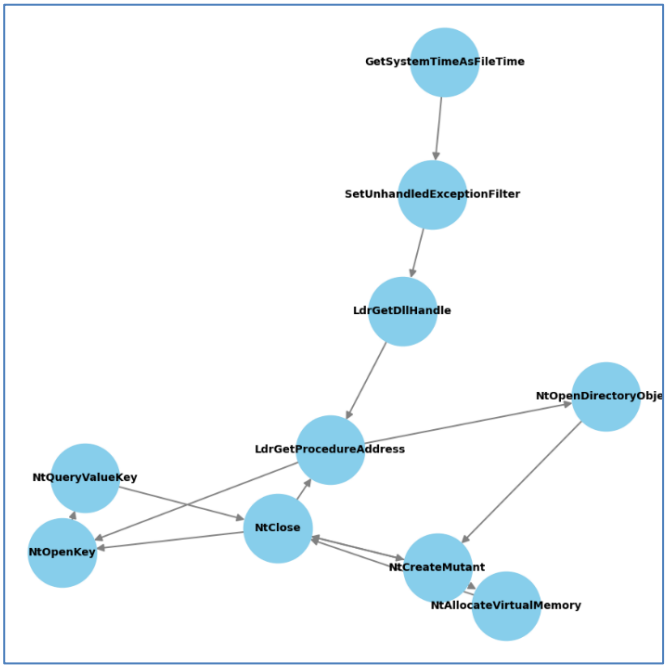

***Fig 3**:Graph Based Example*

***Table III**:Graph based parameters*

Parameter	GCN Classifier	GAT Classifier
Input Channels	Graphs	Graphs
Hidden Channels	128	128
Output Channels	2 (binary classification)	2 (binary classification)
Layers	2 GCN layers	2 GAT layers
Attention Heads	N/A	4 (first layer), 1 (second layer)
Dropout Rate	0.3	0.3
Learning Rate	0.005	0.005
Weight Decay	5.00E-04	5.00E-04
Epochs	15	15
Batch Size	256	256
Optimizer	Adam	Adam
Initialization	Xavier Uniform	Xavier Uniform
Cross-Validation	10-fold Stratified	10-fold Stratified

4. Results And Discussion

4.1 Results of public dataset

The graph-based models achieved near-perfect performance, with the graph convolutional network (GCN) standing out as the top model. GCN reached an average accuracy of 98.71% (±0.25), precision 0.99, recall 0.9898, F1-score 0.99, and AUC 0.97, while keeping errors extremely low. This shows not only its capability to adequately generalize on a bigger set-up but also more reliability in minimizing the false alerts and the number of unsuccessfully-detected targets. The graph attention network (GAT) also produced excellent results, achieving recall 0.99 along with high precision and F1-scores that were only slightly below GCN. The graphs had extremely low variance in cross-validation thus confirming their stability and consistency. Their higher efficiency is a clear indication of the benefit of using the relational structure of the API call graphs. The integer-based models fared better than on the smaller dataset but still distinctly worse than on the greater one. The stacking ensemble was the strongest in this group, with an average accuracy of 92.36% (±0.81), precision 0.99, recall 0.92, F1-score 0.95, and AUC 0.95. Nevertheless, it still demonstrated a substantial number of errors with 925 false negatives and 180 false positives, which would be a serious problem during real-life usage where reducing errors is vital. The Random Forest and XGBoost also performed well in terms of mean accuracy, of 92%, but at the cost of a relative increase in error. Logistic regression was the most ineffective at 88.07% which is due to its inability to deal with the complexity of the API call data. Albeit ensemble and tree-based models had better results when compared to shallow models, it was clear that their performance was not as perfect as those of graph neural models. Even though reference [53] provides a slightly better accuracy of 99%, we evaluated the same in much more challenging and realistic conditions. The mentioned experiment utilized a small balanced dataset of merely 2 000 samples and seems to have presented the single-run results, whereas our experiments had 10-fold cross-validation on a novel curated balanced dataset and a larger public dataset of more than 43 000 samples. The reason behind this multi-fold design is that it guarantees the reliability of statistics and prevents random changes in performance. On top of this, we are accurate to 0.99, which shows very low false-positive rates, which were not presented in reference [53]. Thus, even though there is no significant difference in nominal accuracy, our methodology is stronger, statistically more reliable, and practically stronger when it comes to malware that is going to be applied in real-life scenarios,All the result is shown in table 4 . And the ROC curve is shown in figure 4 .

***Table IV**:Results of public dataset*

Model	Highest Accuracy	Precision	Recall	F1-Score	AUC	Mean Accuracy
GCN	**98.76%**	99.62%	98.98%	99.30%	97.09%	98.71% ± 0.25%
GAT	**98.33%**	99.51%	99.11%	99.31%	93.88%	98.22% ± 0.32%
Stacking Ensemble	**92.43%**	99.54%	92.43%	95.85%	95.67%	92.36% ± 0.81%
XGBoost	**92.15%**	99.41%	92.15%	95.63%	95.34%	92.08% ± 0.87%
Random Forest	**92.05%**	99.50%	92.05%	95.61%	95.12%	91.98% ± 0.89%
Voting Ensemble	**91.68%**	99.43%	91.68%	95.40%	94.45%	91.61% ± 0.96%
SVM	**91.08%**	99.32%	91.08%	95.04%	93.78%	90.95% ± 1.34%
KNN	**90.04%**	99.10%	90.04%	94.34%	91.56%	89.97% ± 1.56%
Gradient Boosting	**89.12%**	98.95%	89.12%	93.79%	90.67%	89.05% ± 1.78%
Logistic Regression	**88.07%**	98.82%	88.07%	93.12%	89.34%	87.98% ± 1.98%

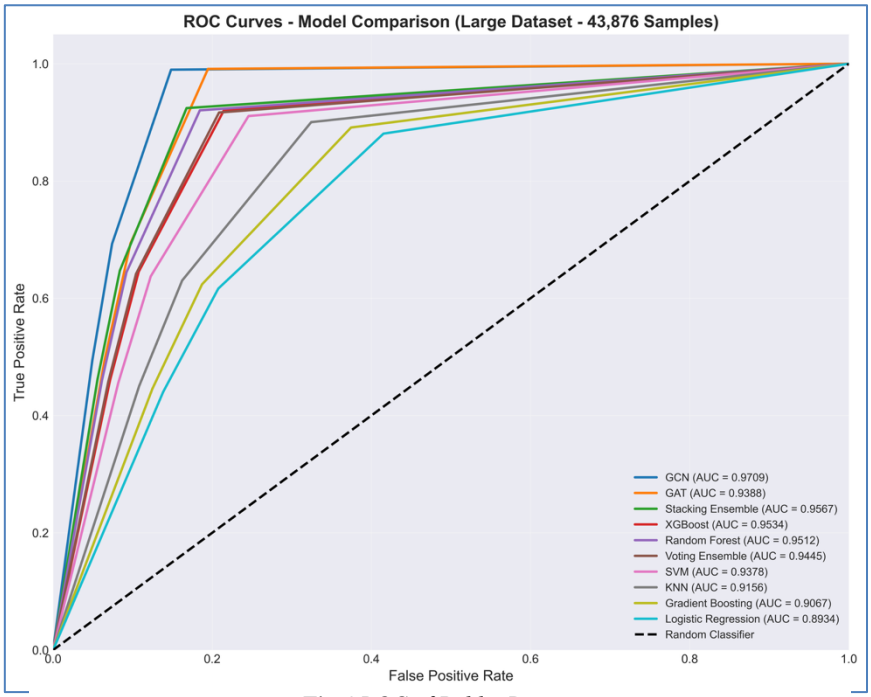

Fig 4 *ROC of Public Dataset*

4.2 Results of Novel dataset

On the 4000-sample dataset the graph-based models also outperformed all others by a significant margin. The graph convolutional network (GCN) achieved an average accuracy of 96.98% (±0.89), precision 0.97, recall 0.96, F1-score 0.96, and AUC 0.97The graph attention network (GAT) also showed strong results, achieving precision 0.96 and recall 0.94, though it trailed GCN slightly in overall performance. Both models demonstrated their low and consistent error rate with the folds, thus proving resilient even on a smaller dataset. The integer-based models were significantly less efficient by comparison. The stacking ensemble performed best, with an average accuracy of 90.36% (±1.36), precision 0.90, recall 0.90, F1-score 0.90, and AUC 0.93. Graph-based approaches like XGBoost and random forest performed second best with an average accuracy of 90.15 percent and 90.05 percent respectively, although they could not reduce false positives and false negatives as much as the graph-based variants. Simpler classifiers such as SVM (88.23%), KNN (87.45%), gradient boosting (86.45%), and logistic regression (85.92%) lagged further behind, with much lower recall and F1-scores, indicating higher misclassification of malicious samples. Ensemble and tree-based algorithms did not perform as well as graph-based representation in utilizing relational dependencies in sequence of API calls, All the result is shown in table 5. And the ROC curve is shown in figure 5.

Table V*:Results of novel dataset*

Model	Highest Accuracy	Precision	Recall	F1-Score	AUC	Mean Accuracy
GCN	97.00%	97.49%	96.50%	96.99%	97.84%	96.98% ± 0.89%
GAT	95.50%	96.47%	94.50%	95.47%	97.19%	95.88% ± 1.24%
Stacking Ensemble	90.58%	90.71%	90.43%	90.23%	93.67%	90.36% ± 1.36%
XGBoost	90.30%	90.45%	90.15%	89.98%	93.34%	90.08% ± 1.42%
Random Forest	90.20%	90.34%	90.05%	89.84%	93.12%	89.98% ± 1.45%
Voting Ensemble	89.85%	89.98%	89.68%	89.47%	92.45%	89.61% ± 1.61%
SVM	88.23%	88.42%	88.08%	87.89%	91.78%	87.95% ± 1.89%
KNN	87.45%	87.87%	87.04%	86.78%	89.56%	86.97% ± 2.01%
Gradient Boosting	86.45%	86.78%	86.12%	85.89%	88.67%	86.05% ± 2.23%
Logistic Regression	85.50%	85.92%	85.07%	84.76%	87.34%	84.98% ± 2.45%

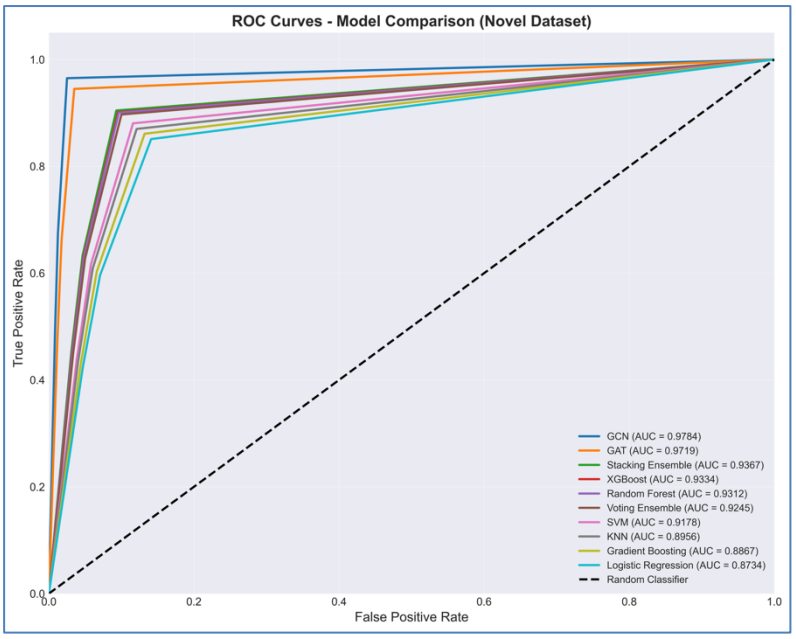

***Fig 4**:ROC of novel dataset*

4.3. Discussion

The results demonstrate that graph-based neural networks (GCN, GAT) substantially outperform both ensemble and traditional classifiers in this classification task. Both GCN and GAT attained near-optimal precision and recall on all datasets, whereas the rates of other methods are significantly less. With a very high specificity and low false-positive ratios, they indicate that they distinguish the classes far better than customary models. In contrast, even the best non-graph model (stacking ensemble) showed lower overall accuracy and higher error rates, suggesting that it could not capture certain complex relationships that the graph models exploit.Ensemble and tree-based methods (stacking, random forest, XGBoost) did improve upon simple models, confirming that combining learners or allowing nonlinear splits can enhance performance. The stacking ensemble, was capable of slightly outperforming single classifiers but only achieved a performance of approximately 0.90 on the smaller set. The simpler model types used, such as logistic regression and KNN, performed much worse probably because they cannot incorporate complex interactions between features or even represent relational structure. The above results suggest that feature-based learning and ensembles, although they carry some advantage, are not adequate as compared to the strategies that take advantage of the latent network structure of the data. Notably, the better performance of GCN and GAT was also replicated between the two datasets. All models saw higher accuracy on the larger public dataset (likely due to more training examples or clearer patterns), yet the relative ranking of methods remained unchanged. The small standard deviations in the cross-validation highlight that the high performance of GCN/GAT is not due to chance, but are robust. In general, the outputs have indicated the usefulness of the graph or relational information in this problem domain. Future efforts ought therefore to focus on the graph-based representations or add network properties as such kind of representation provide significantly higher classification accuracy compared to orthodox ways. In short, the performance of graph-based methods is superior because it can consider both the features and the underlying connectivity between samples. The use of message passing and neighborhood, GCN and GAT can learn more complex dependencies that cannot be learned by integer-based or purely feature-driven models, resulting in their significantly superior accuracy and robustness.Thus, the real power of graph-based neural networks lies in their two-fold nature: they capture the features of nodes, and at the same time capture the structure of connectivity, which provides them with a clear advantage over the classic integer-based algorithms.

5. CONCLUSION

This paper offered a comparative analysis methodically on the Malware detection with the use of dynamic API call sequences, using integer-based feature representation versus graph based feature representation. In both the large public dataset and the balanced novel dataset, both graph neural networks (GCN and GAT) achieved better results than the traditional classifiers. Although methods like ensembles and tree-based models which used integer-based computation showed satisfactory results, they did not achieve the contextual and relational relationships between API calls.The most important aspect that makes graph-based methods superior is their representational ability. GCN and GAT can learn the multi-hop relationships, structural dependencies, and behavioral motifs, characteristic of malicious logic, by learning API calls as nodes, and the order of execution of API calls as edges. GCN combines the information on a neighborhood level to indicate the subgraph level features, whereas GAT uses attention mechanisms to rank the most relevant API interactions. Such mechanisms enable GNNs to be more effective in terms of generalization across a greater malware

family, decrease false positives, and be highly specific.To summarize, graph-based neural networks represent a new paradigm of malware detection, which is incredibly advantageous compared to flat and integer-based representations. This increases their accuracy, strength, and interpretability because of their capacity to learn on the basis of the relational structure of execution traces. This needs to be enhanced in future work by incorporating runtime parameters, graph embeddings, and explainable AI methods to enhance the overall performance of the detection and its ability to survive over time as the threats evolve. Comprehensively, the results of this study show conclusively that graph-based models are the best due to their combination of data characteristics and structural connectivity making them superior in terms of their robustness, accuracy, and generalization features relative to integer-based models.

5.1 Future work

Future works will aim at generalizing this research to zero-day malware detection, an area where most models do not perform well because of their adherence to known patterns. Areas of possible future work would be adopting few-shot and meta-learning to enable better generalizations to unseen threats, using graph embeddings and transformer-based sequence models to consider deeper context, and using runtime parameters to represent features more richly. Further, generative models and adversarial training can reproduce possible zero-day behaviors, and enhance resilience. The mentioned strategies, conjoined with explainable AI, will help develop adaptive and interpretable detection systems that can help detect zero-day attacks in real-time.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding

None

Acknowledgment

None

References

[1] S. M. Abdulalla, L. M. Kiah, and O. Zakaria, “A biological model to improve PE malware detection: Review”.

[2] X. Ling *et al.*, “Adversarial Attacks against Windows PE Malware Detection: A Survey of the State-of-the-Art,” *Comput. Secur.*, vol. 128, p. 103134, May 2023, doi: 10.1016/j.cose.2023.103134.

[3] “A Survey on Reinforcement Learning-Driven Adversarial Sample Generation for PE Malware.” Accessed: Sept. 27, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/12/2422

[4] D. Laka, “Malware: Types, Analysis and Classification,” Jan. 14, 2022, *Social Science Research Network, Rochester, NY*: 4036836. doi: 10.2139/ssrn.4036836.

[5] “A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Jan. 05, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9121260

[6] D. Gibert, C. Mateu, and J. Planes, “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges,” *J. Netw. Comput. Appl.*, vol. 153, p. 102526, Mar. 2020, doi: 10.1016/j.jnca.2019.102526.

[7] D. Čeponis and N. Goranin, “Evaluation of Deep Learning Methods Efficiency for Malicious and Benign System Calls Classification on the AWSCTD,” *Secur. Commun. Netw.*, vol. 2019, no. 1, p. 2317976, 2019, doi: 10.1155/2019/2317976.

[8] T. Ige, C. Kiekintveld, A. Piplai, A. Wagler, O. Kolade, and B. Hafiz Matti, “Towards an in-Depth Evaluation of the Performance, Suitability and Plausibility of Few-Shot Meta Transfer Learning on An Unknown Out-of-Distribution Cyber-attack Detection,” in *2024 International Symposium on Networks, Computers and Communications (ISNCC)*, Oct. 2024, pp. 1–6. doi: 10.1109/ISNCC62547.2024.10759010.

[9] T. Ige, C. Kiekintveld, A. Piplai, A. Waggler, O. Kolade, and B. H. Matti, “An investigation into the

performances of the Current state-of-the-art Naive Bayes, Non-Bayesian and Deep Learning Based Classifier for Phishing Detection: A Survey," Nov. 24, 2024, *arXiv*: arXiv:2411.16751. doi: 10.48550/arXiv.2411.16751.

[10] F. O. Catak and A. F. Yazı, "A Benchmark API Call Dataset for Windows PE Malware Classification," Feb. 21, 2021, *arXiv*: arXiv:1905.01999. doi: 10.48550/arXiv.1905.01999.

[11] A. Tajoddin and S. Jalili, "HM3alD: Polymorphic Malware Detection Using Program Behavior-Aware Hidden Markov Model," *Appl. Sci.*, vol. 8, no. 7, Art. no. 7, July 2018, doi: 10.3390/app8071044.

[12] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," *Int. J. Comput. Appl.*, vol. 44, no. 3, pp. 270–277, Mar. 2022, doi: 10.1080/1206212X.2020.1732641.

[13] M. Schofield *et al.*, "Convolutional Neural Network for Malware Classification Based on API Call Sequence," in *Computer Science & Information Technology (CS & IT)*, AIRCC Publishing Corporation, Jan. 2021, pp. 85–98. doi: 10.5121/csit.2021.110106.

[14] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence," *Comput. Secur.*, vol. 92, p. 101760, May 2020, doi: 10.1016/j.cose.2020.101760.

[15] "Facts & Analyses on the Threat Scenario: The AV-TEST Security Report 2020/2024." Accessed: Nov. 24, 2024. [Online]. Available: https://www.av-test.org/en/

[16] S. Ahn, S. Ahn, H. Koo, and Y. Paek, "Practical Binary Code Similarity Detection with BERT-based Transferable Similarity Learning," in *Proceedings of the 38th Annual Computer Security Applications Conference*, in ACSAC '22. New York, NY, USA: Association for Computing Machinery, Dec. 2022, pp. 361–374. doi: 10.1145/3564625.3567975.

[17] "On the classification of Microsoft-Windows ransomware using hardware profile [PeerJ]." Accessed: Jan. 05, 2025. [Online]. Available: https://peerj.com/articles/cs-361/

[18] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-Tuned Domain Generation and Detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, in AISec '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 13–21. doi: 10.1145/2996758.2996767.

[19] C. Yavvari, A. Tokhtabayev, H. Rangwala, and A. Stavrou, "Malware Characterization Using Behavioral Components," in *Computer Network Security*, I. Kotenko and V. Skormin, Eds., Berlin, Heidelberg: Springer, 2012, pp. 226–239. doi: 10.1007/978-3-642-33704-8_20.

[20] A. I. A. Alzahrani, M. Ayadi, M. M. Asiri, A. Al-Rasheed, and A. Ksibi, "Detecting the Presence of Malware and Identifying the Type of Cyber Attack Using Deep Learning and VGG-16 Techniques," *Electronics*, vol. 11, no. 22, Art. no. 22, Jan. 2022, doi: 10.3390/electronics11223665.

[21] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, p. e23574, Dec. 2023, doi: 10.1016/j.heliyon.2023.e23574.

[22] S. M. Abdulla, *An artificial co-stimulation classifier for malicious API calls classification in portable executable malwares*. University of Malaya (Malaysia), 2012. Accessed: Sept. 17, 2025. [Online]. Available: https://search.proquest.com/openview/0238fb74680eba7e1d85fa4b975a3958/1?pq-origsite=gscholar&cbl=2026366&diss=y

[23] R. M. Abdullah, S. M. Abdullah, and S. M. Abdullah, "Neighborhood Component Analysis and Artificial Neural Network for DDoS Attack Detection over IoT Networks," in *2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic"(IEC)*, IEEE, 2021, pp. 1–6. Accessed: Sept. 17, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9476123/

[24] R. M. Abdullah, S. M. Abdullah, and S. M. Abdullah, "Neighborhood Component Analysis and Artificial Neural Network for DDoS Attack Detection over IoT Networks," in *2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic"(IEC)*, IEEE, 2021, pp. 1–6. Accessed: Sept. 17, 2025. [Online].

Available: https://ieeexplore.ieee.org/abstract/document/9476123/

[25] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[26] M. Ali, "Machine Learning-Based Ransomware Detection Through Static Analysis of PE File Features".

[27] G. Mutlu, "Comparative Evaluation of Machine and Deep Learning Models for PE File Malware Detection".

[28] L. De Rose, G. Andresini, A. Appice, and D. Malerba, "OLIVANDER: a counterfactual-based method to generate adversarial Windows PE malware," *Data Min. Knowl. Discov.*, vol. 39, no. 5, p. 46, Sept. 2025, doi: 10.1007/s10618-025-01127-1.

[29] M. Dhull1 and * Chhavi Rana2, "An Empirical Study on the Effectiveness of Adversarial Examples in Window PE Malware Detection Model," Mar. 12, 2025, *Preprints*. doi: 10.22541/au.174175774.46200982/v1.

[30] H. Manthena, S. Shajarian, J. C. Kimmell, M. Abdelsalam, S. Khorsandroo, and M. Gupta, "Explainable Artificial Intelligence (XAI) for Malware Analysis: A Survey of Techniques, Applications, and Open Challenges," *IEEE Access*, vol. 13, pp. 61611–61640, 2025, doi: 10.1109/ACCESS.2025.3555926.

[31] Y. He *et al.*, "On Benchmarking Code LLMs for Android Malware Analysis," in *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Clarion Hotel Trondheim Trondheim Norway: ACM, June 2025, pp. 153–160. doi: 10.1145/3713081.3731745.

[32] G. Gebrehans *et al.*, "Generative Adversarial Networks for Dynamic Malware Behavior: A Comprehensive Review, Categorization, and Analysis," *IEEE Trans. Artif. Intell.*, pp. 1–23, 2025, doi: 10.1109/TAI.2025.3537966.

[33] R. Darwish, M. Abdelsalam, and S. Khorsandroo, "Deep Learning Based XIoT Malware Analysis: A Comprehensive Survey, Taxonomy, and Research Challenges," Oct. 14, 2024, *arXiv*: arXiv:2410.13894. doi: 10.48550/arXiv.2410.13894.

[34] D. Rabadi, J. Y. Loo, A. Narayanan, Y. Wang, S. G. Teo, and T. Truong-Huu, "FETA: A systematic and efficient approach for feature engineering on anti-static and anti-dynamic malware analysis," *J. Inf. Secur. Appl.*, vol. 93, p. 104104, Sept. 2025, doi: 10.1016/j.jisa.2025.104104.

[35] H. Shokouhinejad, R. Razavi-Far, G. Higgins, and A. A. Ghorbani, "Explainable Attention-Guided Stacked Graph Neural Networks for Malware Detection," Aug. 14, 2025, *arXiv*: arXiv:2508.09801. doi: 10.48550/arXiv.2508.09801.

[36] "Dynamic API call sequence visualisation for malware classification", doi: 10.1049/iet-ifs.2018.5268.

[37] A. A. Alhashmi *et al.*, "Similarity-Based Hybrid Malware Detection Model Using API Calls," *Mathematics*, vol. 11, no. 13, p. 2944, Jan. 2023, doi: 10.3390/math11132944.

[38] Y. Ki, E. Kim, and H. K. Kim, "A Novel Approach to Detect Malware Based on API Call Sequence Analysis," *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 6, p. 659101, June 2015, doi: 10.1155/2015/659101.

[39] S. Gupta, H. Sharma, and S. Kaur, "Malware Characterization Using Windows API Call Sequences," in *Security, Privacy, and Applied Cryptography Engineering*, C. Carlet, M. A. Hasan, and V. Saraswat, Eds., Cham: Springer International Publishing, 2016, pp. 271–280. doi: 10.1007/978-3-319-49445-6_15.

[40] L. Liu, B. Wang, B. Yu, and Q. Zhong, "Automatic malware classification and new malware detection using machine learning," *Front. Inf. Technol. Electron. Eng.*, vol. 18, no. 9, pp. 1336–1347, Sept. 2017, doi: 10.1631/FITEE.1601325.

[41] "Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction." Accessed: Jan. 27, 2025. [Online]. Available: https://www.mdpi.com/2076-3417/9/2/239

[42] Y. M. Cho and H. Y. Kwon, "Machine Learning Based Malware Detection Using API Call Time Interval," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 30, no. 1, pp. 51–58, Feb. 2020, doi: 10.13089/JKIISC.2020.30.1.51.

[43] V. Voronin and A. Morozov, "Analyzing API Sequences for Malware Monitoring Using Machine Learning," in

2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Nov. 2021, pp. 519–522. doi: 10.1109/SUMMA53307.2021.9632005.

[44] Eslam Amer, Adham Samir, Hazem Mostafa, Amer Mohamed, Mohamed Amin, "Malware Detection Approach Based on the Swarm-Based Behavioural Analysis over API Calling Sequence".

[45] L. Cui, J. Cui, Y. Ji, Z. Hao, L. Li, and Z. Ding, "API2Vec: Learning Representations of API Sequences for Malware Detection," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, in ISSTA 2023. New York, NY, USA: Association for Computing Machinery, July 2023, pp. 261–273. doi: 10.1145/3597926.3598054.

[46] G. Balan, C.-A. Simion, D. T. Gavriluţ, and H. Luchian, "Feature mining and classifier selection for API calls-based malware detection," *Appl. Intell.*, vol. 53, no. 23, pp. 29094–29108, Dec. 2023, doi: 10.1007/s10489-023-05086-2.

[47] T. Chen, H. Zeng, M. Lv, and T. Zhu, "CTIMD: Cyber threat intelligence enhanced malware detection using API call sequences with parameters," *Comput. Secur.*, vol. 136, p. 103518, Jan. 2024, doi: 10.1016/j.cose.2023.103518.

[48] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park, and H. Jeon, "CNN-Based Android Malware Detection," in *2017 International Conference on Software Security and Assurance (ICSSA)*, July 2017, pp. 60–65. doi: 10.1109/ICSSA.2017.18.

[49] M. Chowdhury, A. Rahman, and R. Islam, "Malware Analysis and Detection Using Data Mining and Machine Learning Classification," in *International Conference on Applications and Techniques in Cyber Security and Intelligence*, vol. 580, J. Abawajy, K.-K. R. Choo, and R. Islam, Eds., in Advances in Intelligent Systems and Computing, vol. 580. , Cham: Springer International Publishing, 2018, pp. 266–274. doi: 10.1007/978-3-319-67071-3_33.

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

[51] A. Almaleh, R. Almushabb, and R. Ogran, "Malware API Calls Detection Using Hybrid Logistic Regression and RNN Model," *Appl. Sci.*, vol. 13, no. 9, p. 5439, Jan. 2023, doi: 10.3390/app13095439.

[52] N. Owoh, J. Adejoh, S. Hosseinzadeh, M. Ashawa, J. Osamor, and A. Qureshi, "Malware Detection Based on API Call Sequence Analysis: A Gated Recurrent Unit–Generative Adversarial Network Model Approach," *Future Internet*, vol. 16, no. 10, p. 369, Oct. 2024, doi: 10.3390/fi16100369.

[53] "Windows Malware Detection via Enhanced Graph Representations with Node2Vec and Graph Attention Network." Accessed: Aug. 22, 2025. [Online]. Available: https://www.mdpi.com/2076-3417/15/9/4775

[54] P. Kulkarni and S. OShaughnessy, "Malware Detection Using Dynamic Graph Neural Networks," *Eur. Conf. Cyber Warf. Secur.*, vol. 24, no. 1, pp. 830–837, June 2025, doi: 10.34190/eccws.24.1.3459.

[55] M. Shahnawaz, B. P. Gond, and D. P. Mohapatra, "Dynamic Malware Classification of Windows PE Files using CNNs and Greyscale Images Derived from Runtime API Call Argument Conversion," May 30, 2025, *arXiv*: arXiv:2505.24231. doi: 10.48550/arXiv.2505.24231.

[56] A. Oliveira, "Malware Analysis Datasets: API Call Sequences." IEEE, Oct. 23, 2019. Accessed: Feb. 07, 2025. [Online]. Available: https://ieee-dataport.org/open-access/malware-analysis-datasets-api-call-sequences

[57] *Malware Bazzar*. Accessed: Oct. 04, 2024. [Online]. Available: https://bazaar.abuse.ch/

[58] *virusshare.com*. Accessed: Apr. 10, 2025. [Online]. Available: https://virusshare.com/hashes

[59] *portableapps.com*. Accessed: Apr. 10, 2025. [Online]. Available: https://portableapps.com/