Malware Analysis Using Transformer Based Models: An Empirical Study

Abhishek Joshi[1,*], Divyateja Pasupuleti[1,*], Nischith P.[1,*], Sarvesh Sutaone[1,*], Soumil Ray[1,*], Soumyadeep Dey[2] and Barsha Mitra[1,**]

[1]*Department of CSIS, BITS Pilani, Hyderabad Campus, Hyderabad, India*

[2]*Microsoft, India*

Keywords: Malware, Android Malware, IoT Malware, Transformer Models, Malware Classification.

Abstract: The massive demand for connected and smart applications and the growth of high-speed communication technologies like 5G have led to a surge in the use of Android and Internet-of-Things (IoT) devices. The popularity of such devices has resulted in a huge number of malware attacks and infections being inflicted upon these devices. Cyber criminals relentlessly target the Android and IoT devices by developing new strains of malware. To defend against these malware attacks, researchers have developed different types of malware detection and categorization techniques. In this paper, we investigate the applicability and effectiveness of different transformer-based models, which use self-attention to learn global dependencies and contextual information, for malware classification on two platforms: Android and IoT. We consider two types of inputs for malware analysis - images and sequences. For image-based analysis, we convert Android APKs and IoT traffic into images that reflect their structural and behavioral features. We compare various convolutional neural network (CNN) based models with and without transformer layers, and a pure transformer model that directly processes the images. For sequence-based analysis, we extract the API call sequences from Android APKs, and apply a transformer model to encode and classify them. We also explore the effect of pretraining and embedding initialization on the transformer models. Our experiments demonstrate the advantages and limitations of using transformer-based models for malware classification, and provide insights into the training strategies and challenges of these models. To the best of our knowledge, this is the first work that systematically explores and compares different transformer-based models for malware classification on both image and sequence inputs.

1 INTRODUCTION

Malicious applications, or malware, pose a serious threat to the security and privacy of various computing systems, as they can damage, steal, or spy on the data and resources of the host devices. The Internet plays a key role in the dissemination and infection of malware across different types of device, such as desktops, laptops, servers, mobile phones, tablets, and Internet-of-Things (IoT) devices. These devices have diverse resource constraints, such as memory, battery, and processing power. Mobile phones, tablets, and IoT devices have become essential for our daily activities and communications. However, they also face a surge of malware attacks, with more than 5.04 billion incidents reported in 2022 as per 2022 SonicWall Cyber Threat Report[1]. 2023 experienced a worldwide increase in malware volume by 2% with a massive surge of 87% in IoT malware[2]. Moreover, in 2023, 6.06 billion malware attacks were recorded alone by researchers associated with SonicWall Capture Labs[3]. Android, being one of the most popular mobile operating systems, attracts a large number of malware developers and distributors. To combat this challenge, various malware detection and classification techniques have been proposed and implemented

*Abhishek Joshi, Divyateja Pasupuleti, Nischith P., Sarvesh Sutaone and Soumil Ray have equal contribution

**Corresponding Author

[1]https://www.sonicwall.com/resources/white-papers/2022-sonicwall-cyber-threat-report/

[2]https://www.sonicwall.com/news/2023-sonicwall-cyber-threat-report-casts-new-light-on-shifting-front-lines-threat-actor-behavior/

[3]https://www.sonicwall.com/medialibrary/en/white-paper/2024-cyber-threat-report.pdf

Joshi, A., Pasupuleti, D., Nischith, P., Sutaone, S., Ray, S., Dey, S. and Mitra, B.
Malware Analysis Using Transformer Based Models: An Empirical Study.
DOI: 10.5220/0012855100003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 858-865
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

in the literature.

Attention mechanism is a powerful technique that allows a model to dynamically weigh the relevance of different parts of the input and output sequences, and to process them in parallel without relying on recurrent or convolutional operations. This enables the model to capture long-range dependencies and contextual information in data, which are essential for many natural language processing and computer vision tasks. Transformer architecture, which is based on attention mechanism, has achieved remarkable results and efficiency in these domains, surpassing previous state-of-the-art models (Vaswani et al., 2023). However, the applications of attention and transformer are not limited to natural language and computer vision. Recently, they have also been explored in various malware analysis and classification tasks, where they have shown promising performance and robustness against malware obfuscation and evasion techniques. Malware analysis and classification is a challenging problem that requires the model to understand the semantic and syntactic features of malicious code as well as the behavioral and contextual aspects of its execution. Attention and transformer can potentially address these challenges by learning high-level representations of malware from different sources of data, such as binary, assembly, API calls, or network traffic (Seneviratne et al., 2022), (Jo et al., 2023), (Ravi et al., 2023).

Malware classification is a complex task that requires the ability to capture the complex and diverse behaviors of ever-evolving malicious programs. In recent years, transformer-based models have been increasingly used for malware categorization. In this paper, we investigate the applicability and effectiveness of different transformer-based models for malware classification on two platforms: Android and IoT. Our inputs are primarily of two types: images and sequences. Image-based malware analysis involves conversion of the Android APKs and IoT traffic traces into images. Such images capture the morphological and behavioral features of the corresponding data. We then compare the performance of various convolutional neural network (CNN) based models, such as ResNet-50 (He et al., 2015), MobilenetV2 (Sandler et al., 2018), and LCNN (Yuan et al., 2022), with and without transformer layers on top of them. We also evaluate the effect of a pure transformer model, namely the Visual Transformer (ViT), which directly processes the malware images without any CNN layers. For sequence-based malware analysis, the API call sequences are extracted from Android APKs. These sequences represent the interactions between the apps and the system. We then apply a transformer model, namely BERT (Devlin et al., 2018), to encode and classify the API call sequences. Our experiments present a detailed performance comparison of malware classification strategies using transformer-based models. Moreover, various aspects of the training strategies and the challenges of these models are also highlighted. To the best of our knowledge, this is the first work that explores the capabilities of different transformer-based models for malware classification on both image and sequence inputs.

The rest of the paper is organized as follows. Section 2 provides details of how Android applications can be converted into images and sequences and the manner in which images can be obtained from IoT traffic traces. Section 3 outlines the models considered and the experimental setup. In Section 4, we present the details of the datasets that we have used for our experimental analysis. Section 5 presents the detailed experimental results and the corresponding analysis providing insights into the relative performances of the different models. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

This section describes the approaches for generating images and sequences from Android applications (APKs), which are the main inputs for our malware detection and analysis models. Additionally, we illustrate how we transform IoT traffic data into images, which can capture the patterns and anomalies of the network activities of the different connected devices.

Android Application to Image: One way to visualize the internal structure of a binary file is to convert it into an array of 8-bit unsigned integers, where each array element represents a byte value from 0 to 255. This array can then be displayed as a grayscale image, where darker pixels correspond to lower byte values and lighter pixels represent higher byte values. This image representation of a binary file is called a byteplot and it reveals the patterns and features that reflect the file's content and format. Byteplot was first proposed by Conti et al. (Conti et al., 2008) as a tool to analyze and compare binary files. Later, Nataraj et al. (Nataraj et al., 2011) used byteplot to transform malware samples into images and classify them using image processing techniques. In this work, we use byteplot to convert APK files into images and extract features from them for malware classification.

Android Application to Sequences: To obtain sequences of API invocations from an APK, we first need to extract the function call graph (FCG) of the

APK. FCG represents the possible paths of execution among the methods in the APK, and the API invocations are the calls to the methods provided by the Android framework or other libraries. We then apply different graph traversal algorithms on the FCG to generate sequences that capture the relative order of API invocations as proposed in (Cannarile et al., 2022) in different scenarios. These sequences are called API sequences and they can be used to characterize the behavior and functionality of the APK.

IoT Traffic Data to Image: One of the most common sources of IoT malware analysis is pcap files, which are a standard format for storing network packets captured by packet sniffing tools. These tools can monitor network traffic on a specific interface, either in real-time or based on predefined filters and settings. Each packet contains information like the IP addresses for source and destination, protocols, port numbers, and payload data. The packets are the basic units of data transmission over a network. By analysing pcap files, we can examine the communication patterns, anomalies, and potentially malicious activities within IoT networks, as malware often exhibits distinctive network behaviour, like using unusual protocols, generating high traffic volume, or connecting to suspicious IP addresses. We can also use pcap files for behavioural analysis of IoT devices based on how they interact with the network and other devices using flow or session data. A flow is a sequence of packets that have common attributes, such as the source and destination IP or port pairs and the protocol type (e.g., TCP, UDP). A session is a bidirectional flow, i.e., the source and destination IP or port pairs are interchangeable. To generate images from pcap files, we use SplitCap, an open-source tool that can split a pcap file into multiple pcap files based on a given criterion, such as splitting per flow or per session. We also use SplitCap to select whether to extract only the application layer data (the 7th layer in the OSI model) or the whole packet. After splitting, we either trim the files or pad the files with 0s until they reach a uniform size that can be reshaped into an NxN image, for example, 784 bytes for a 28x28 image. We then read the files byte-wise and append the values to an array, where each value represents the pixel intensity. We reshape the array to the desired size, resulting in an NxN image.

3 METHODOLOGIES

To explore the potential and performance of different transformer-based models for malware classification, we conduct experiments using Android APKs and IoT traffic traces. We choose data corresponding to the Android and IoT platforms because they are widely used and are vulnerable to various types of malware attacks. We design two types of inputs for our models: images and sequences, which capture different aspects of malware features.

For image-based malware analysis, we transform the raw bytes of Android APKs and IoT traffic into grayscale images (as described in Section 2). We adopt several CNN based models, which are commonly used for image recognition tasks, and enhance them with transformer layers, which can learn global dependencies and attention among image patches. Specifically, we compare the following models: ResNet-50 (He et al., 2015), which uses residual connections and deep layers to achieve high accuracy; MobilenetV2 (Sandler et al., 2018), which uses depth wise separable convolutions and inverted residuals to reduce the computational cost and model size; and LCNN (Yuan et al., 2022), which uses local convolutional layers to capture local features and reduce the number of parameters. We also append a transformer encoder and a classification head on top of these CNN models, and examine how the transformer layers affect the classification performance and the model complexity. Furthermore, we evaluate a pure transformer model, namely Vision Transformer (ViT) (Dosovitskiy et al., 2020). ViT does not employ any CNN layers and directly processes the image patches. This model relies on the transformer encoder and the classification head to encode and classify the malware images.

API call sequences imbibe the high-level interactions between Android apps and the system. Such sequences can be inspected to classify malware. To analyze malware based on their API call sequences, we employ a transformer model architecture, namely BERT (Devlin et al., 2018), that can encode and comprehend natural language sequences. We train BERT from scratch on our malware dataset, and use its output embeddings and a classification head to distinguish the API call sequences. We experiment with two methods for initializing the embeddings of the APIs. In one method, we assign a unique 128-dimensional embedding to each API in the dataset, and learn them using a linear projection layer. In another method, we use a Skip-gram model similar to Word2Vec (Mikolov et al., 2013) to generate vector representations for the API calls. The Skip-gram model is adapted from natural language processing to learn embeddings from large corpora of Android APKs. It processes pairs of API calls that occur within the same method, and assigns a unique index to each API call in the vocabulary. The model iterates through batches of API-API pairs from both benign and malicious APKs during training.

4 DATASET

In this section, we describe the datasets that we have considered for our experiments.

AndroZoo: The AndroZoo dataset was first published in (Allix et al., 2016). Since then, the dataset has been continuously augmented. In its current form, AndroZoo consists of 24,470,628 APKs collected from various sources, including the Google Play app market and is possibly the largest malware dataset. These APKs are analyzed by numerous anti-virus tools to identify malicious and benign contents. Androzoo uses a vt_detection (Allix et al., 2016) mechanism which is the number of virus total engines that detected an APK as malicious. For the set of APKs that we have considered, we have used a criterion that an APK is considered as malicious if it is identified as malicious by a minimum of 5 engines, otherwise the APK is considered as benign. We randomly chose APKs such that the total size of the APKs is not more than 150 GB. Thus, we considered 1,23,880 malicious APKs and 3,23,095 benign APKs.

MALNET-IMAGE and MALNET-IMAGE TINY Datasets: The MALNET-IMAGE dataset (Freitas et al., 2022) consists of 12,62,024 images from APKs in the Androzoo dataset, which spans 47 types and 696 families of malware. The MALNET-IMAGE Tiny dataset is a reduced version of MALNET-IMAGE that removes the four most common malware types in the original dataset. In the rest of the paper, we will refer to MALNET-IMAGE-Tiny dataset as MALNET-Tiny. Freitas et al. generated images from APKs by extracting the DEX file from each APK and converting it into a byte sequence, which correspond to a 1D array of pixel values between 0 and 255. They then converted the 1D array into a 2D array using linear plotting with the size parameters recommended in (Nataraj et al., 2011), thereby creating a grayscale image. They resized the image to 256x256 and assigned a colour to each byte based on its position in the DEX file by adopting the approach in (Gennissen and Blasco, 2017).

Drebin: Drebin (Arp et al., 2014) is a public dataset for Android malware family classification, consisting of 1,29,013 applications collected from various sources over two years (August 2010 - October 2012). Among them, 5,560 are malicious, belonging to 179 families, some of which are still active in the market. The malicious applications have been identified by at least two of the ten anti-virus scanners used. The dataset is highly imbalanced in terms of malware family distribution. Therefore, we focus on the top 20 malware families for our experiments. Drebin is one of the largest and most widely used datasets for Android malware classification and provides a valuable resource for testing and developing malware identification algorithms.

CICAndMal2017: CICAndMal2017 (Lashkari et al., 2018) is a collection of Android applications with different security labels, obtained from various sources. The dataset consists of 5,491 Android applications, out of which 426 applications are malware and 5,065 applications are benign. The malware samples belong to four categories: Ransomware, Adware, SMS Malware and Scareware.

IoT dataset (USTC-TFC2016): The USTC-TFC2016 dataset (Wang et al., 2017) is a collection of 20 raw network traffic capture files (pcap) that represent 20 different classes of traffic. Out of the 20 classes, 10 are malicious and the remaining 10 are benign. The malicious traffic classes were obtained from public sources by Stratosphere Research Laboratory, and these classes include various types of malware and botnet activities. Due to the large sizes of some of the malicious traffic files, only segments of them were used for the analysis, while smaller files that were generated by the same applications were merged together. The benign traffic classes were generated by IXIA BPS, a professional network traffic simulation tool that can emulate realistic network scenarios. The benign traffic classes cover eight common application categories, such as MySQL, Facetime, and others and reflect typical network usage patterns.

5 RESULTS AND DISCUSSION

Table 1 shows the performance of different image-based models for Android malware classification on four datasets: MALNET-Tiny (Freitas et al., 2022), Drebin Top 10, Drebin Top 20 (Arp et al., 2014), and CICAndMal2017 (Lashkari et al., 2018). The metrics used to evaluate the models are Accuracy (Acc) and F1-score (F1-s), which is the harmonic mean of precision and recall. The models include ResNet-50 (He et al., 2015), MobileNetV2 (Sandler et al., 2018), and LCNN (0.25) (Yuan et al., 2022). These are all convolutional neural networks (CNNs) with different architectures and complexities. The table also shows the variants of these models with one or two transformer layers (1T or 2T) attached after the CNN feature extraction. The rationale behind us-

Table 1: Results for Android Malware Classification using CNN and Transformer based Models on 4 datasets: MALNET-Tiny (43 malware classes), Drebin Top 10 (top 10 malware classes), Drebin Top 20 (top 20 malware classes), and CICAndMal2017 (1 benign, and 4 malware classes). Acc represents Accuracy and F1-s denotes F1-score. NA values indicate that pretaining has been done on the respective dataset. For a dataset, values in boldface correspond to the best performing model. The gray colored cells represent the best performance across a model and its two variations for a specific dataset.

Model	**MALNET-Tiny**		**Drebin Top 10**		**Drebin Top 20**		**CICAndMal2017**	
	Acc	*F1-s*	*Acc*	*F1-s*	*Acc*	*F1-s*	*Acc*	*F1-s*
ResNet-50	70.05	70.95	**90.56**	**90.60**	**86.88**	**86.99**	75.00	75.05
ResNet-50-1T	**74.23**	**75.48**	87.95	88.64	84.84	86.10	64.21	63.94
ResNet-50-2T	72.01	72.66	89.56	89.70	80.86	82.50	60.78	61.21
MobileNetV2	74.54	75.11	87.45	87.65	84.62	85.14	69.61	69.64
MobileNetV2-1T	72.29	73.19	81.12	81.77	75.91	77.42	66.18	67.01
MobileNetV2-2T	71.30	72.72	75.78	77.00	74.73	77.03	66.18	61.80
LCNN (0.25)	70.68	71.52	**90.56**	**90.63**	84.30	86.59	**77.94**	**78.22**
LCNN (0.25)-1T	71.27	72.20	88.57	89.38	82.69	84.01	74.51	70.81
LCNN (0.25)-2T	68.91	69.76	85.22	85.51	77.63	79.93	67.64	67.85
ViT	70.77	70.26	66.83	68.06	64.19	63.66	58.33	58.30
ViT-PreMALNET-Tiny	NA	NA	70.65	71.23	61.74	56.99	68.89	66.27
ViT-PreMALNET	NA	NA	60.41	59.52	49.42	41.55	61.24	62.71

ing transformer layers is that they are expected to capture the global dependencies and long-range interactions among the image pixels. Table 1 also includes results for the visual image transformer model, ViT (Dosovitskiy et al., 2020). ViT is the model that uses only transformer layers for feature extraction and classification, without any convolutional layers. Additionally, the table shows the results for ViT models pretrained on MALNET-Tiny (model name ViT-PreMALNET-Tiny) and MALNET (model name ViT-PreMALNET) datasets, which are larger and more diverse than the target datasets. From the table, we can observe the following:

- ResNet-50 is the best performing CNN model on Drebin Top 10 and CICAndMal2017 datasets, achieving the highest Accuracies and F1-scores among all the models. However, it is outperformed by MobileNetV2 and LCNN (0.25) on MALNET-Tiny dataset, and by LCNN (0.25) on Drebin Top 20 dataset. This suggests that ResNet-50 is more effective on datasets with fewer classes and more balanced samples, but may suffer from overfitting or complexity issues on datasets with more classes and imbalanced samples.
- Adding one or two transformer layers after CNN feature extraction generally reduces the performance of the models on all the datasets, except for LCNN (0.25)-1T on MALNET-Tiny dataset. This indicates that the transformer layers may not be beneficial for the Android malware image classification task, as they may introduce more parameters, noise, or redundancy into the models. Alternatively, the transformer layers may require more data or fine-tuning to adapt to the task.
- MobileNetV2 and LCNN (0.25) are the best performing lightweight CNN models on the datasets. These two models achieve comparable or better results than ResNet-50 on MALNET-Tiny and Drebin Top 20 datasets, and exhibit slightly lower performance on Drebin Top 10 and CICAndMal2017 datasets. These models have fewer parameters and lower computational cost than ResNet-50, and may be more suitable for mobile or resource-constrained devices.
- ViT is the worst performing model on all the datasets, achieving the lowest accuracy and F1-score values among all the models. This suggests that ViT is not well suited for image-based Android malware classification. The reason for this can be because ViT may not be able to capture the semantic or structural features of the images, or may require more data or fine-tuning to adapt to the task.
- Pretraining ViT on MALNET-Tiny or MALNET-IMAGE datasets slightly improves the performance of ViT on some of the datasets, but not on others. For example, ViT-PreMALNET-Tiny achieves higher accuracies and F1-scores than ViT on Drebin Top 10 and CICAndMal2017 datasets, but lower metric values on Drebin Top 20 dataset. ViT-PreMALNET achieves higher accuracy and F1-score than ViT on CICAndMal2017 dataset, but lower scores on Drebin Top 10 and Drebin Top 20 datasets. This indicates that the pretraining data may not be representative or sufficient for the target datasets, or that the transfer learning may not be effective for the task.

Table 2 shows the results of using BERT to clas-

Table 2: Results for Android Malware Classification through API Sequence Classification using BERT on three datasets: Androzoo (benign and malware classes), Drebin Top 20 (top 20 malware classes) and CICAndMal2017 (4 malware classes). The upper half of the table presents results where APIs are initialized with linear projection and the lower half shows results where APIs are initialized with Skip-gram. Acc represents Accuracy and F1-s denotes F1-score. For a dataset, values in boldface correspond to the best performing model.

API initialized with Linear Projection				
Dataset	*Acc*	*Precision*	*Recall*	*F1-s*
Androzoo	70.75	70.23	74.01	72.07
Drebin Top 20	69.27	02.99	03.70	03.19
CICAndMal2017	63.53	16.05	15.68	15.80
API initialized with Skip-gram				
Dataset	*Acc*	*Precision*	*Recall*	*F1-s*
Androzoo	**84.00**	**85.10**	**82.56**	**83.81**
Drebin Top 20	**79.42**	**05.56**	**05.65**	**05.52**
CICAndMal2017	**63.53**	**16.75**	**16.12**	**16.18**

sify Android malware based on API sequences, with two different ways of initializing the API embeddings: Linear Projection and Skip-gram. Skip-gram is a method of learning API embeddings from the context of the API corpus, which can capture the semantic and syntactic similarities of the APIs. The table reports the accuracy, precision, recall and F1-score of the models on each dataset. The results are reported on three datasets: Androzoo, Drebin Top 20 and CICAndMal2017, which vary in size, diversity and imbalance of malware classes. Some of the key findings from the analysis are:

- BERT with Skip-gram consistently outperforms BERT with linear projection on all three datasets, achieving higher accuracy, precision, recall and F1-score. This suggests that Skip-gram is a better way of initializing the APIs for BERT, as it preserves the semantic and syntactic information of the APIs, which may be useful for malware classification.
- BERT with Skip-gram achieves the highest performance on Androzoo, a large and balanced dataset of benign and malicious Android applications. The model achieves an accuracy of 84.00%, a precision of 85.10%, a recall of 82.56% and an F1-score of 83.81%, indicating that it can effectively distinguish between benign and malicious applications based on their API sequences. This also shows that BERT can handle long and complex API sequences, as Androzoo contains applications with varying lengths and complexities of API sequences.
- BERT with Skip-gram performs poorly on Drebin Top 20 and CICAndMal2017, two smaller and imbalanced datasets of Android malware samples across different malware classes. The model achieves low accuracy, precision, recall and F1-score on both datasets, indicating that it cannot capture the subtle differences between different malware classes based on their API sequences. This may be due to the lack of sufficient training data, high class imbalance, or the high similarity of API sequences among different malware classes.

Table 3: IoT Malware classification Results on USTC-TFC2016 dataset. Acc represents Accuracy and F1-s denotes F1-score. For a dataset, values in boldface correspond to the best performing model. The gray colored cells represent the best performance across a model and its two variations for a specific dataset.

Model	*Acc*	*Precision*	*Recall*	*F1-s*
ResNet-50	**99.81**	**99.83**	**99.76**	**99.79**
ResNet-50-1T	99.67	99.24	99.71	99.45
MobileNetV2	99.79	99.69	99.70	99.69
MobileNetV2-1T	99.58	99.55	99.47	99.51
LCNN (0.25)	99.20	99.38	99.17	99.27
LCNN (0.25)-1T	98.34	97.77	97.75	97.74
ViT	82.67	76.41	72.83	72.45

Table 3 shows the performance of different image classification models on the IoT dataset, USTC-TFC2016 (Wang et al., 2017). This dataset consists of 20 traffic classes that are commonly found in the IoT domain. The models include ResNet-50 (He et al., 2015) MobileNetV2 (Sandler et al., 2018), LCNN (0.25) (Yuan et al., 2022), and ViT (Dosovitskiy et al., 2020), as well as their variants with one transformer layer (1T) added after the convolutional feature extraction. The metrics used to evaluate the models are Accuracy, Precision, Recall, and F1-score. From Table 3, we observe the following:

- ResNet-50 achieves the highest performance among all the models, with an accuracy of 99.81%, a precision of 99.83%, a recall of 99.76%, and an F1-score of 99.79%. This indicates that ResNet-50 is very effective at recognizing the classes in the IoT dataset, with high accuracy and balanced precision and recall. ResNet-50 is a deep residual network that uses skip connections to overcome the degradation problem of deep networks. ResNet-50 has 50 layers and is widely used for various computer vision tasks.
- Adding one transformer layer to ResNet-50 slightly degrades its performance, as shown by ResNet-50-1T, which has an accuracy of 99.67%, a precision of 99.24%, a recall of 99.71%, and an F1-score of 99.45%. The transformer layer is a

self-attention mechanism that can capture long-range dependencies and semantic relations among the features. However, in this case, the transformer layer may introduce some noise or redundancy which in turn reduces the precision of the model, while maintaining a high recall.

- MobileNetV2 is another high-performing model, with an accuracy of 99.79%, a precision of 99.69%, a recall of 99.70%, and an F1-score of 99.69%. MobileNetV2 is a lightweight and efficient network that uses inverted residual blocks and depth wise separable convolutions to reduce the computational cost and parameter size. MobileNetV2 is suitable for mobile and embedded devices as well as IoT applications.
- Similar to ResNet-50, adding one transformer layer to MobileNetV2 also lowers its performance, as shown by MobileNetV2-1T, which has an accuracy of 99.58%, a precision of 99.55%, a recall of 99.47%, and an F1-score of 99.51%. The transformer layer may not bring much benefit to the already compact and powerful MobileNetV2, and may instead introduce some overhead or complexity that affects the precision and recall of the model.
- LCNN (0.25) is a low-complexity network that uses a fraction of the parameters and operations of standard CNNs by applying a low-rank decomposition to the convolutional filters. LCNN (0.25) has an accuracy of 99.20%, a precision of 99.38%, a recall of 99.17%, and an F1-score of 99.27%. This shows that LCNN (0.25) can achieve comparable performance to the other models, while reducing the computational and memory requirements. LCNN (0.25) is also suitable for resource-constrained IoT devices.
- However, adding one transformer layer to LCNN (0.25) significantly lowers its performance, as shown by LCNN (0.25)-1T, which has an accuracy of 98.34%, a precision of 97.77%, a recall of 97.75%, and an F1-score of 97.74%. The transformer layer may not be compatible with the low-rank structure of LCNN (0.25), and may instead degrade the quality and efficiency of the model. The transformer layer may also increase the parameter size and computational cost of the model, which defeats the purpose of using LCNN (0.25).
- ViT is a visual image transformer model that uses only transformer layers to process the images without any convolutional layers. ViT has an accuracy of 82.67%, a precision of 76.41%, a recall of 72.83%, and an F1-score of 72.45%. This indicates that ViT performs poorly on the IoT dataset, compared to the other models. ViT is not able to capture the fine-grained and local features of the IoT data, and may suffer from the low resolution and diversity of the images. ViT may also require more data and training to achieve optimal performance, as it is a data-hungry and complex model. Thus, ViT may not be suitable for IoT applications as it has a large parameter size and computational cost.

6 CONCLUSIONS

In this paper, we have investigated the applicability and effectiveness of different transformer-based models for classifying malware targeting Android applications and IoT devices. We have performed two types of malware analysis - image-based and sequence-based. For image-based malware analysis, we have converted Android APKs and IoT traffic into fixed size images, and have compared various CNN architectures with and without transformer layers, and a pure transformer model to classify the converted images. For sequence-based malware analysis, we extracted the API call sequences from Android APKs, and applied a transformer model to encode and classify them. We have also explored the effect of pretraining and embedding initialization on the transformer models.

Our experiments demonstrate the advantages and limitations of using transformer-based models for malware categorization, and provide insights into the training strategies of these models. The main findings of our study can be summarized as follows:

- For image-based malware analysis, CNN based models in general, have outperformed the transformer-based models on all the datasets. ResNet-50 is the best performing CNN model on datasets with fewer classes and more balanced samples, while MobileNetV2 and LCNN (0.25) are the best performing lightweight CNN models on datasets with more classes and imbalanced samples.
- Adding transformer layers after the CNN feature extraction generally reduces the performance of the lightweight CNN models.
- ViT is the worst performing model on all the datasets, and pretraining ViT on larger and more diverse datasets slightly improves its performance on some of the datasets, but not on others. These results suggest that the transformer-based models may not be beneficial or suitable for the image-based malware analysis task, as they may intro-

duce more parameters, noise, or redundancy to the models, or may require more data or fine-tuning to adapt to the task.

- For sequence-based malware analysis, BERT is an effective model for encoding and classifying the API call sequences extracted from Android APKs. Initializing the API embeddings with Skip-gram, an API2vec technique, significantly improves the performance of BERT. These results suggest that the Skip-gram embeddings capture more semantic and syntactic information about the API sequences than the linear projection, and that they help BERT to learn better representations and classifications of the malware families. However, the Skip-gram embeddings may fail to capture the diversities of the API sequences across different malware and benign families. Moreover, BERT may need higher volume of data or more sophisticated methods to achieve higher performance on dataset containing a large number of malware families.

Our study provides a comprehensive and systematic comparison of different transformer-based models for malware analysis on image and sequence inputs, and has revealed the strengths and weaknesses of these models. Our study has also highlighted the challenges and opportunities for future research on applying transformer-based models for accurate malware analysis and related tasks.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the computing time provided on the high performance computing facility, Sharanga, at the Birla Institute of Technology and Science - Pilani, Hyderabad Campus.

REFERENCES

Allix, K., Bissyandé, T. F., Klein, J., and Le Traon, Y. (2016). Androzoo: Collecting millions of android apps for the research community. In *13th Int. Conf. on Mining Software Repositories*, pages 468–471.

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., and Rieck, K. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*. The Internet Society.

Cannarile, A., Carrera, F., alantucci1, S., Iannacone, A., and Pirlo, G. (2022). A study on malware detection and classification using the analysis of api calls sequences through shallow learning and recurrent neural networks. *Italian Conference on Cybersecurity*, 3260.

Conti, G., Dean, E., Sinda, M., and Sangster, B. (2008). Visual reverse engineering of binary and data files. In *Int. Workshop on Visualization for Computer Security*, page 1 – 17.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929.

Freitas, S., Duggal, R., and Chau, D. H. (2022). Malnet: A large-scale image database of malicious software. In *31st ACM Int. Conf. on Information & Knowledge Management*, page 3948–3952.

Gennissen, J. and Blasco, J. (2017). Gamut : Sifting through images to detect android malware.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Jo, J., Cho, J., and Moon, J. (2023). A malware detection and extraction method for the related information using the vit attention mechanism on android operating system. *Applied Sciences*, 13(11).

Lashkari, A. H., Kadir, A. F. A., Taheri, L., and Ghorbani, A. A. (2018). Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 Int. Carnahan Conf. on Security Technology*, pages 1–7.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. (2011). Malware images: Visualization and automatic classification. In *8th Int. Symposium on Visualization for Cyber Security*.

Ravi, A., Chaturvedi, V., and Shafique, M. (2023). Vit4mal: Lightweight vision transformer for malware detection on edge devices. *ACM Transactions on Embedded Computing Systems*, 22(117).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520.

Seneviratne, S., Shariffdeen, R., Rasnayaka, S., and Kasthuriarachchi, N. (2022). Self-supervised vision transformers for malware detection. *IEEE Access*, 10:103121–103135.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need. In *31st Conf. on Neural Information Processing Systems*.

Wang, W., Zhu, M., Zeng, X., Ye, X., and Sheng, Y. (2017). Malware traffic classification using convolutional neural network for representation learning. In *2017 Int. Conf. on Information Networking*, pages 712–717.

Yuan, B., Wang, J., Wu, P., and Qing, X. (2022). Iot malware classification based on lightweight convolutional neural networks. *IEEE Internet of Things Journal*, 9(5):3770–3783.