AN ENSEMBLE OF PRE-TRAINED TRANSFORMER MODELS FOR IMBALANCED MULTICLASS MALWARE CLASSIFICATION

FERHAT DEMİRKIRAN

A thesis submitted to
the School of Graduate Studies of Kadir Has University
in partial fulfilment of the requirements for the degree of
Master of Science in
Cyber Security

Istanbul, January, 2022

APPROVAL

This thesis titled AN ENSEMBLE OF PRE-TRAINED TRANSFORMER MODELS FOR IMBALANCED MULTICLASS MALWARE CLASSIFICATION submitted by FERHAT DEMİRKIRAN, in partial fulfillment of the requirements for the degree of Master of Science in Cyber Security is approved by

Prof. Dr. Hasan Dağ (Advisor) .....
Kadir Has University

Prof. Dr. Berk Canberk .....
Istanbul Technical University

Assist. Prof. Dr. E. Fatih Yetkin .....
Kadir Has University

I confirm that the signatures above belong to the aforementioned faculty members.

.....
Prof. Dr. Mehmet Timur Aydemir
Dean of School of Graduate Studies
Date of Approval: 11.01.2022

DECLARATION ON RESEARCH ETHICS AND PUBLISHING METHODS

I, FERHAT DEMİRKIRAN; hereby declare

- that this Master of Science Thesis that I have submitted is entirely my own work and I have cited and referenced all material and results that are not my own in accordance with the rules;
- that this Master of Science Thesis does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes and Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with university legislation.

FERHAT DEMİRKIRAN

.....

11.01.2022

To my family

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my thesis supervisor Prof. Dr. Hasan Dağ, Director of Center for Cybersecurity and Critical Infrastructure Protection, Kadir Has University, for always believing in me. His support, motivation, patience, and mentorship have provided me to accomplish this thesis with great pleasure. I also would like to thank Prof. Dr. Berk Canberk and Assist. Prof. Dr. E. Fatih Yetkin for their invaluable comments.

I owe a great sense of gratitude to Dr. Aykut Çayır, Kadir Has University, for his constructive contributions and technical supports. He has always been very supportive. I am incredibly grateful for his guidance through each stage of the dissertation.

I would like to acknowledge Dr. Uğur Ünal, from Center for Cybersecurity and Critical Infrastructure Protection, for his valuable contributions and helpful advice. His recommendations make my dissertation a step further. I would like to thank my colleague Berkant Düzgün for his help and support in this process.

I would like to thank my friends Faruk Cürebal and İsmail Erbaş for their endless support. They made this process much easier.

I am extremely thankful to my parents, Abdulkudüs and Selma Demirkıran, and my siblings for their lifelong supports. They have brought joy and happiness into my life.

Last but not least, I would like to thank Bilge Alır for all her love and support.

AN ENSEMBLE OF PRE-TRAINED TRANSFORMER MODELS FOR IMBALANCED MULTICLASS MALWARE CLASSIFICATION

ABSTRACT

Classification of malware families is crucial for a comprehensive understanding of how they can infect devices, computers, or systems. Hence, malware identification enables security researchers and incident responders to take precautions against malware and accelerate mitigation. API call sequences made by malware are widely utilized features by machine and deep learning models for malware classification as these sequences represent the behavior of malware. However, traditional machine and deep learning models remain incapable of capturing sequence relationships among API calls. Unlike traditional machine and deep learning models, the transformer-based models process the sequences in whole and learn relationships among API calls due to multi-head attention mechanisms and positional embeddings. Our experiments demonstrate that the transformer model with one transformer block layer surpass the performance of the widely used base architecture, LSTM. Moreover, BERT or CANINE, the pre-trained transformer models, outperforms in classifying highly imbalanced malware families according to evaluation metrics: F1-score and AUC score. Furthermore, our proposed bagging-based random transformer forest (RTF) model, an ensemble of BERT or CANINE, reaches the state-of-the-art evaluation scores on the three out of four datasets, specifically it captures a state-of-the-art F1-score of 0.6149 on one of the commonly used benchmark dataset.

Keywords: Transformer, Tokenization-free, API Calls, Imbalanced, Multiclass, BERT, CANINE, Ensemble, Malware Classification

DENGESİZ SINIF DAĞILIMINA SAHİP ÇOK SINIFLI KÖTÜCÜL YAZILIMLARIN SINIFLANDIRILMASINDA ÖNCEDEN EĞİTİLMİŞ DÖNÜŞTÜRÜCÜ MODELLERİNİN TOPLULUĞU

ÖZET

Kötü amaçlı yazılım ailelerinin sınıflandırılması, bu yazılımların cihazlara, bilgisayarlara veya sistemlere bulaştıktan sonra nasıl bir sürecin gerçekleşebileceğinin kapsamlı bir şekilde anlaşılabilmesi için çok önemlidir. Böylece, kötü amaçlı yazılımların belirlenmesi, siber güvenlik araştırmacılarının ve olay müdahele ekiplerinin kötü amaçlı yazılımlara karşı önlem almalarını ve olası hasarları asgari düzeyde tutmalarını sağlar. Kötü amaçlı yazılımlar tarafından yapılan API çağrı dizileri, kötü amaçlı yazılımların davranışını temsil ettiğinden, makine ve derin öğrenme modelleri tarafından kötü amaçlı yazılım sınıflandırması için yaygın olarak kullanılan özniteliklerdir ancak geleneksel makine ve derin öğrenme modelleri, API çağrıları arasındaki ilişkileri tespit etmekte yetersiz kalmaktadır. Geleneksel makine ve derin öğrenme modellerinin aksine, dönüştürücü tabanlı modeller, API çağrı dizilerini bir bütün olarak işleyip, çok başlı dikkat mekanizmaları ve konumsal gömmeler sayesinde API çağrıları arasındaki ilişkileri öğrenebilmektedir. Yaptığımız deneyler, bir dönüştürücü blok katmanına sahip bir dönüştürücü modelinin, yaygın olarak kullanılan ve temel bir mimari olan LSTM modelinin performansından üstün geldiğini göstermektedir. Önceden eğitilmiş dönüştürücü modellerinden BERT veya CANINE ise, yüksek derecede dengesiz bir sınıf dağılımına sahip kötü amaçlı yazılım ailelerinin sınıflandırılmasında F1 puanına ve AUC puanına göre daha iyi bir performans göstermektedir. Bizim önerdiğimiz, rastgele örnekleme toplama tekniğine dayalı, BERT veya CANINE modellerinin topluluk modeli olan RTF, değerlendirme metrikleri bazında, dört veri setinin üçünde en yüksek skorları elde etmektedir. Aynı zamanda yaygın olarak kullanılan veri setlerinden birinde RTF modeli 0,6149'luk literatürdeki en yüksek F1 puanına ulaşmaktadır.

Anahtar Sözcükler: Dönüştürücü, Bölütlemesiz, API Çağrıları, Dengesiz, Çok sınıflı, BERT, CANINE, Topluluk, Kötücül Yazılım Sınıflandırma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

N	Number of base estimators

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application programming interface
AUC	Area under curve
AvAcc	Classes average accuracy
BERT	Bidirectional encoder representations from transformers
BiLSTM	Bidirectional long short term memory
CANINE	Character architecture with no tokenization in neural encoders
CBA	Class balanced accuracy
CEN	Confusion entropy
CM	Confusion matrix
CNN	Convolutional neural network
DGCNN	Deep graph convolutional neural network
DT	Decision tree
FN	False negative
FP	False positive
FPR	False positive rate
GBM	Gradient boosting method
GMDH	Group method of data handling
GPU	Graphical processing unit
GRU	Gated recurrent unit
KNN	K-Nearest neighbour
LCS	Longest common subsequence
LR	Logistic regression
LSTM	Long short term memory
Malware	Malicious software
MCC	Matthew's correlation coefficient
MLM	Masked language modeling
MLP	Multilayer perceptron
MSA	Multiple sequence alignment
NSP	Next sentence prediction

Opcode	Operation code
PE	Portable executable
RF	Random forest
RNN	Recurrent neural network
ROC	Received operating curve
RTF	Random transformer forest
SVM	Support vector machine
TF-IDF	Term frequency–inverse document frequency
TP	True positive
TPR	True positive rate

1. INTRODUCTION

In recent times, with our dependence on information technologies, the Internet has been widely used by people of all ages. Those who want to quickly meet their daily needs such as online banking, online shopping, health, and transportation-related transactions cause an enormous increase in internet usage as well. This exponential growth of the usage of Internet plays a significant role in making life easier. On the other hand, this situation poses a severe threat as cyber attacks increase drastically in parallel with the growth of the Internet. Among these cyber attacks, malicious software (malware) is the primary weapon for attackers to conduct their malicious activities against a victim's machine such as computer, smartphone, or computer networks in order to disrupt system's functions and gain unauthorized access (Jang-Jaccard and Nepal, 2014; Aslan and Samet, 2020).

Cybercriminals use several ways to spread malware, such as phishing e-mails with malicious links and attachments, text messages, and malicious advertisements etc. According to the state of e-mail security report, 61% of organizations were exposed to e-mail-based ransomware in 2020, with an increase of 10% compared to the previous year (Mimecast, 2021). The average amount spent to recover from a ransomware attack, when factors such as downtime, device, human, and network costs are included, is about $1.85 million (Sophos, 2021). According to another cyber threat report, 5.6 billion malware attacks were carried out in 2020 (SonicWall, 2021).

In lieu all of these findings, one can safely claim that excessive malware, without considering the identification/classification methods, affects many victims destructively. Since the numbers of malicious software and the damages they cause to the institutions are increasing every day, it is crucial to map malware behavior that can be provided by malware family identification so that security researchers and incident responders can speed up the recognition and mitigation processes.

There are two main approaches used the most to detect malware. One of them is the signature-based malware detection method. The signatures, sequences of bytes, created using static, dynamic, or hybrid methods are uniquely located in the database. Whether a given file is malware or not is determined by looking at the unique signature of this file from a predefined database (Shijo and Salim, 2015). Although signature-based methods are the most generally utilized procedure in antivirus programming, since the only predefined list of known malware variants are kept, they are not able to catch previously unidentified malware (Ucci et al., 2019).

The other main malware detection approach is behavior-based method, which examines the behavior and characteristics of a given file and then decides whether the related file is malware, and if it is a malware, then the approach also defines the malware family the file belongs to. Although the effort and time spent and the storage complexity are much more, the unknown attacks can be detected and classified by using behavior-based methodologies better contrary to the signature-based methods (Gibert et al., 2020).

According to the report, among detected malware, 268,362 of them have never been seen before in 2020, with a rise of 74% from the preceding year (SonicWall, 2021). Considering the increasing number of unseen malware over the years, performing a behavior-based approach is more reasonable. This report indicates the significance of developing more innovative and effective malware defense mechanisms to detect and classify unknown malware.

The effectiveness of the malware defense mechanism is directly associated with the right choice of behavioral features exploited from malware. Several features can be extracted from malware due to its diverse nature. Obtaining adequate features is time-consuming for a model. This situation can make learning difficult for a model if some of the features used are non-distinctive (Jindal et al., 2019). In our study, API call sequences are leveraged to classify malware families since these sequences repre-

sent behavioral patterns for each sample. Considering API call sequences, machine learning becomes the primary choice to capture sequence relationships between the sequence elements for malware classification.

Different machine learning algorithms have been used in the literature for malware detection and classification so far (Komatwar and Kokare, 2021; Ucci et al., 2019). Considering the sequence, traditional machine learning models may not be sufficient as the order of the API calls must be preserved. During training, the relations among API calls must be taken into account to successfully predict the malware families of unseen API call sequences. The current deep learning based models, mainly pre-trained transformer models outperform traditional machine learning based approaches for sequential text classification (Li et al., 2020; Minaee et al., 2021).

In this thesis, we have answered the following research questions respectively:

> *RQ.1: What are the suitable classification metrics for imbalanced datasets in multiclass malware classification?*
>
> *RQ.2: What are the appropriate base models for multiclass malware classification based on API call sequences?*
>
> *RQ.3: What are the effects of pre-processing on API call sequences to the model results?*
>
> *RQ.4: What are the effects of tokenizer-based (word piece) pre-trained transformer model (e.g. BERT) and tokenizer-free transformer model (e.g. CANINE) to our model results?*
>
> *RQ.5: What is the effect of ensemble of pre-trained transformer models, BERT and CANINE, which is based on bagging for imbalanced multiclass malware classification?*

Our main contributions through this study, in the light of the answers to our research questions, can be summarized as follows:

- Noticing inconsistent evaluation results due to a logical error in the code of a published article (Schofield, 2021).

- To the best of our knowledge, we have used the pre-trained CANINE transformer model for the first time in the field of malware in this study.
- Again, to the best of our knowledge, a bagging-based ensemble of pre-trained transformer models has been used for the first time in malware classification.
- Our proposed model Random Transformer Forest (RTF), has surpassed the state-of-the-art results obtained in the malware classification.
- We have achieved a state-of-the-art result on one of the well-known API call dataset in the literature (Catak and Yazı, 2019) with our proposed RTF model.

This thesis is structured as follows: Chapter 2 presents the related work. The description of the datasets, base models, pre-trained models, and our proposed model are presented in Chapter 3. The test results are discussed and compared with the related studies in Chapter 4. Chapter 5 presents the papers that contribute to the thesis and lastly, the conclusion and future work are given in Chapter 6.

2. RELATED WORK

Cybercriminals leverage malware to exploit any device or system to steal sensitive data and hence cause enormous problems for victims. Analyzing and classifying incoming malware helps us define the problem and understand how to recover from the damage as quickly as possible.

There are two techniques most commonly used in malware analysis, static analysis and dynamic analysis. Static analysis is a process of malware analysis that analyzes the given malware without running it. Unlike static analysis, a given malware file is executed in an isolated environment to avoid harming the computer system in dynamic analysis.

Malware developers may implement various techniques to evade detection mechanisms such as code obfuscation, dynamic code loading, polymorphism, and metamorphism. For instance, the MD5 hash based detection method can be easily bypassed by malware authors with the methods mentioned above. As these methods cause the binary of the file to change, they also cause a change in the hash of the file. While the hash of the malicious file is changed and the file is defined as benign, the behavior of the file, thus its effect, remains unchanged (Ucci et al., 2019).

Since dynamic analysis requires the execution of a given sample to be monitored and observed in an isolated environment, malware even written with code obfuscation techniques hardly eludes dynamic analysis contrary to static analysis. This identified situation provides dynamic analysis to be more robust than static analysis (Or-Meir et al., 2019).

Performing dynamic analysis requires more time than static analysis and organizations are dealing with millions of attacks carried out in a day. These shortcomings

provide an excellent opportunity for machine learning to collaborate with dynamic and static analysis since machine learning can handle large volumes of data (Fraley and Cannady, 2017).

In the malware detection and classification process, understanding malware behavior is one of the substantial parts of detecting and classifying malware. API calls are obtained by tracing the sequences of calls by way of calling operating system services such as creating a file and allocation of virtual memory by malware samples. Since API call sequences generate specific behavioral patterns and hence represent malware families, they can be considered as one of the most distinguished features among malware families (Ding et al., 2018; Fujino et al., 2015).

Related studies, base models for malware analysis using API calls and transformer-based models on sequence problems will be examined respectively in the rest of Chapter 2.

2.1 Base Models for Malware Analysis using API Calls

Ki et al. (2015) used DNA sequence alignment algorithms, Multiple Sequence Alignment (MSA), and Longest Common Subsequence (LCS) to extract the most critical API call sequence patterns among different malware families and generate a signature-based malware detection mechanism to determine whether a program is a malware or not based on these extracted patterns. The API call sequences determined by the MSA and LCS algorithms can be misleading for the model if sequences get more extended than a preset API call sequence length.

Sundarkumar et al. (2015) proposed a model using text mining and topic modeling for feature extraction and selection processes based on API call sequences. Machine Learning based Group Method of Data Handling (GMDH) method, traditional machine learning models, Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) are compared on two different datasets. Although DT and SVM models outperformed the results, and they suggest

DT for malware detection expert system, the size of the datasets are inadequate to rely on the models.

Kolosnjaji et al. (2016) integrated Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) layer into one neural network architecture. With this model, they have achieved the accuracy score of 89.4%, the precision score of 85.6%, and the recall score of 89.4% to classify malware families. Newly generated subsequences of original API call sequences are given to the model as an input. For each API call sequence, if the same API call is repeated more than two times in a row, only two consecutive identical API calls are included in the resulting sequence. Since their corpus contains only 60 different API calls, they did not set any boundaries. Otherwise, they may have to set a predetermined length to avoid tracking loops and make the model less complex.

Tobiyama et al. (2016) applied two stages of Deep Neural Networks for the malware detection process. The proposed model CNN is used to classify feature images extracted with Long Short-Term Memory (LSTM) model using API calls. Although the authors achieved an Area Under Curve (AUC) score of 96%, since the size of the dataset is relatively small, the score may be misleading.

Mathew and Kumara (2018) leveraged N-gram and Term Frequency–Inverse Document Frequency (TF–IDF) for feature extraction and selection, respectively. The proposed LSTM model is used for binary classification, benign or malware, using API call sequences. The authors reached a 92% accuracy score on unknown test API call sequences.

Xiao et al. (2019) trained two different LSTM networks on system call sequences for both malware and benign Android applications, respectively. The new sequence has been classified by comparing two similarity scores obtained from two different LSTM networks. The LSTM model has been compared with two n-gram models, MALINE and BMSCS, based on accuracy, precision, recall, and False Positive Rate (FPR). They have shown that the LSTM model outperformed MALINE and BMSCS

models.

Catak et al. (2020) tried several models from LSTM to traditional machine learning models, RF, DT, SVM, and K-Nearest Neighbour (KNN), on a dataset containing 7,107 samples of API call sequences generated by them. In multiclass classification, they have achieved the highest F1-score of 47% using the single-layer LSTM model compared to the two-tier LSTM and common machine learning models.

Li and Zheng (2021) generated new API call sequences by applying data pre-processing steps to benchmark dataset (Catak and Yazı, 2019). If any unique API call repeated more than once in the sequence, they kept only one and removed the continuously same API. Similarly, they removed continuously same sub sequences when the length of sub sequences are two or three. They have proposed two different models, LSTM and Gated Recurrent Unit (GRU) models, which are based on RNN architecture. They have significantly increased their precision, recall, and F1-score after data pre-processing.

Oliveira and Sassi (2019) prepared new API call sequences and kept only 100 non-consecutive sequences to avoid repeating API calls loop, similar to feature pre-processing method applied in (Kolosnjaji et al., 2016). They achieved similar AUC score and F1-score results compared to LSTM with their proposed model, which is based on Deep Graph Convolutional Neural Networks (DGCNNs).

LSTM model is widely used as the underlying architecture for malware detection and classification based on API calls, as seen in the above mentioned studies.

2.2 Transformer-Based Models on Sequence Problems

Erciyes and Görür (2021) used several deep learning models from mostly used traditional deep learning methods such as CNN, RNN, LSTM, GRU, and BiLSTM with GLOVE and fastText embedding to pre-trained transformer models for multi-class text classification. For their experiment, they utilized the highly imbalanced

RCV1-v2 dataset, which contains 800,000 news stories. They have shown that transformer models outperformed traditional deep learning models based on F1-score for multiclass text classification.

Paul and Saha (2020) used the BERT model for the cyber-bullying detection task. The proposed model has been tested on three different datasets taken from Twitter, Wikipedia, and FormSpring. Compared to common machine learning models, SVM and Logistic Regression (LR), and deep-learning based models, CNN, RNN + LSTM, and Bidirectional LSTM (BiLSTM), they have achieved higher F1-scores.

Alvares (2021) generated word embeddings for each opcode of malware samples by using Word2Vec and BERT. They classified malware with different classifiers such as LR, SVM, and MLP to see the effect of different word embeddings. They have achieved higher results using BERT for word embeddings with the same set of input parameters and the same set of classifiers based on classification accuracy among five unique malware families distributed almost balanced.

Nassar and Hubballi (2021) proposed transformer-based architecture for detection and classification of malware using opcode sequences of windows executable files. The proposed transformer model has achieved better results compared to Gradient Boosting Method (GBM) and BiLSTM based on accuracy, precision, recall, and F1-score evaluation metrics.

Xu et al. (2021) proposed a pre-trained transformer model, Malbert, which is pre-trained on 15,000 malware and 15,000 benign samples (Ki et al., 2015) first to learn the relationships among API calls. This pre-trained transformer model and existing pre-trained transformer model, *Bert-base-uncased* are fine-tuned on two different datasets for the malware detection process. Pre-trained transformer models have achieved higher results compared to LSTM model and traditional machine learning models based on mostly used evaluation metrics, such as accuracy, precision, recall, and F1-score.

McDonnell et al. (2021) proposed a model, called CyberBert which uses bidirectional transformer architecture for two different tasks, session-based recommendation, and malware classification based on API calls. Compared to the LSTM model and transformer-encoder, a unidirectional model, they have achieved higher F1-scores for binary and multiclass classification with CyberBert.

Oak et al. (2019) leveraged pre-trained BERT transformer model for malware detection, malware, and benign, on Android operating system API calls called by the application. The set of experiments made by the study shows BERT model obtained state-of-the-art results compared to the LSTM model on sequence classification.

Recent surveys (Qiu et al., 2020; Minaee et al., 2021) with the studies mentioned above clearly show that current transformer-based models, mainly pre-trained transformer models fine-tuned on downstream tasks, outperformed traditional machine and deep learning models on sequence classification.

3. METHODOLOGY

In the methodology chapter, firstly, the datasets used in experiments are introduced. Secondly, the most suitable evaluation metrics for highly imbalanced datasets are specified. Then, base model structures, the effect of the pre-processing method, pre-trained transformer models, CANINE and BERT, and the proposed RTF model architectures are explained.

3.1 Datasets

To verify how effectively a model classifies malware, it is necessary to test the model on different malware datasets. Since malware constantly evolves, working on an up-to-date malware dataset is required to assess the effectiveness of proposed models. Comparing several models on a single dataset containing outdated malware samples and highlighting one model might not be reliable. Thus, four different datasets containing API call sequences of malware samples and their corresponding malware families are utilized to evaluate the models used in this study.

3.1.1 Catak

This study obtained sequences of Windows Operating System API calls within the Cuckoo Sandbox isolated environment for each malware file. Malware family labels were determined using unique hash codes of each malware on the Virus Total website. In total, 7,107 samples, which contain hash codes of malware, Windows operating system API call sequences, and their malware family classes, were created (Catak and Yazı, 2019).

3.1.2 Oliveira

42,797 malware and 1,079 benign API call sequences were obtained via Cuckoo Sandbox for dynamic malware analysis. Instead of using whole API call sequences, the first 100 non-consecutive API call sequences were extracted from the parent processes to reduce complexity and detect the malicious pattern as quickly as possible. The generated dataset containing hashcodes, label (malware or benign), and 100 non-consecutive API Calls for each sample has been used for binary malware classification (Oliveira and Sassi, 2019).

Since we are working on a multiclass classification problem, malware families of 42,797 malware samples are determined through virus total. Out of 42,797 malware samples, 2,081 were labeled as "unknown" by virus total. Several malware families hold a small number of malware samples, less than 100. These malware samples are removed since they could be misleading for the models. Thus, the dataset in question has been turned into a multiclass classification case. The compiled dataset used in our study consists of 40,566 malware samples with their API call sequences and malware families.

3.1.3 VirusShare

Unique hash codes represent malware samples obtained from Virus Share. Each unique hash code in text files is passed to Virus Total to learn their corresponding malware families. Python module named PEfile is leveraged to extract API calls from each malware sample. Lastly, malware families having less than 100 samples are removed. Thus, 13,849 malware samples with their corresponding API call sequences and malware families are obtained (Düzgün et al., 2021).

3.1.4 VirusSample

Malware samples taken by Virus Sample are kept with their unique hash code text files. Corresponding malware families and API calls are obtained from Virus Total site and PEfile module, respectively, as in Virus Share. Finally, malware families having less than 100 are removed from the dataset. Therefore, 9,732 malware samples with their corresponding API call sequences and malware families are obtained (Düzgün et al., 2021). Since the malware samples in this dataset consist of the most up-to-date data based on API calls, we also find an opportunity to test our models on recent malware samples. Table 3.1 shows the total samples of malware families for each dataset and Figure 3.1 shows a malware sample in the VirusShare dataset to gain a general insight into datasets.

Table 3.1 Distribution of malware families.

Malware Family	**Oliveira**	**VirusShare**	**Catak**	**VirusSample**
Trojan	31,979	8,919	1,001	6,153
Virus	102	2,490	1,001	2,367
Adware	5,444	908	379	222
Backdoor	135	510	1,001	447
Downloader	1,948	218	1,001	N/A
Worms	N/A	524	1,001	441
Agent	220	165	N/A	102
Ransomware	404	115	N/A	N/A
Dropper	118	N/A	891	N/A
Riskware	216	N/A	N/A	N/A
Spyware	N/A	N/A	832	N/A
Total	**40,566**	**13,849**	**7,107**	**9,732**

Hashcode	API call sequences	Class
de1079a3a4070016591541571c2babc4	'LoadLibraryA,GetProcAddress, VirtualProtect,VirtualAlloc, VirtualFree,ExitProcess, URLDownloadToFileA'	Trojan

Figure 3.1 A malware sample in VirusShare dataset

3.2 RQ.1-) What are the suitable classification metrics for imbalanced datasets in multiclass malware classification?

The degree of imbalance may vary within different domains. One of these domains is malware, as specific malware families are used chiefly in particular periods for cyber attacks.

According to the report released by Malwarebytes: The total number of Trojan detected by Malwarebytes is almost 26 times higher than the total number of Worm in 2018. The total number of Riskware detected by Riskware tools in 2019 was 6,632,817, with a decrease of 35% compared to the previous year. In another chart containing the number of detection of malware families by months, it is seen that the number of Trojan attacks increased dramatically at the beginning of 2019, with the spread of the Emotet, one of the advanced Trojan campaign in that period (Malwarebytes Labs, 2020).

These situations demonstrate that there could be significant differences in the distribution of malware families according to years or even months. Therefore, when the collected malware is classified according to their families, the distribution will vary according to the malware type prevailing at the time of collection and hence lead to imbalance.

For these reasons, almost all of the datasets belonging to malware have an imbalanced class distribution in the literature. Thus, we are required to leverage the most suitable metrics to evaluate our classification performance on imbalanced datasets. The datasets leveraged in our study have highly imbalanced class distribution as expected and shown in Table 3.1.

Evaluation metrics are one of the crucial steps to assess model performance. An incorrectly chosen evaluation metric can make a poor performance algorithm seems effective. The metrics used to evaluate a model performance may vary for balanced and imbalanced datasets. For instance, using accuracy metric for a balanced

dataset may provide an objective evaluation, yet may not be the right choice for an imbalanced dataset as it has a bias against the majority class. Taking malware classification for example. Assume there are six different classes in the dataset, and 95% of the samples belong to Trojan. In this case, a dummy model that predicts all samples in the test data as Trojan will achieve an accuracy score of 95% even though it does not predict any other classes correctly. It may not always be correct to use the most widely preferred evaluation metric without examining the distribution of classes in the dataset for the reasons mentioned above.

Recently, researchers have started to use Matthew's Correlation Coefficient (MCC) to evaluate model performance on imbalanced datasets. Although this metric was previously used mostly in biomedical research, it has now been used in many areas, including malware classifications (Kim et al., 2021; Jahromi et al., 2020) yet according to the experimental results to investigate the behavior of MCC metric, MCC is not suitable for directly applying on imbalanced datasets (Zhu, 2020). Empirical research conducted on 54 imbalanced datasets demonstrates that the AUC score is more discriminating than MCC (Halimu et al., 2019). Most frequently used metrics to assess model performance for multiclass classification tasks have been shown to be inadequate on imbalanced datasets such as Precision, Recall, MCC, Confusion Entropy (CEN), Classes Average Accuracy (AvAcc), and Class Balanced Accuracy (CBA) (Branco et al., 2017).

Although the choice of right metrics is still an open issue, following the searches to find the most suitable metrics used for multiclass classification on imbalanced datasets, we have used AUC, which is a summary of probability curve, Receiver Operating Characteristic (ROC), based on FPR and True Positive Rate (TPR), as an evaluation metric. On the other hand, F1-score has been used to be comparable with studies conducted on one of the well-known datasets in the literature (Catak et al., 2020).

The equation (3.1) and (3.2) define the recall and precision metrics respectively. The

equation (3.3) defines F1-score in terms of precision and recall. Also, the equation (3.3) contains the explicit form of the formula in terms of True Positive (TP), False Negative (FN), and False Positive (FP).

$$Recall = \frac{TP}{TP + FN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$F_1 score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3.3}$$

3.3 Base Models

LSTM and One-layer Transformer Block Based Transformer architecture are leveraged as base models for malware classification based on *RQ.2*. For the rest of the thesis, One-layer Transformer Block Based Transformer architecture is referred to as *Transformer* model.

3.3.1 LSTM-Based Malware Classification

Recurrent Neural Networks (RNNs) have a structure that uses recurrent relation, which is a situation of performing the same step, processing current output depends on the previously computed hidden state, for each element of a sequence repeatedly. In these structures, information is retained through the previous hidden state, and hence processing continues in time steps. The recursion between sequence elements hinders parallelization during the training phase and consequently causes a longer run time for training.

LSTM can learn relatively long-term dependencies compared to other RNNs because it provides deeper processing of hidden states through specific units. This situation causes an increase in the number of parameters used for training. Besides, since

LSTM has a recursive structure like other RNNs inherently and hence can not be trained in parallel, the training period may take relatively longer compared to other RNNs (Hochreiter and Schmidhuber, 1997).

Since our purpose is to classify malware families, fully connected layer output that captured the information from the LSTM networks, is given to the softmax layer for multiclass classification.

The standard LSTM network is preferred as one of the base models for comparison since it has been used widely as a base network and performed successfully for several malware classification problems using API call sequences (Berman et al., 2019).

3.3.2 Transformer Model

Transformer model is a recently used network architecture that designed to overcome the deficiencies of sequence-to-sequence neural network approaches such as LSTM and RNN for the sequence modeling and transduction problems in 2017 (Vaswani et al., 2017).

Since transformer-based architectures avoid recursion, they can overcome the parallelization problem that both the LSTM and other RNNs suffer from. In traditional sequence-to-sequence architectures the information coming from the previously hidden state is processed recursively to capture dependencies. On the contrary, since the transformer models refrain from recurrence and convolution, positional encodings are used to preserve the order of the sequence and provide position-related information of the tokens in the sequence. Transformer models leverage the attention mechanism to caption and preserve long-term dependencies for the sequences processed as a whole. Positional information is retained with attention layers instead of the recurrent and convolutional layers in transformer model (Vaswani et al., 2017). Figure 3.2 shows the Transformer model architecture utilized in this study.

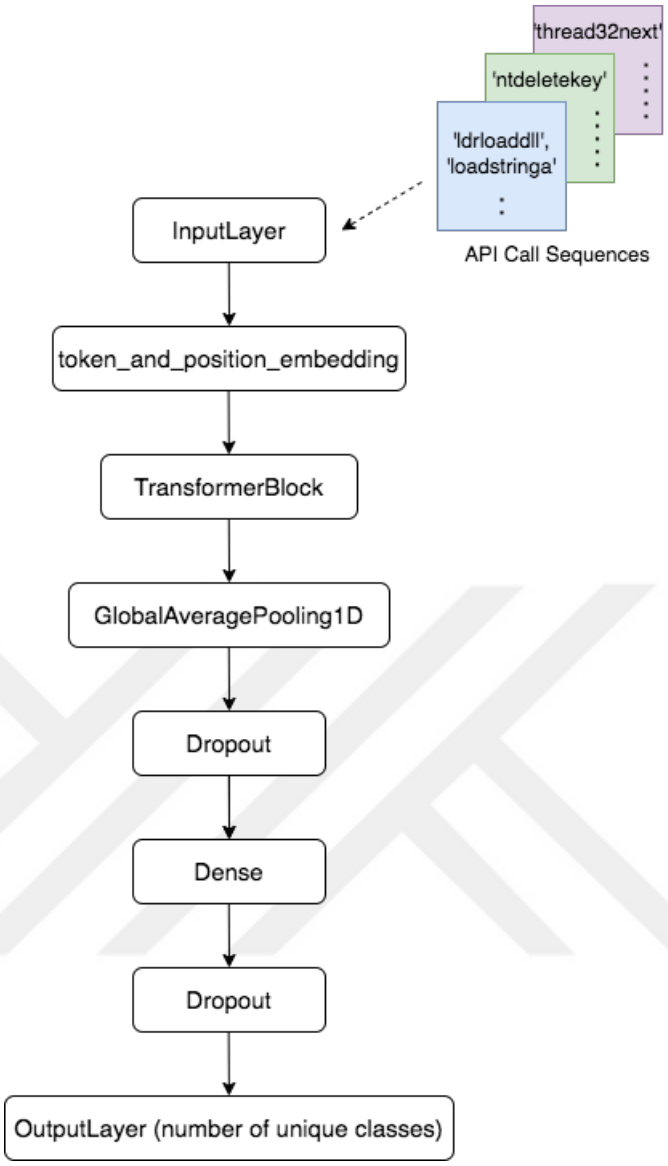


Figure 3.2 Transformer model architecture.

3.4 Pre-processing Method on Datasets

Each API call sequence is pre-processed, similar to the steps taken part in (Li and Zheng, 2021). Pre-processing part consists of 3 main steps.

In the first step, for any API call in a sequence that repeated more than one time in a row, the continuously same API calls are removed from the sequence. This pre-processing step generates a new sequence that does not consist of the consecutive same API call. In the second and third steps respectively, repetitive binary and

triple sub-sequences are removed from the new sequence created by the first step. The following Figure 3.3 shows the pre-processing step outputs respectively based on randomly given sequences.

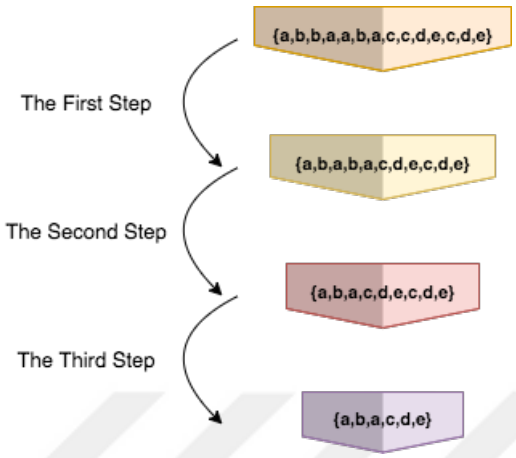


Figure 3.3 The outputs of pre-processing steps.

These pre-processing steps have not been applied to the Oliveira dataset as this dataset has already given pre-processed and limited to the 100 non-consecutive API calls only. On the other hand, although the pre-processing steps have been applied to the VirusSample and VirusShare datasets as well, only one sample out of 9,732 samples for VirusSample dataset and only two samples out of 13,849 samples for VirusShare dataset are affected. Of these affected samples, only two or three API calls are affected. Thus, we have continued with the original API call sequences of VirusSample and VirusShare datasets. After performing pre-processing steps on Catak dataset, only 11 API call sequences remained constant. The effect of the pre-processing steps on the Catak dataset can easily be seen by the Figure 3.4 shown below.

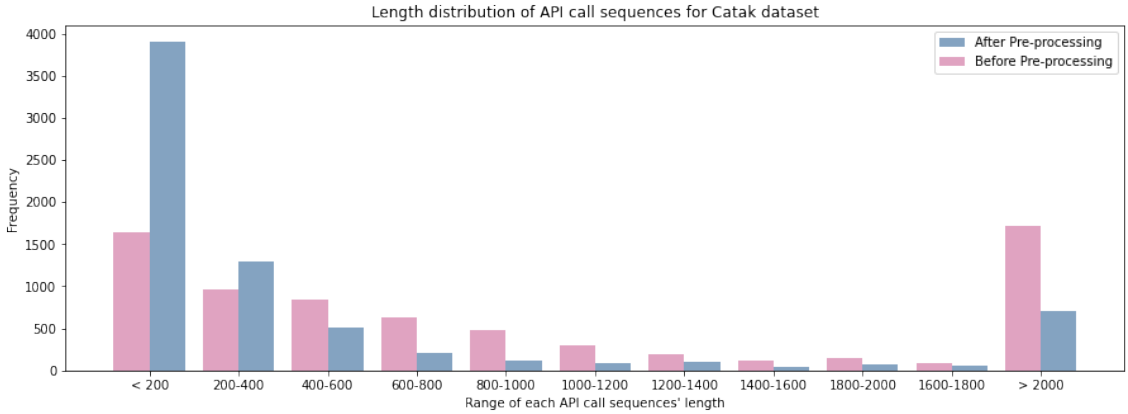


Figure 3.4 The effect of pre-processing on Catak dataset.

After pre-processing steps, the number of samples where the length of the API call sequences is less than 200 has increased more than twice, and the number of samples where the length of the API call sequences is more than 200 has decreased more than twice. These changes clearly show the effect of pre-processing steps on the Catak dataset. Finally, after performing all the pre-processing steps to the Catak set only, the effect of the pre-processing is examined with model performances.

3.5 CANINE and BERT

Large-scale pre-trained models have recently become very popular in the field of artificial intelligence. Due to the model previously trained on large-scale data, captured information can be used for specific tasks that utilize the pre-trained model by fine-tuning (Han et al., 2021). This study utilizes two different pre-trained models architectures, BERT and CANINE.

3.5.1 BERT

BERT, Bidirectional Encoder Representations from Transformers, is a language model that uses transformer architecture which is pre-trained on Wikipedia and Book Corpus of unlabelled text (Devlin et al., 2018).

BERT preserves the semantic content thanks to the masked language modeling (MLM) and next sentence prediction (NSP) unsupervised tasks, which enables to generate deep bidirectional representations while pre-training. BERT uses Word-piece tokenization to create a token vocabulary that consists of learned representations of the words. Figure 3.5 shows BERT architecture in details.

3.5.2 CANINE

CANINE, Character Architecture with No tokenization In Neural Encoders, is a tokenizer-free pre-trained encoder model that is designed to overcome the shortcomings of the tokenization process such as word-piece and sentence-piece tokenization (Clark et al., 2021). For example, a pre-trained model that uses specific tokenization may not be convenient for specialized domains. Boukkouri et al. (2020) showed that word-piece tokenization based pre-training strategy is not well-suited compared to character-piece when fine-tuned on medical data.

Similar to BERT (Devlin et al., 2018), CANINE is pre-trained on the MLM and NSP tasks as well. Unlike commonly used pre-trained models, CANINE uses neural encoders that encode the sequence of characters or optionally sub-words are used as a soft inductive bias without doing explicit tokenization on input data. The CANINE structure is shown in Figure 3.6.

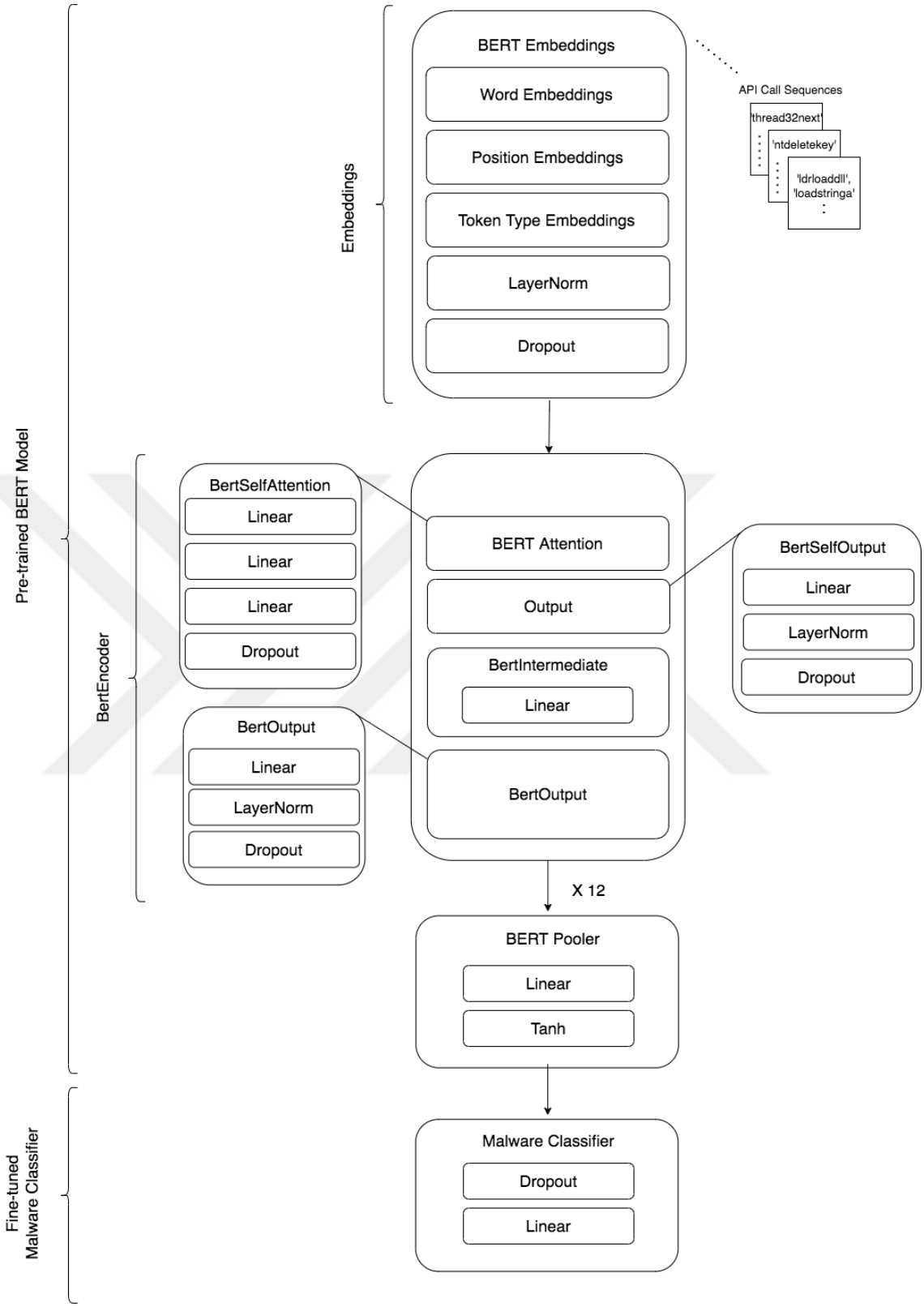


Figure 3.5 BERT architecture.

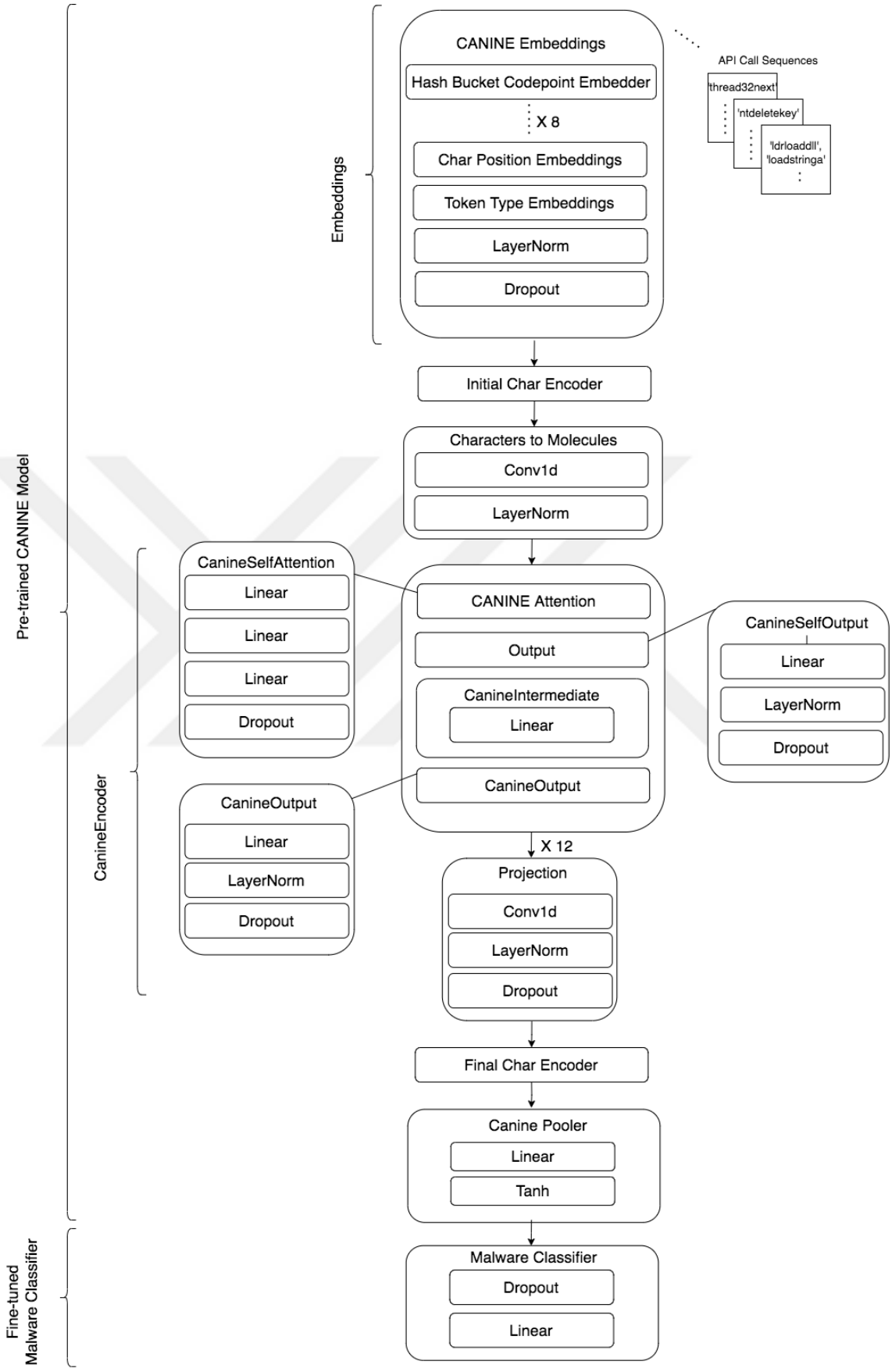


Figure 3.6 CANINE architecture.

In general, the BERT model is widely used by many studies for malware classification, as shown in the *Related Work*. We have assumed that the tokenization-free strategy used by the CANINE model might be well-suited for API calls since an API call such as 'ldrloaddll' may not be appropriate for word-tokenization. Therefore, we have included the CANINE model in our study.

Thus, BERT, and two different CANINE models, CANINE-C (Pre-trained with autoregressive character loss), and CANINE-S (Pre-trained with subword loss) pre-trained transformer models are leveraged regarding the *RQ.4*.

3.6 Proposed Model: Random Transformer Forest (RTF)

There are several important studies that show the success of using different ensemble types of pre-trained transformer models such as stacking and majority voting of heterogeneous pre-trained transformer models on varying downstream tasks. (Marcinczuk, 2021; Morio et al., 2020; Malla and Alphonse, 2021). Unlike these type of ensemble models, Random Transformer Forest (RTF) is a bagging-based ensemble model inspired from the Random Forest (RF) machine learning model (Breiman, 2001). Similar to RF, using an ensemble of pre-trained transformer models is assumed to increase classification performance on highly imbalanced malware datasets rather than using a single pre-trained transformer model (Çayır et al., 2021; Kobayashi et al., 2021).

The training phase requires creating N different training subsets by using the bootstrap sampling method from the original training set. The malware class distribution coming from the original training set must be preserved in the resampling step due to the highly imbalanced class distribution. After the resampling step, each training subset is used to fine-tune the pre-trained transformer model. Each pre-trained transformer model has the same structure. Therefore base estimators are homogeneous.

In the testing phase, each fine-tuned transformer model takes a given malware API

call sequence, and the class probabilities of each fine-tuned transformer model are aggregated by taking the average. For the majority voting, the final prediction is accepted as the malware family, which takes the highest probability coming from the aggregation part. Figure 3.7 shows the structure of RTF model.

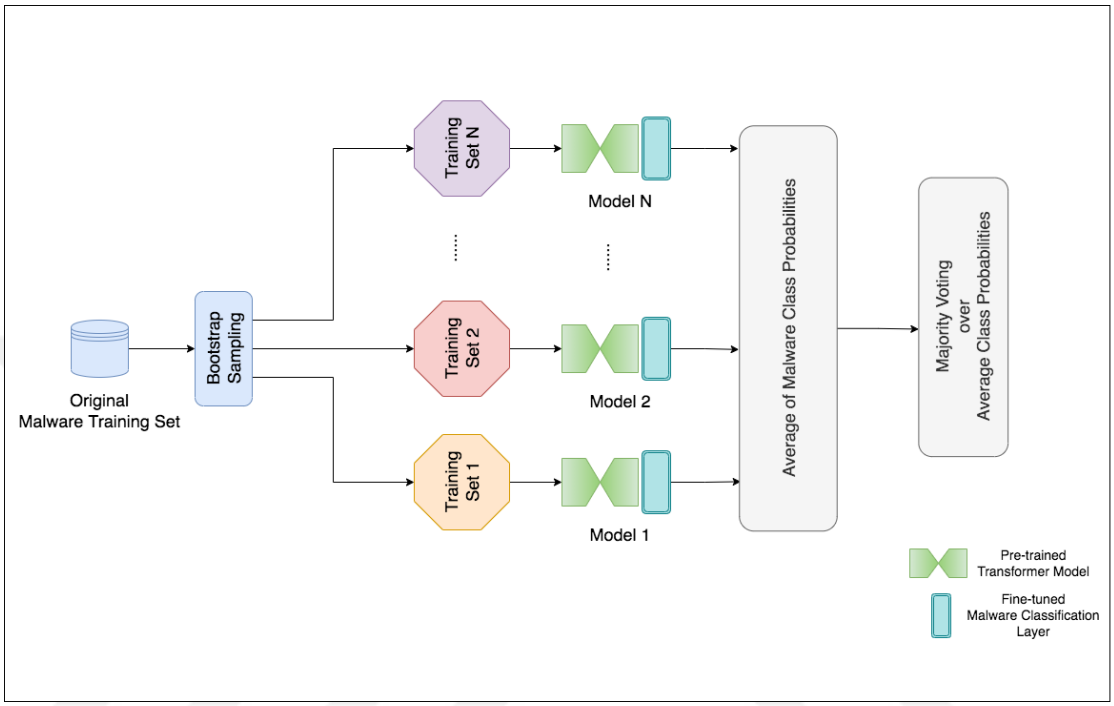


Figure 3.7 The proposed RTF model architecture.

4. EXPERIMENT AND RESULTS

All conducted experiments will be clarified respectively regarding the predetermined Research Questions except *RQ.1* which is elaborated in *Methodology* part.

4.1 RQ.2-) What are the appropriate base models for multiclass malware classification based on API call sequences?

In this part of the study, Transformer and LSTM model are leveraged as base architectures for comparison on four datasets.

All datasets used to evaluate the base model performances are divided into three parts, training, validation, and testing. 20% of the original datasets are allocated for testing. The splitting data process is performed in a stratified way as to preserve class distribution is one of the crucial steps on highly imbalanced datasets. Considering the imbalanced distribution of the classes, we have leveraged the class weight approach to give different weights to both the majority and minority classes so that class weights are taken into account by training algorithms.

Stratified 10 Fold strategy is used on training data for each dataset, and 10% of training data is used for validation for each iteration. Thus, we guarantee that each fold has the same distribution of malware families and ensure that every sample from the dataset has the chance of appearing in both training and validation data. Standard deviation and mean of 10 validation results are calculated for each evaluation metric used in our study for robust interpretation. We have provided a dummy classifier as a simple baseline since class distribution of each dataset is imbalanced. We have used the "most frequent" strategy for our dummy classifiers. Base model comparison results on each dataset is shown in the tables 4.1 to 4.4.

Table 4.1 Base model comparison results for Catak dataset.

Base Model	**F1-score**	**AUC score**
Validation Mean Scores		
LSTM	0.4873 ± 0.0126	0.7887 ± 0.0128
Transformer	**0.5689 ± 0.0578**	**0.8676 ± 0.0329**
Test Scores		
LSTM	0.4638	0.7885
Transformer	**0.5042**	**0.8246**
Dummy	0.0308	0.5000

Table 4.2 Base model comparison results for Oliveira dataset.

Base Model	**F1-score**	**AUC score**
Validation Mean Scores		
LSTM	0.5570 ± 0.0182	0.8853 ± 0.0142
Transformer	**0.5792 ± 0.0379**	**0.9280 ± 0.0142**
Test Scores		
LSTM	0.5637	0.8844
Transformer	**0.5650**	**0.8855**
Dummy	0.0980	0.5000

Table 4.3 Base model comparison results for VirusSample dataset.

Base Model	**F1-score**	**AUC score**
Validation Mean Scores		
LSTM	0.7690 ± 0.0419.	0.9656 ± 0.0110
Transformer	**0.8070 ± 0.0323**	**0.9885 ± 0.0055**
Test Scores		
LSTM	0.7531	**0.9701**
Transformer	**0.7548**	0.9680
Dummy	0.1291	0.5000

Table 4.4 Base model comparison results for VirusShare dataset.

Base Model	**F1-score**	**AUC score**
Validation Mean Scores		
LSTM	0.7121 ± 0.0231	0.9274 ± 0.0130
Transformer	**0.7641 ± 0.0297**	**0.9700 ± 0.0182**
Test Scores		
LSTM	0.7071	0.9298
Transformer	**0.7125**	**0.9350**
Dummy	0.0980	0.5000

Considering the two base models, LSTM and Transformer model, the standard deviation of the mean validation scores of evaluation metrics for the Transformer model is higher. Even in this case, we get higher results on unseen test data.

Evaluation results demonstrate that the Transformer model is more reasonable to continue with compared to the LSTM model.

4.2 RQ.3-) What are the effects of pre-processing on API call sequences to the model results?

Data pre-processing steps mentioned in *Pre-processing Method on datasets* part are applied to the Catak dataset only as pre-processing has no effect on VirusSample and VirusShare datasets, and Oliveira dataset has already been pre-processed and limited with 100 non-consecutive API calls.

Original API call sequences and pre-processed API call sequences on Catak dataset have been compared with the LSTM and Transformer model.

Table 4.5 shows the comparison results of Original API call sequences and pre-processed API call sequences.

Table 4.5 Comparison of the original and pre-processed Catak datasets.

Catak Dataset	**F1-score**	**AUC score**
On Original API Call Sequences		
LSTM	0.4638	0.7885
Transformer	0.5042	0.8246
On Pre-processed API Call Sequences		
LSTM	**0.5020**	**0.8156**
Transformer	**0.5106**	**0.8372**

Evaluation results show that pre-processing step outperforms for Catak dataset. Both the AUC score and F1-score have increased after the pre-processing steps for both LSTM and Transformer model. New sequences generated after pre-processing steps are leveraged for the following experiments, pre-trained models and RTF model for the Catak dataset.

4.3 RQ.4-) What are the effects of tokenizer-based (word piece) pre-trained transformer model (e.g. BERT) and tokenizer-free transformer model (e.g. CANINE) to our model results?

In this part, Due to the large number of parameters used in pre-trained models, 20% of the training data is allocated for validation instead of the stratified k fold strategy. The best model that gives higher scores for the CANINE model is included only in the result tables between the sub-word and character strategy.

The comparison of the pre-trained models is shown with the RTF model results to see the comparison clearly.

4.4 RQ.5-) What is the effect of ensemble of pre-trained transformer models, BERT and CANINE, which is based on bagging for imbalanced multiclass malware classification?

In the RTF model N different training subsets, thus N different base estimators are utilized to fine-tune the N different pre-trained BERT or CANINE model. We have tried several combinations of N and pre-trained transformer models, BERT and CANINE, for each dataset. As a result of several trials, the combinations that provided the best scores are accepted as our RTF score. Table 4.6 shows the best combination for each dataset and model comparison results on each dataset are shown in tables 4.7 to 4.10.

Table 4.6 Best parameters for RTF model.

Dataset	Number of Base Estimators (N)	Pre-trained Model
Catak	6	BERT
Oliveira	2	BERT
VirusSample	10	CANINE-S
VirusShare	5	CANINE-S

Table 4.7 All model comparison on Catak dataset.

Model	F1-score	AUC score
Transformer	0.5106	0.8372
CANINE-S	0.5633	0.8339
BERT	0.5919	0.8735
RTF	**0.6149**	**0.8818**

Table 4.8 All model comparison on Oliveira dataset.

Model	F1-score	AUC score
Transformer	**0.5650**	**0.8855**
CANINE-S	0.4725	0.8636
BERT	0.4839	0.8321
RTF	0.4745	0.8058

Table 4.9 All model comparison on VirusSample dataset.

Model	F1-score	AUC score
Transformer	0.7548	0.9680
CANINE-C	0.7893	0.9570
BERT	0.7759	0.9690
RTF	**0.8170**	**0.9714**

Table 4.10 All model comparison on VirusShare dataset.

Model	F1-score	AUC score
Transformer	0.7125	0.9350
CANINE-S	0.7064	0.9286
BERT	0.7145	0.9364
RTF	**0.7459**	**0.9436**

According to the results, at least one of the pre-trained transformer models, CANINE or BERT, surpassed Transformer Model, and the RTF model obtained the highest scores on three out of four datasets.

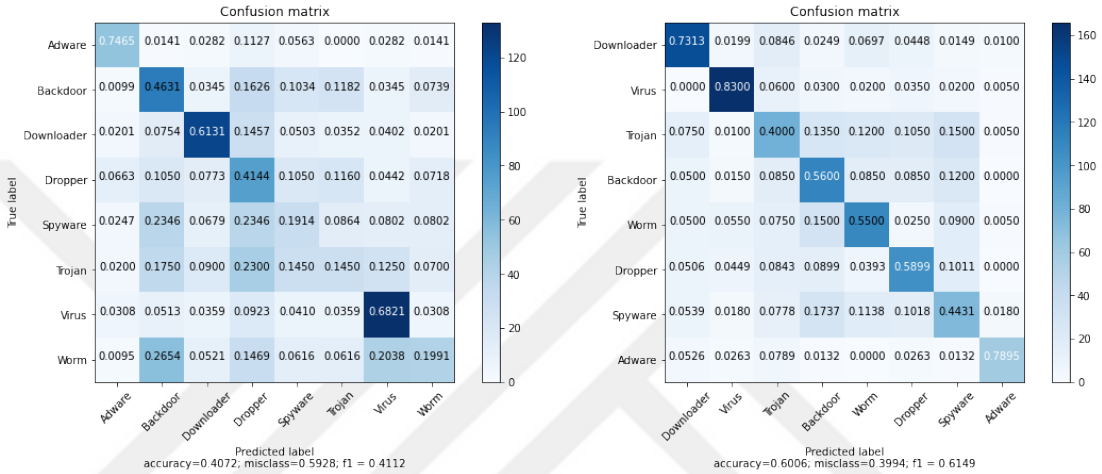
Among the four datasets we have experimented on, only the Catak dataset is comparable since VirusShare and VirusSample datasets are newly published and Oliveira dataset is transformed to a multiclass problem by us.

Schofield (2021) has proposed a model for multiclass classification on Catak Dataset. In this study, all the samples in the Catak dataset are shuffled, and 80% of the dataset is allocated for the train set. Then, all the samples in the Catak dataset are shuffled *again* and 20% of the dataset is allocated for the test set. The logical error made here is whole samples are shuffled twice. This situation causes the test part to have some samples which exactly fall into the train part. Thus, the model might test what it learned from the train. We have performed a test on the code shared with us by the authors Schofield (2021) from the GitHub link [1] . We have performed a test to divide the whole dataset into the training and testing dataset with the exact code script performed by authors for their study. Finally, we realized that 1,117 of 1,371 samples allocated for the test set intersect with the train set. For these reasons, evaluation scores obtained by this article are not taken into account to compare our results. The logical error has been reported to the article authors.

To the best of our knowledge highest F1-score obtained on the Catak dataset for multiclass classification is 0.57 (Li and Zheng, 2021) compared to the baseline score of 0.47 obtained by the Catak dataset creators (Catak et al., 2020). Although the F1-score reported by Catak et al. (2020) is 0.47, the calculated F1-score from the given confusion matrix (CM) in (Catak et al., 2020) is 0.41 as in Figure 4.1a. The 20% of original Catak dataset is allocated as unseen test data for RTF experiments like in (Catak et al., 2020). Catak et al. (2020) showed their CM of LSTM model results on unseen test dataset. Their CM is referred to as source CM as this is the

[1] `https://github.com/MattScho/MalwareClassificationCNN`

first study performed on Catak dataset. Since only this study contains CM, we have compared their CM with our proposed RTF model CM on unseen test data. Among the experimental studies conducted on the Catak dataset for multiclass classification (Li and Zheng, 2021; McDonnell et al., 2021; Catak et al., 2020) our proposed RTF model has surpassed and reached the state-of-the-art F1-score of 0.6149 as shown in Figure 4.1b.



(a) SOURCE CM (b) RTF CM

Figure 4.1 Comparison of confusion matrix on Catak dataset.

4.5 Experimental Setup

We have utilized the google-cloud colab for base model comparison using Keras (Chollet et al., 2018). We have worked on Tesla K80 GPU with 12 GB available RAM provided by the colab platform. Since the pre-trained transformer models and hence RTF experiments faced GPU memory exhausted error on colab, we tested our remaining experiments on the computer with 2 GPU (GeForce GTX 1080 Ti) with 12 GB RAM per each GPU. For pre-trained transformer models and RTF model, the PyTorch framework (Paszke et al., 2017) is leveraged with pre-trained models taken from HuggingFace (Wolf et al., 2019). All jupyter notebook files that contain source codes regarding the research questions, from base model comparison to RTF model, can be found in the Github repository [2] .

[2] `https://github.com/Ferhat94/Random-Transformer-Forest`

5. INCLUDED PAPERS AND CONTRIBUTIONS

In this part of the thesis, our papers contributed to this study are shown.

5.1 Paper-I

Website Category Classification Using Fine-tuned BERT Language Model (Demirkıran et al., 2020).

5.1.1 Summary

The contents on the Word Wide Web is expanding every second providing web users a rich content. However, this situation may cause web users harm rather than good due to its harmful or misleading information. The harmful contents can contain text, audio, video, or image that can be about violence, adult contents, or any other harmful information. Especially young people may readily be affected with these harmful information psychologically. To prevent youth from these harmful contents, various web filtering techniques, such as keyword filtering, Uniform Resource Locator (URL) based filtering, Intelligent analysis, and semantic analysis, are used. We propose an algorithm that can classify websites, which may contain adult contents, with 67.81% (BERT) accuracy among 32 unique categories. We also show that a BERT model gives higher accuracy than both the Sequential and Functional API models when used for text classification.

5.1.2 Contributions

In this study, a highly imbalanced dataset is leveraged for multiclass website category classification based on the the descriptions of the categories. This study has shown us the success of the pre-trained transformer model, BERT, over LSTM for sequence-

based text classification. From this point of view, we have seen the promise of pre-trained transformer models on sequential data, and leveraged BERT and CANINE for malware family classification using API calls as these API calls are inherently sequential.

5.2 Paper-II

New Datasets for Dynamic Malware Classification (Düzgün et al., 2021).

5.2.1 Summary

Nowadays, malware and malware incidents are increasing daily, even with various anti-viruses systems and malware detection or classification methodologies. Many static, dynamic, and hybrid techniques have been presented to detect malware and classify them into malware families. Dynamic and hybrid malware classification methods have advantages over static malware classification methods by being highly efficient. Since it is difficult to mask malware behavior while executing than its underlying code in static malware classification, machine learning techniques have been the main focus of the security experts to detect malware and determine their families dynamically. The rapid increase of malware also brings the necessity of recent and updated datasets of malicious software. We introduce two new, updated datasets in this work: One with 9,795 samples obtained and compiled from VirusSamples and the one with 14,616 samples from VirusShare. This paper also analyzes multi-class malware classification performance of the balanced and imbalanced version of these two datasets by using Histogram-based gradient boosting, Random Forest, Support Vector Machine, and XGBoost models with API call-based dynamic malware classification. Results show that Support Vector Machine, achieves the highest score of 94% in the imbalanced VirusSample dataset, whereas the same model has 91% accuracy in the balanced VirusSample dataset. While XGBoost, one of the most common gradient boosting-based models, achieves the highest score of 90% and 80%.in both versions of the VirusShare dataset. This paper also presents the base-

line results of VirusShare and VirusSample datasets by using the four most widely known machine learning techniques in dynamic malware classification literature. We believe that these two datasets and baseline results enable researchers in this field to test and validate their methods and approaches.

5.2.2 Contributions

Two datasets, VirusSample and VirusShare, prepared in this study are leveraged to test our models from LSTM to our proposed RTF model with two different datasets as mentioned in *Datasets*. To our best knowledge, VirusSample dataset have the most up-to-date malware observations based on API calls. Thus, we have the opportunity to work on an up-to-date malware dataset, which is crucial considering that malware evolves over time. Therefore, we find an opportunity to test performance of our models reliably due to the total of four different datasets.

6. CONCLUSIONS AND FUTURE WORK

In this study, we have leveraged several deep learning models for highly imbalanced multiclass malware classification based on API calls, which are inherently sequence problems. We have assessed the performance of our models with AUC score and F1-score evaluation metrics as the four datasets used in this study are imbalanced.

Our evaluation results demonstrate that the Transformer Model with one transformer block layer has achieved slightly better results than the LSTM model. Moreover, the pre-trained transformer models, BERT or CANINE, outperformed one transformer block layer Transformer Architecture.

The CANINE model has been used for the first time in the field of malware classification in this study. We have reached a state-of-the-art results on the VirusShare and VirusSample dataset with a bagging-based ensemble of the CANINE model. Therefore, we have demonstrated the success of the CANINE model with the strength of RTF on two out of four datasets.

We have achieved a state-of-the-art F1-score of 0.6149 on the Catak dataset with the power of bagging-based ensemble of BERT model and pre-processing since pre-processing steps have enabled us to increase our results significantly on the Catak dataset. In addition, the F1-score of 0.6149 obtained on the well-known benchmark Catak dataset has proved the success of our proposed RTF Model. In general, our proposed RTF Model has obtained state-of-the-art results on three out of four datasets.

This study can be extended by integrating our proposed ensemble model with the AUC maximization paradigm (Yuan et al., 2021). We believe we may increase our results in this way.

REFERENCES

Alvares, J. L. (2021). Malware classification with bert. Master's thesis, San José State University.

Aslan, Ö. A. and R. Samet (2020). A comprehensive review on malware detection approaches. *IEEE Access 8*, 6249–6271.

Berman, D. S., A. L. Buczak, J. S. Chavis, and C. L. Corbett (2019). A survey of deep learning methods for cyber security. *Information 10*(4), 122.

Boukkouri, H. E., O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii (2020). Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. *arXiv preprint arXiv:2010.10392*.

Branco, P., L. Torgo, and R. P. Ribeiro (2017). Relevance-based evaluation metrics for multi-class imbalanced domains. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 698–710. Springer.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Catak, F. O. and A. F. Yazı (2019). A benchmark api call dataset for windows pe malware classification. *arXiv preprint arXiv:1905.01999*.

Catak, F. O., A. F. Yazı, O. Elezaj, and J. Ahmed (2020). Deep learning based sequential model for malware analysis using windows exe api calls. *PeerJ Computer Science 6*, e285.

Çayır, A., U. Ünal, and H. Dağ (2021). Random capsnet forest model for imbalanced malware type classification task. *Computers & Security 102*, 102133.

Chollet, F. et al. (2018). Keras: The python deep learning library. *Astrophysics Source Code Library*, ascl–1806.

Clark, J. H., D. Garrette, I. Turc, and J. Wieting (2021). Canine: Pre-training an efficient tokenization-free encoder for language representation. *arXiv preprint arXiv:2103.06874*.

Demirkıran, F., A. Çayır, U. Ünal, and H. Dağ (2020). Website category classification using fine-tuned bert language model. In *2020 5th International Conference on Computer Science and Engineering (UBMK)*, pp. 333–336. IEEE.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, Y., X. Xia, S. Chen, and Y. Li (2018). A malware detection method based on family behavior graph. *Computers & Security 73*, 73–86.

Düzgün, B., A. Çayır, F. Demirkıran, C. N. Kayha, B. Gençaydın, and H. Dağ (2021). New datasets for dynamic malware classification. *arXiv preprint arXiv:2111.15205*.

Erciyes, N. E. and A. K. Görür (2021). Deep learning methods with pre-trained word embeddings and pre-trained transformers for extreme multi-label text classification. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 50–55. IEEE.

Fraley, J. B. and J. Cannady (2017). The promise of machine learning in cybersecurity. In *SoutheastCon 2017*, pp. 1–6. IEEE.

Fujino, A., J. Murakami, and T. Mori (2015). Discovering similar malware samples using api call topics. In *2015 12th annual IEEE consumer communications and networking conference (CCNC)*, pp. 140–147. IEEE.

Gibert, D., C. Mateu, and J. Planes (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications 153*, 102526.

Halimu, C., A. Kasem, and S. S. Newaz (2019). Empirical comparison of area under roc curve (auc) and mathew correlation coefficient (mcc) for evaluating machine learning algorithms on imbalanced datasets for binary classification. In *Proceedings of the 3rd international conference on machine learning and soft computing*, pp. 1–6.

Han, X., Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang, et al. (2021). Pre-trained models: Past, present and future. *AI Open*.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Jahromi, A. N., S. Hashemi, A. Dehghantanha, K.-K. R. Choo, H. Karimipour, D. E. Newton, and R. M. Parizi (2020). An improved two-hidden-layer extreme learning machine for malware hunting. *Computers & Security 89*, 101655.

Jang-Jaccard, J. and S. Nepal (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences 80*(5), 973–993.

Jindal, C., C. Salls, H. Aghakhani, K. Long, C. Kruegel, and G. Vigna (2019). Neurlux: Dynamic malware analysis without feature engineering. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 444–455.

Ki, Y., E. Kim, and H. K. Kim (2015). A novel approach to detect malware based on api call sequence analysis. *International Journal of Distributed Sensor Networks 11*(6), 659101.

Kim, M., D. Kim, C. Hwang, S. Cho, S. Han, and M. Park (2021). Machine-learning-based android malware family classification using built-in and custom permissions. *Applied Sciences 11*(21), 10244.

Kobayashi, S., J. von Oswald, and B. Grewe (2021). On the reversed bias-variance tradeoff in deep ensembles. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*.

Kolosnjaji, B., A. Zarras, G. Webster, and C. Eckert (2016). Deep learning for classification of malware system call sequences. In *Australasian joint conference on artificial intelligence*, pp. 137–149. Springer.

Komatwar, R. and M. Kokare (2021). A survey on malware detection and classification. *Journal of Applied Security Research 16*(3), 390–420.

Li, C. and J. Zheng (2021). Api call-based malware classification using recurrent neural networks. *Journal of Cyber Security and Mobility*, 617–640.

Li, Q., H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He (2020). A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364* .

Malla, S. and P. Alphonse (2021). Covid-19 outbreak: An ensemble pre-trained deep learning model for detecting informative tweets. *Applied Soft Computing 107*, 107495.

Malwarebytes Labs (2020). State of malware report. Technical report, Malwarebytes.

Marcinczuk, M. (2021). Punctuation restoration with ensemble of neural network classifier and pre-trained transformers. *Proceedings ofthePolEval2021Workshop*, 47.

Mathew, J. and M. A. Kumara (2018). Api call based malware detection approach using recurrent neural network—lstm. In *International Conference on Intelligent Systems Design and Applications*, pp. 87–99. Springer.

McDonnell, S., O. Nada, M. R. Abid, and E. Amjadian (2021). Cyberbert: A deep dynamic-state session-based recommender system for cyber threat recognition. In *2021 IEEE Aerospace Conference (50100)*, pp. 1–12. IEEE.

Mimecast (2021). The state of email security. Technical report, Mimecast.

Minaee, S., N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao (2021). Deep learning–based text classification: A comprehensive review. *ACM Computing Surveys (CSUR) 54* (3), 1–40.

Morio, G., T. Morishita, H. Ozaki, and T. Miyoshi (2020). Hitachi at semeval-2020 task 11: An empirical study of pre-trained transformer family for propaganda detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pp. 1739–1748.

Nassar, F. and N. Hubballi (2021). *Malware detection and classification using transformer-based learning.* Ph. D. thesis, Discipline of Computer Science and Engineering, IIT Indore.

Oak, R., M. Du, D. Yan, H. Takawale, and I. Amit (2019). Malware detection on highly imbalanced data through sequence modeling. In *Proceedings of the 12th ACM Workshop on artificial intelligence and security*, pp. 37–48.

Oliveira, A. and R. Sassi (2019). Behavioral malware detection using deep graph convolutional neural networks. *TechRxiv*.

Or-Meir, O., N. Nissim, Y. Elovici, and L. Rokach (2019). Dynamic malware analysis in the modern era—a state of the art survey. *ACM Computing Surveys (CSUR) 52*(5), 1–48.

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). Automatic differentiation in pytorch.

Paul, S. and S. Saha (2020). Cyberbert: Bert for cyberbullying identification. *Multimedia Systems*, 1–8.

Qiu, X., T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 1–26.

Schofield, M. (2021). Comparison of malware classification methods using convolutional neural network based on api call stream. *International Journal of Network Security & Its Applications (IJNSA) Vol 13*.

Shijo, P. and A. Salim (2015). Integrated static and dynamic analysis for malware detection. *Procedia Computer Science 46*, 804–811.

SonicWall (2021). Cyber threat report. Technical report, SonicWall.

Sophos (2021). The state of ransomware 2021. Technical report, Sophos.

Sundarkumar, G. G., V. Ravi, I. Nwogu, and V. Govindaraju (2015). Malware detection via api calls, topic models and machine learning. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1212–1217. IEEE.

Tobiyama, S., Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi (2016). Malware detection with deep neural network using process behavior. In *2016 IEEE 40th annual computer software and applications conference (COMPSAC)*, Volume 2, pp. 577–582. IEEE.

Ucci, D., L. Aniello, and R. Baldoni (2019). Survey of machine learning techniques for malware analysis. *Computers & Security 81*, 123–147.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008.

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xiao, X., S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah (2019). Android malware detection based on system call sequences and lstm. *Multimedia Tools and Applications 78*(4), 3979–3999.

Xu, Z., X. Fang, and G. Yang (2021). Malbert: A novel pre-training method for malware detection. *Computers & Security 111*, 102458.

Yuan, Z., Y. Yan, M. Sonka, and T. Yang (2021). Large-scale robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3040–3049.

Zhu, Q. (2020). On the performance of matthews correlation coefficient (mcc) for imbalanced dataset. *Pattern Recognition Letters 136*, 71–80.

CURRICULUM VITAE

Personal Information

Name Surname : FERHAT DEMİRKIRAN

Education

Undergraduate Education : Electronic and Communication Engineering, Doğuş University Istanbul, Turkey

Foreign Language Skills : English

Work Experience

Companies and Dates :

General Mobile November 2017 – january 2018

Kadir Has University Center For Cybersecurity and Critical Infrastructure Protection February 2019 - Present

Kadir Has University (Research Assistant) January 2020 – Present