Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.036555
Article

Tech Science Press

Augmenting Android Malware Using Conditional Variational Autoencoder for the Malware Family Classification

Younghoon Ban, Jeong Hyun Yi and Haehyun Cho*

Soongsil University, Seoul, 06978, Korea
*Corresponding Author: Haehyun Cho. Email: haehyun@ssu.ac.kr
Received: 04 October 2022; Accepted: 23 November 2022

Abstract: Android malware has evolved in various forms such as adware that continuously exposes advertisements, banking malware designed to access users' online banking accounts, and Short Message Service (SMS) malware that uses a Command & Control (C&C) server to send malicious SMS, intercept SMS, and steal data. By using many malicious strategies, the number of malware is steadily increasing. Increasing Android malware threats numerous users, and thus, it is necessary to detect malware quickly and accurately. Each malware has distinguishable characteristics based on its actions. Therefore, security researchers have tried to categorize malware based on their behaviors by conducting the familial analysis which can help analysists to reduce the time and cost for analyzing malware. However, those studies algorithms typically used imbalanced, well-labeled open-source dataset, and thus, it is very difficult to classify some malware families which only have a few number of malware. To overcome this challenge, previous data augmentation studies augmented data by visualizing malicious codes and used them for malware analysis. However, visualization of malware can result in misclassifications because the behavior information of the malware could be compromised. In this study, we propose an android malware familial analysis system based on a data augmentation method that preserves malware behaviors to create an effective multi-class classifier for malware family analysis. To this end, we analyze malware and use Application Programming Interface (APIs) and permissions that can reflect the behavior of malware as features. By using these features, we augment malware dataset to enable effective malware detection while preserving original malicious behaviors. Our evaluation results demonstrate that, when a model is created by using only the augmented data, a macro-F1 score of 0.65 and accuracy of 0.63%. On the other hand, when the augmented data and original malware are used together, the evaluation results show that a macro-F1 score of 0.91 and an accuracy of 0.99%.

Keywords: Android; data augmentation; artificial intelligence; cybersecurity

1 Introduction

Deep learning is playing a vital role in various cyber security systems such as fraud detection, intrusion detection, spam detection, and malware detection [1,2]. Naturally, a lot of deep learning-based Android malware classification systems have been proposed to detect Android malware which has been evolving in various forms such as Adware that continuously exposes advertisements, Banking malware designed to access users' online banking accounts, and SMS Malware that uses a C&C server to send malicious SMS, intercept SMS, and steal data. Attackers have been designing new attack methods, and thus, there are various types of emerging malware and the number of malware is continuously increasing [3,4]. In addition, there are malware that aim to leak users' personal information. Therefore, there have been a line of research work on approaches for protecting various information such as analysis of abnormal data collected from various devices such as mobile devices and Internet of Things (IoT) devices and data collected from sensors [5–9]. As such, malware is threatening numerous users and thus it is necessary to detect malware quickly and accurately. Malware belonged in a specific malware family has distinguishable characteristics of malicious actions. Therefore, in order to detect malware faster and more accurately, a lot of research on malware analysis and detection based on deep learning algorithms was conducted [10–22]. To effectively analyze/classify the rapidly increasing number of malware, it is more effective to classify malware in detail by family or behavior than to classify benign/malicious [12,23]. In addition, if malware is classified into a family, we can minimize damage affecting users, and the time for manually analyzing malware can be reduced [4,24]. However, to make a classifier based on deep learning algorithms, a large amount of malware data of various types is required and should be labeled as a family. However, collecting and labeling malware data belonging to a specific family in the wild is time-consuming and expensive. In addition, even if a lot of time is invested in collecting malicious code data, there are cases where the number of data belonging to a specific family is not sufficient [17,25,26]. Due to this problem, a large body of research has used well-labeled open-source datasets such as Drebin [27] and Android Malware Dataset (AMD) [28]. However, the well-labeled open-source dataset also has a disadvantage in that the number of data for each family is imbalanced. As a result, the malware family classifier may not detect a specific label and has poor performance for the multi-class classification [29,30]. To overcome this problem, there have been studies that augment data in various fields [31]. However, among these studies, none of them could not completely preserve malicious behaviors when augmenting malware because malware augmentation studies visualize and use features extracted from malware as images.

In this paper, we propose an android malware familial analysis system based on a data augmentation method to create an effective multi-class classifier for malware family analysis. In this work, we particularly focus on the data augmentation method using a generative model to augment data by reflecting various behavioral characteristics of malware. By alleviating the imbalance of the dataset used through the data augmentation, we overcome the problem of the malware family multi-class classifier. Our approach makes a malware family multi-class classifier to learn with rich (augmented) data so that it can effectively classify malware into each family. We extract features that can reflect the behavior of the malicious code by statically analyzing the malicious code. Our proposed method is distinct from the previous work of augmenting data which visualize malware as an image. The extracted features are trained to augment data for each family using a generative model, Conditional Variational Autoencoder (CVAE) [32]. Also, by using the augmented data, we create a Convolutional Neural Network (CNN)-based malware family multi-class classifier. The evaluation result of our proposed method demonstrates through experiments that the malware family multi-class classifier trained with the augmented data can detect the well-labeled open-source dataset data, which is the original data.

In summary, the contributions of this paper are as follows.

- To effectively augment malware while preserving behaviors of them, we study API and Permissions of malicious applications can be used.
- In order to overcome the problem that the previous deep learning-based classifier, which learned the imbalanced dataset, cannot detect a class with little data, we augment the insufficient malware data. Therefore, our deep learning-based android malware familial analysis model can provide enough data to learn.
- We perform evaluations on open-source datasets used in many malwares detection studies. Our evaluation results show that our model has the performance of multi-class family classifiers with high accuracy (99%) and macro F1-score (0.91).

2 Related Work

There are many studies that detect malicious applications based on deep learning/machine learning with augmented malware dataset. Kim et al. [18] extracted five features that can reflect the characteristics of malicious applications. Their evaluation results show the high accuracy because it uses various features. However, the approach may not be suitable for the malware family classification because the binary classification between benign and malicious applications can leverage a clear difference between behaviors of benign and malicious applications, but the familiar analysis requires to find detailed behavioral differences. Qiu et al. [19] showed that the detection efficiency of existing malware detection and family classification studies is low due to the advent of zero-day family malware. To overcome this, Qiu et al. proposed Automatic Capability Annotation for Android Malware (A3CM) that automatically identifies security/privacy-related capabilities rather than classifying malware families. However, since there is no annotated capability malware dataset, the open-source dataset was used to designate security/privacy-related capabilities labels. DL-Droid [10] extracted four features through the hybrid analysis using both dynamic and static analysis. Then, they performed the binary classification against applications using a deep learning classifier. However, the performance deviation of the deep learning classifier is poor because it is highly dependent on features extracted through dynamic analysis. Taheri et al. [20] detected malicious apps using the similarity between malicious code of an open-source dataset. However, there is a limitation in analyzing large-size dataset due to the time complexity of the Nearest Neighbors algorithm for measuring the similarity, and imbalanced open-source data was used as it is. Kim et al. [17] assumed that each malware family performs similar malicious behaviors. Based on this assumption, Kim et al. performed the malware family classification by using permissions that are relatively immune to obfuscation techniques. However, it is difficult for the classifier to accurately classify when the application requests many Permissions.

Those studies performed the binary classification of malware or used imbalanced open-source datasets. In order to perform effective malware detection based on deep learning, other studies have been conducted to alleviate the imbalance of open-source datasets. Arp et al. [27] addressed the issue of not enough family-labeled malware samples. First, it analyzes the malware's raw bytes, extracts features, and creates new data by inserting data into the empty space between each section of the Portable Executable (PE) file. However, since the extracted feature uses only important sequences longer than a certain length for data augmentation, malicious codes of 200 or less that perform malicious actions cannot be used for data augmentation. Raff et al. [33] solved the data imbalance between benign and malicious or malware families through oversampling. To this end, Raff et al. proposed Stochastic Hashed Weighted Lempel-Ziv (SHWeL), a method for vectorizing data. However, since SHWeL is based on Lempel-Ziv Jaccard Distance (LZJD), compressing a shared subsequence of 100 bytes in a byte sequence can damage the binary code structure or malicious behavior information. Chen et al. [13] aimed to mitigate the problem

of data imbalance caused by the scarcity of some malware. To this end, Chen et al. proposed a deep learning-based malware detection framework that visualizes the opcode of an application and uses it as a feature and performs data augmentation through a Generative Adversarial Network (GAN) [34]. However, in the process of normalizing and visualizing opcodes, the behavior information and code structure of the malicious code can be disappeared. Burks et al. [11] also compared and evaluated using GAN and Variational Autoencoder (VAE) [35] to solve the problem of imbalance of malware data. Experimental results show that the efficiencies of the two generative models are different. However, as a result, both generative models increase the accuracy of the classifier. Also, since the feature used in their study is an image, Burks et al. concluded that GAN is more efficient for data augmentation than VAE. Wong et al. [36] proposed Marvolo using semantics-preserving transformations to augment labeled datasets. However, due to Marvolo's efficiency-oriented optimization, the accuracy is improved only for a limited number of binaries.

3 Background

In this section, we discuss technical backgrounds and the limitation of previously proposed deep learning-based approaches for detecting Android malicious applications. Also, we introduce technical challenges for augmenting malware dataset.

Threats to Validity: For this paper, we included the studies that (1) deal with deep learning-based Android malware detection, or malware familial analysis, and malware data augmentation, (2) we have used combinations of strings such as 'deep learning', 'data augmentation', 'Android malware familial analysis', and 'Android malware detection, (3) and discuss our purposed android malware familial analysis system based on a data augmentation method to create an effective multi-class classifier for malware family analysis.

3.1 Deep Learning-Based Malware Detection (Analysis)

To use Machine Learning (ML) and Deep Learning (DL) for malware detection/analysis, most studies extract and use various features such as API, Permission, Operation code (Opcode), and Control Flow Graph (CFG) that can infer the behavior of malware [10,14,15,17–20,37,38]. Previous ML/DL based on malware detection studies show that different classification and evaluation indicators are used using various algorithms, datasets, and features as shown in Table 3. Many of these studies performed malware familial analysis using a well-labeled open-source dataset [17,19,20,37,39]. Also, accuracy, which is the most reasonable performance indicator used by the most studies, is difficult to trust when the data set is imbalanced. This is because the classifier classifies all data into a major class where there is a lot of data to improve accuracy [29,40].

Even well-labeled open-source datasets such as Drebin [27] and AMD [28] used in many studies have imbalanced data distribution between classes. Fig. 1 shows Drebin's imbalanced data distribution. Previous research used an imbalanced open-source dataset as it is, so when splitting train/test data for model generation, the number of classes evenly distributed on both sides is small, and a specific class can be included only in the training set or test set. The classifier will not be able to sufficiently learn the features of minor class malware with little data. As a result, it shows a low detection rate for the minor class or no detection at all. As a result, the malware family classifier cannot detect a specific family and cannot perform effective family analysis, and it can be difficult to respond to new malware [17,19,39]. In addition, the evaluation metric used in previous studies is not suitable for multi-class classification.

For example, suppose you have 990 data in the benign class and 10 data in the malware class. At this time, the malware classifier learns to correctly classify the malware. The classifier 1 in Table 1 shows 99% accuracy by classifying all data into benign classes in order to optimize the accuracy. However, this classifier cannot be used to detect malware because it misclassifies malware. The classifier 2 in Table 2 shows an

accuracy of 98.6% in the same scenario. Although the classifier 2 in Table 2 has low accuracy, it can be seen that it is more accurate for malware detection. This situation is called the accuracy paradox. As such, the accuracy can demonstrate the performance of the model, but only the underlying class distributions are sometimes reflected for unbalanced data. Consequently, when the data is imbalanced, the reliability of the accuracy is low [28,30].

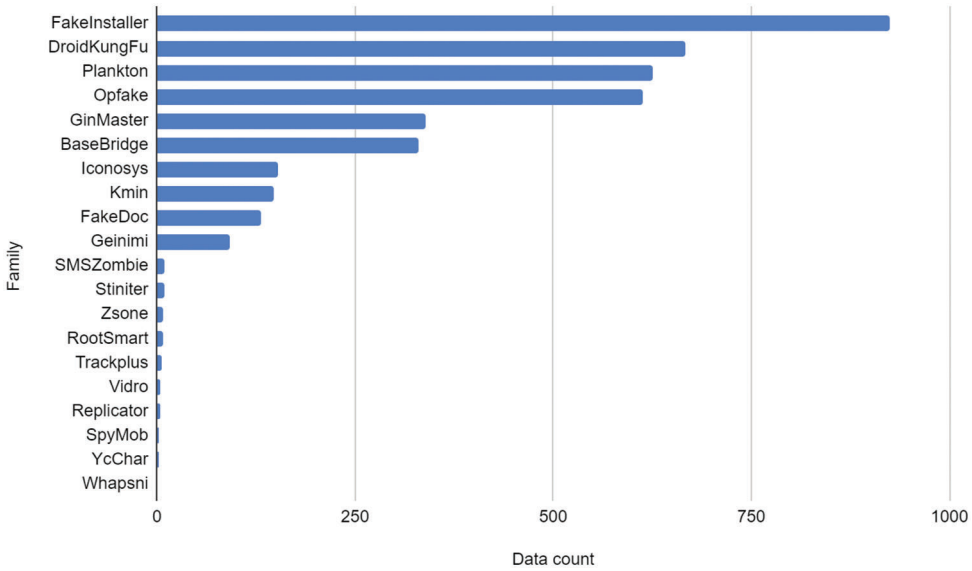

Figure 1: The data distribution of each malware family in Drebin [27]

Table 1: Confusion matrix for hypothesis classifier 1

	Classified benign	Classified malware
Actual benign	0	10
Actual malware	0	990

Table 2: Confusion matrix for hypothesis classifier 2

	Classified benign	Classified malware
Actual benign	6	10
Actual malware	4	980

Table 3: Summary of ML/DL-based malware classification studies

Name	Dataset	Accuracy or F1-score	Classification	Features
DL-Droid [10]	McAfee labs	99%	Binary	API calls, permission, intents, actions/events,
Kim et al. [17]	Drebin	91%	Multiclass	Permission

(Continued)

Table 3 (continued)

Name	Dataset	Accuracy or F1-score	Classification	Features
Kim et al. [18]	Genome project	99%	Binary	API, permission, component, string, opcode
A3CM [19]	Drebin, AMD	99%	Multiclass	API calls, permission, network address
Blanc et al. [37]	Drebin	98%	Multiclass	App review, bytecode instruction, class, method
AFCGDroid [39]	Drebin	95%	Multiclass	Attributed function call graph (Contain API call, permission, etc.)

3.2 Malware Data Augmentation

Alleviating the data imbalance is to increase or decrease the number of training data through sampling. Raff et al. [33] introduced a probabilistic component to vectorization to oversample the byte sequence to mitigate the imbalance.

Another method for alleviating data imbalance is to augment data by using generative models. In general, data generation models, such as GAN [34] and VAE [35], are used to solve the unbalanced data distribution. These approaches can be used to increase the accuracy of the classification model by about 2% to 20% with newly generated data samples. Also, it has been verified through several experiments that data augmentation has a positive effect on malware analysis and detection research [11,13,21,26,36]. On the other hand, there is a line of research work that uses the excellent performance of computer vision research to visualize malware as an image to augment data and detect malware [11,13,21]. For example, Cavli et al. [12] augmented the data with GAN to solve the imbalance in the dataset used in mobile malware detection by visualizing the opcode of the malicious code as an image. However, since the malware is different from the image, the contextual information confusion of the malware binary may occur while visualizing it as an image. The three examples below are typical examples of contextual information confusion that can occur while converting malware into images [22].

- Edge loss: If a binary instruction is placed on an edge of an image, the binary instruction is truncated into two parts and converted to an image. Due to binary instructions truncated in two parts, the model may have difficulty recognizing multiple long instructions. Also, the truncated binary instruction loses contextual information.
- Resampling Noise: Sometimes an image feature is resized to match the size of other images. At this time, as different unrelated binary commands are merged, contextual information may be confused.
- Padding problem: The addition of padding may make it difficult for the model to recognize the start and end of the original binary sequence. In order to make the image size the same, the padding is filled as much as the insufficient size. However, unlike image processing, malware image classification is affected by padding. Therefore, it is difficult for the model to train accurately.

Therefore, the malware family classification model may not properly learn important features of malware because the behavior of the malware can be damaged due to the preprocessing process for augmenting the malware data [11,13,21,26,33,36]. For more accurate and effective malware data augmentation, it is necessary to augment the data based on the original behavior of the malware.

4 Overview

In this section, we describe the overview of our goal and our approach.

4.1 Our Goal

In this paper, our goal is to design an effective multi-class classification system based on a data augmentation scheme for the Android malware familial analysis. To generate an effective deep learning-based model for the familial analysis of real-world Android malware, we first need a rich dataset which contains various malware families affiliated to each malware family. Also, we need abundant malware in each family to effectively train them. In general, when generating deep learning classifiers, the lack of data leads to the poor performance and inadequate outcomes of deep learning models. Unfortunately, in the wild, there is a significant imbalance in the number of malware in each family. Some of them have numerous samples but there are malware families that do not have sufficient samples to train them as we showed in Fig. 1. This characteristic of the real-world malware dataset seriously hinders us from making an effective multi-class classifier for the malware familial analysis.

4.2 Our Approach

To overcome the challenge in the malware data augmentation, we first statically analyze the malware and extract APIs and Permissions, which are important features that can be used to infer behaviors of each malware. Then, we can create rich malware for each family by performing the data augmentation by using the generation model based on the API and Permission extracted earlier. To create an effective multi-class classifier for our target malware family analysis, we use the malware augmented for each family and the actual malware dataset together. Because our proposed technique augments data samples by using the features that can reflect behaviors of malware, it can effectively detect the original data samples. In addition, there is a strong advantage in classifying malware of some families where there are a few of real-world samples. The following section introduces the detail design of our approach.

5 Design

In this section, we introduce detail design of our malware family analysis system based on the data augmentation method. Fig. 2 shows the architecture of our system. Since our proposed method is to design an android family multi-class classifier based on data augmentation, we first design a generative model for data augmentation. We extract APIs and Permissions that can infer behaviors of malicious applications from each application. The extracted API and Permission are marked 0 and 1 after determining whether each application is used or not. Afterward, it is used as input with family information into the generation model CVAE [32] of Fig. 2, and malware samples for each family is augmented. Finally, using the augmented data and the original data together as a training dataset, a classifier for the android family multi-class classification is generated.

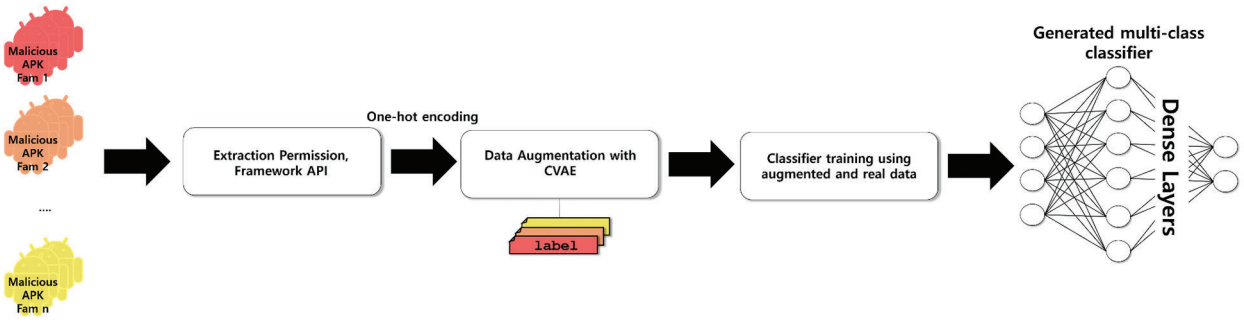

Figure 2: The architecture of the proposed scheme

5.1 Feature Extraction and Encoding

App developers can implement and use various functions such as mobile device network connection, camera, Global Positioning System (GPS), call, SMS transmission, and storage access through APIs to utilize various hardware functions of the mobile device. Also, Android applications depend on framework APIs for inter-application communication and hardware interactions [41]. In addition, the Android system uses the permission-based security mechanism to restrict applications' accesses to system resources. As such, permissions are at the heart of Android's security model [17].

For example, to use the *getExternalFilesDir()* API to access app files from any external storage, we must first request the READ_EXTERNAL_STORAGE permission in AndroidManifest.xml file. Through this, we know that in order to use a specific API, a specific permission must be declared first. After all, by analyzing the permissions and APIs used in an application, we can know what functionalities are provided to the user and what the application does. For more details, in the case of APIs, when an application uses *android.telephony* and *android.telecom* APIs, it can be inferred that the application monitors the network status of a mobile device and manages phone calls. As a result, the APIs used by an application is a useful feature that provides information about the functionalities of the application [15]. Also, combinations of multiple permissions can reflect some harmful behaviors and requesting specific permissions is essential for malware to achieve its goals. Permission can also play an important role in malicious code analysis [17,42]. For an effective malware data augmentation, we focus on which APIs and permissions are used rather than the order or frequency of use of APIs and permissions. To this end, we use two arrays, in which each entry stands for a specific API and permission, that consists of only 1 and 0 to express whether an API or permission is used in an application or not.

5.2 Augmentation Model CVAE

The goal of a generative model is to learn the training data and generate similar data that follow the distribution of the data. VAE [35] has the same structure as Autoencoder with encoder and decoder. The main goal of Autoencoder is to compress and encode data. On the other hand, VAE aims to create a latent vector using an encoder to find the probability distribution of data from input data and to generate new data samples with a decoder.

VAE is a statistical probability distribution model for generating new data samples by learning the distribution of the dataset. When a latent vector z is given, it learns in a direction that maximizes the likelihood that a data sample x will appear. At this time, if we slightly change the latent vector z, we can get the new data sample x' we want. The VAE encoder calculates the mean and variance, the latent variables of the n-dimensional normal distribution, from the high-dimensional data x. The latent vector z is generated using the mean, variance, and ε generated through the encoder. ε is a value obtained by extracting random values from a normal distribution to generate various values. In this process, the mean and variance calculated by the encoder can be updated because of noise samplings obtained from the normal distribution. The decoder generates the reconstructed input data x' from the latent vector z value. Decoder $p(x|z)$ maps the latent vectors back to the original input data. In conclusion, the VAE is trained to minimize the reconstructed error between the input data x and the reconstructed input data x'. In this case, the loss function of VAE is defined as follows Eq. (1).

$$L_{VAE}(\theta,\ \phi) = -E_{z\sim q\phi(x)}[(z)\ ] + D_{KL}[q\phi(z|x)\ \|\ p\theta(z)] \tag{1}$$

$Encoder(q(z|x))$ is a posterior probability function. This approximates the posterior probability distribution $p(z|x)$. Also, it is assumed that the prior probability distribution $p(z)$ is a Gaussian distribution. In the loss function, D_{KL} is the KL Divergence to measure and minimize the dissimilarity between the two distributions, which measures the degree of similarity between $p(z)$ and $q(z|x)$.

CVAE [32] learns with the same structure and goals as VAE. However, unlike VAE, CVAE uses condition information with input data. Our proposed method uses behavioral characteristics of a malware family as condition information. Specifically, in this paper, we use the extracted API and permission information together with the malware family information in CVAE, it is possible to generate high-quality malware samples than using the existing VAE. Therefore, to perform our proposed data augmentation in a family unit, CVAE is used to augment the malware data.

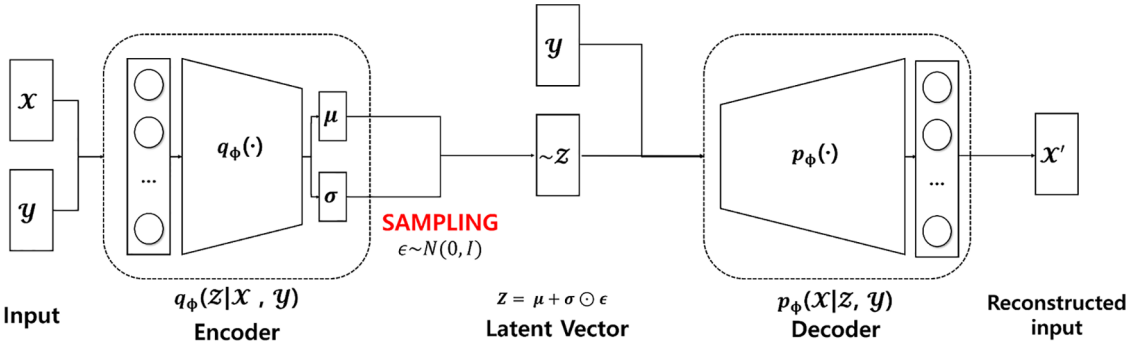

Figure 3: The architecture of the CVAE

5.3 Malware Family Classification Model

We use the Convolution Neural Network (CNN) to implement a malware family multi-class classifier. The input feature is used with the one-hot encoding of the API and Permission used by the application. An embedding layer maps input features to vectors in the embedding space and generates embedding vectors. The embedding vector goes through a one-dimensional convolution layer (Conv1D) using Rectified Linear Unit (ReLu) as an activation function. Conv1D recently showed excellent performance in Natural Language Processing (NLP) and high performance in existing malware detection studies [43]. Then, the data family label is classified as the last fully connected layer. Since our model performs multi-class classification, we use Sigmoid as the activation function of the last dense layer. Finally, our model minimizes the binary cross-entropy loss and learns from the given data. Fig. 4 shows the structure of the malware family multi-class classifier of the CNN-based model we used.

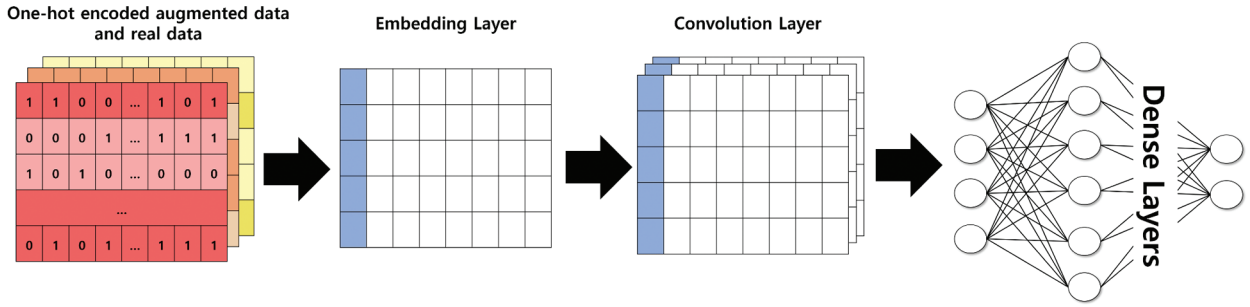

Figure 4: The architecture of the CNN-based malware family multi-class classifier

6 Evaluation

In this section, we evaluate the effectiveness of malware data augmentation and the deep learning-based multi-class classifier generated with the augmented malicious applications. The generative model and multi-class classifier used in our evaluation used the CVAE and CNN mentioned in Sections 5.2 and 5.3.

6.1 Experiment Environment

Experimental environment was performed on 2 Intel(R) Xeon(R) Gold 6230 20-core 2.10 GHz CPUs, 256 GB RAM, 4 NVIDIA GeForce RTX 2080 Ti GPUs, and Ubuntu 18.04.3 LTS. The language used was Python 3.7.7, and AndroGuard [44] 3.4.0a1 was used to extract APIs and Permissions by statically analyzing the application. We used TensorFlow-GPU 1.14.0, Keras 2.2.4, and CUDA 10.1 for training the deep learning model we used.

6.1.1 Model Setting

The CVAE [32] used for data augmentation used 4 dense layers. Each dimension of hidden states is 2048, 70, 1024, and 2048 respectively. The activation function uses ReLu and sigmoid. The data were augmented by 2000 for each label using the Adam optimizer with learning rate = 0.0002 and beta_1 = 0.5. The CNN-based malware family multi-class classifier uses two Conv1Ds and uses ReLu as an activation function. Each of the two dense layers. Each dimension of hidden states is 32, 4 respectively. The Adam optimizer with a learning rate of 0.001 was used. The batch size is 256. CNN learns by using augmented and raw data together. The generative model we used and the hyperparameters of the CNN-based multi-class classifier are most suitable for our dataset, and we selected the hyperparameters with the best performance through several experiments.

6.1.2 Dataset

Among the datasets used by various malware detection and classification research, we perform experiments with Drebin [27], a well-labeled open-source dataset. Drebin is a very useful dataset for algorithms using supervised learning that require labels because 5,560 pieces of the sample are labeled with 179 malware families. Fig. 1 shows some sample distributions from Drebin. FakeInstaller, the family with the most samples, has 925 malwares, accounting for about 17% of the total data. On the other hand, the families with the least sample, Whapsni, Qicsom, and Updtbot, have only one malware, accounting for 0.01% of the total data. This shows that Drebin is very imbalanced. If we extract the API and Permission that we use as a feature, a total of 5,480 samples are available. With 80 bad applications, the usable samples are reduced. Therefore, we performed data augmentation for a total of 179 malware families using a total of 5,480 samples in the experiment. Also, we use the AMD dataset [28] that consists of 24,553 malware samples and each of them belongs one of 71 families.

6.2 Evaluation Metrics

Our malware family detection model classifies Drebin's [27] number of family labels into 179. The malware family classification model trained on our augmented data is evaluated using accuracy, recall, precision, and F1-score. Accuracy is the most used and is generally an evaluation metric showing the performance of a model [45]. However, since the data we used is imbalanced and the importance of all labels used in classifying malware families is the same, we used a macro average to use F1-score, Precision, and Recall together [46]. For original data evaluation and augmented data evaluation, 80% is used as training data and the remaining 20% is used as test data. Also, in the evaluation of augmented and original data together, the two data are merged and randomly shuffled. After that, 80% is used as training data and the remaining 20% is used as test data.

6.3 Drebin's Imbalance Evaluation

We evaluate the performance of a CNN-based multi-class classifier using the metric described in Section 6.2 to account for Drebin's [27] imbalance. In addition, the API and permission, which are the features we selected, were divided and compared and evaluated when using only the API and when using the API and permission together. The graph on the left of Fig. 5 shows the results of learning Drebin data in two cases. Both cases that we compared show the accuracy of 0.94, indicating that malware family

classification is effective. However, as we can see in Fig. 1, because Drebin's data is imbalanced, there is no test data for a specific malware family, so the evaluation indicators macro-Recall, macro-Precision, and macro-F1-score all show the low performance. Consequently, because the data imbalance has not been considered, albeit the classification accuracy is high, the family detection is not excellent. Nevertheless, the evaluation results show that the API and permission are effective features for malware classification.

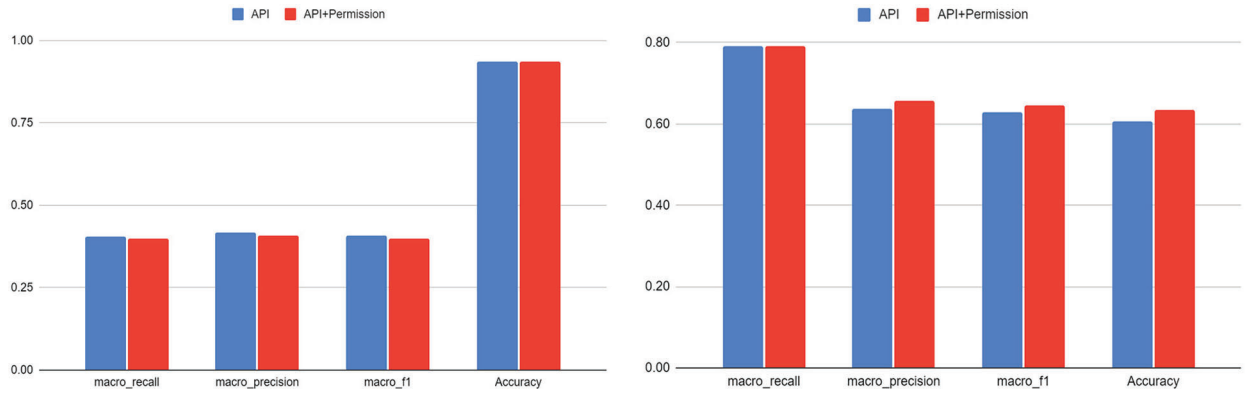

Figure 5: Results of evaluating Drebin's [27] imbalance using API, API, and permission and results of augmented data evaluation

6.4 Augmented Data Evaluation

We evaluate the performance of augmented data with the data augmentation technique that is the basis of our proposed method. The graph on the right of Fig. 5 is the results of evaluating the performance of the multi-class classifier trained with data augmented by using 2,000 data for each family targeting all 179 families of Drebin [27], and the original data.

Both features we used show the accuracy of 0.61 to 0.63 respectively, which is about 0.3 less than the accuracy of Section 6.3. We performed data augmentation using the generative model. It can be seen through experiments that our augmented model did not have enough original family data to learn, and thus the quality of specific augmented family data was low. However, the experimental results show macro-Recall, macro-Precision, and macro-F1-Score higher than the results of Section 6.3. Therefore, we find that the original data can be better detected because the API and permission we selected are features that reflect the actual behaviors of an application. As a result, we verified the validity of our augmented data through experiments. Therefore, the evaluation results demonstrate that we can create a multi-class classifier that does android malware familial analysis based on data augmentation.

6.5 Evaluating a Classifier that Trained Both Original and Augmented Data

Our goal is to evaluate whether we can create a multi-class classifier that performs android malware familial analysis based on data augmentation. For the evaluation, we combine the original and augmented data. The combined data were split and used as previously described in Section 6.1.1. The graph on the left of Fig. 6 shows the evaluation result of the multi-class classifier learned by combining the augmented data and the original data. As a result of the evaluation, the macro-F1-score of 0.91 is the highest when data augmented by using API and Permission together is included. However, when data augmented using APIs are included, a macro-F1-score of 0.9, which is an insignificant difference, is shown. However, it can be seen that both cases are still effective for malware family classification.

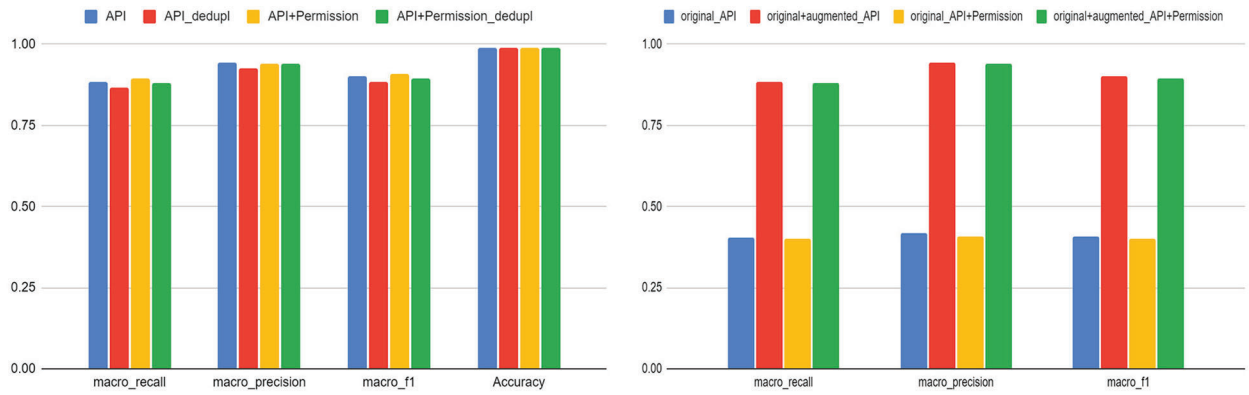

Figure 6: Results of augmented data evaluation, original and augmented data comparison of evaluation result

When duplication is allowed for the augmented data, a macro-F1-score of 0.9 to 0.91 is shown. However, when the deduplicates for the augmented data, a macro-F1-score of 0.88 to 0.89 is shown. This shows an insignificant difference of 0.02, and it can be seen that the redundancy of data does not have a significant effect on malware family classification.

6.6 Evaluation of Comparison of Original and Augmented Data

We evaluate whether data augmentation can generate a multi-class classifier for the Android malware familial analysis. The graph on the right of Fig. 6 compares the performance evaluation result by creating a multi-class classifier using only original data and the performance evaluation result by creating a multi-class classifier using the both original data and augmented data. The classifier created by using the augmented data together with all three evaluation indices used in the performance evaluation shows about 2 times higher performance than the classifier created with the original data. Those evaluation results clearly show that the augmented data is effectively generated by using the APIs and permissions and the features can be effectively used to infer behaviors of malicious applications.

6.7 Evaluation of Comparison of Original and Augmented Data

We evaluate our proposed method using another open-source dataset, AMD [28]. In the AMD dataset, a total of 24,553 pieces of the samples are classified into 71 families. We only use 24,474 samples from which features are extracted. Fig. 7 compares the performance evaluation results of the multi-class classifier created using only original data before data augmentation with the performance evaluation results of the classifier using both original data and augmented data. Since AMD has a more significant number of malware samples than Drebin [27], it shows excellent classification performance of macro-F1-score 0.87 in both cases of using only API as a feature and using API and Permission together as a feature. Through this, we can confirm once again that the APIs and Permissions we used perform effective classification. In addition, the model generated by our proposed method shows a macro-F1-score of 0.98 when only API is used as a feature, and a macro-F1-score of 0.99 when API and Permission are used together as a feature. As a result, it can be seen that our proposed method shows higher performance than the macro-F1-score of the original data. The evaluation results demonstrate that our proposed method can be effectively applied to other datasets besides Drebin.

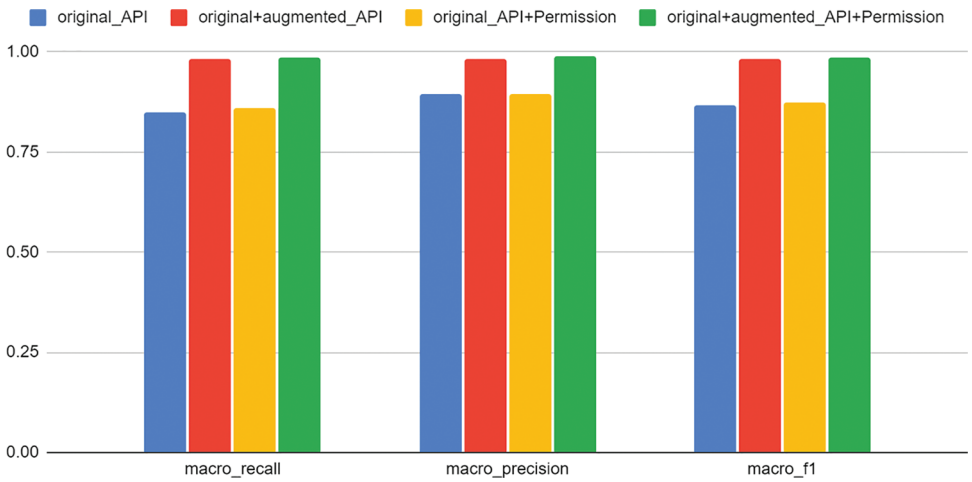

Figure 7: Evaluation of comparison of original and augmented data for AMD [28] dataset

7 Discussion

In this section, we describe the limitations of our approach.

7.1 Why is GAN not Used as a Generative Model?

GAN [34] is a generative model composed of two competing networks, Generator (G) and Discriminator (D). G is trained to produce a more realistic output, and D is simultaneously trained to distinguish the distribution of real data from the synthetic data generated by G. GAN has been successfully applied to methods for generating image samples. However, there is a problem in generating discrete sequence data. After all, the expression method for the features we used is difficult for GANs to learn. Because GAN is designed to generate continuous data, it is difficult to directly generate discrete token sequences (e.g., text) [47]. Therefore, most text GANs rely on gradient-free Reinforcement Learning (RL). However, RL suffers from an unstable training process because it gives up gradient information. Although some studies have investigated the feasibility of gradient-based methods, the optimization is still inefficient.

7.2 Why Not Use Pre-Trained Embeddings?

One could use a pre-trained embedding to initialize the embedding layer in our neural network. Word2Vec [48] is among the most widely used embedding generation techniques. Once an embedding is learned, it initializes the embedding layer with pre-trained ones, making it easy to test embeddings with various neural network architectures, including multi-class classifiers. However, in Word2Vec, the location of the surrounding words and the central word for learning is important when learning. Also, because embeddings are formed by learning only words that appear during learning, an out-of-vocabulary (OOV) problem occurs for unlearned words. The OOV problem is big in generating our proposed multi-class classifier based on data augmentation. Because to solve the OOV problem, a specific token (_UNK_) is added. For example, if an application filled with only _UNK_ exists and the application is augmented, data that helps classify the malware family cannot be generated.

7.3 Why is the Low Quality of Augmented Data?

In this paper, we use CVAE [32] to augment malware samples of each family. In the raw data we used for our evaluation, CVAE could not learn enough about families with a very small number of malware. As shown in Fig. 1, the original Drebin [27] data is also imbalanced. Therefore, the augmented malware samples of a

specific family are not proper to learn and to be used for generating a deep learning-based classifier, but our proposed method augments the data without compromising the behavior of the malware.

The time complexity of the deep learning models used in our proposed technique is affected by the number of layers used in the model. Also, the multi-class classifier we used has a time complexity of $O(n)$ because it is constructed based on CNN [49]. Here, 'n' is the number of features. As a result, the more features used, the more linearly the time complexity of the classifier increases. In this regard, it can be difficult to quickly respond to the increasing cyber threats [50]. We leave optimizations of models generated by the proposed approach as our future work.

8 Conclusion

In this work, we studied a method to design a multi-class classifier based on a data augmentation scheme for a malware familial analysis system. To this end, we extracted APIs and permission that can be used to infer behaviors of malicious applications, performed data augmentation with a generative model, and created a multi-class classifier for malware familial analysis. Our evaluation results show that the multi-class classifier effectively and accurately classified by using the rich (augmented) malware dataset and original data with the accuracy of up to 0.99% and a macro-F1-score of 0.91.

Funding Statement: This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT), and Future Planning under Grant NRF-2020R1A2C2014336 and Grant NRF-2021R1A4A1029650.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, S. Chen *et al.,* "Performance comparison and current challenges of using machine learning techniques in cybersecurity," *Energies*, vol. 13, no. 10, pp. 2509, 2020.

[2] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.

[3] I. Almomani, A. Alkhayer and W. El-Shafai, "An automated vision-based deep learning model for efficient detection of android malware attacks," *IEEE Access*, vol. 10, pp. 2700–2720, 2022.

[4] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *2020 IEEE Int. Conf. on Dependable, Autonomic and Secure Computing, Int. Conf. on Pervasive Intelligence and Computing, Int. Conf. on Cloud and Big Data Computing, Int. Conf. on Cyber Science and Technology Congress*, Calgary, AB, Canada, pp. 515–522, 2020.

[5] A. Nasir, K. Shaukat, K. I. Khan, I. A. Hameed, T. M. Alam *et al.,* "What is core and what future holds for blockchain technologies and cryptocurrencies: A bibliometric analysis," *IEEE Access*, vol. 9, pp. 989–1004, 2020.

[6] I. Javed, X. Tang, K. Shaukat, M. U. Sarwar, T. M. Alam *et al.,* "V2x-based mobile localization in 3D wireless sensor network," *Security and Communication Networks*, vol. 2021, 2021.

[7] K. Shaukat, T. M. Alam, I. A. Hameed, W. A. Khan, N. Abbas *et al.,* "A review on security challenges in internet of things (iot)," in *2021 26th Int. Conf. on Automation and Computing (ICAC)*, Portsmouth, UK, pp. 1–6, 2021.

[8] K. Shaukat, A. Rubab, I. Shehzadi and R. Iqbal, "A socio-technological analysis of cyber crime and cyber security in Pakistan," *Transylvanian Review*, vol. 1, no. 3, pp. 4187–4190, 2017.

[9] M. U. Hassan, M. Shahzaib, K. Shaukat, S. N. Hussain, M. Mubashir *et al.,* "Dear-2: An energy-aware routing protocol with guaranteed delivery in wireless ad-hoc networks," in *Recent Trends and Advances in Wireless and IoT-Enabled Networks*, pp. 215–224, 2019.

[10] M. K. Alzaylaee, S. Y. Yerima and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, pp. 101663, 2020.

[11] R. Burks, K. A. Islam, Y. Lu and J. Li, "Data augmentation with generative models for improved malware detection: A comparative study," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conf.*, New York, NY, USA, pp. 0660–0665, 2019.

[12] O. F. T. Cavli and S. Sen, "Familial classification of android malware using hybrid analysis," in *Int. Conf. on Information Security and Cryptology*, Ankara, Turkey, pp. 62–67, 2020.

[13] Y. -M. Chen, C. -H. Yang and G. -C. Chen, "Using generative adversarial networks for data augmentation in android malware detection," in *IEEE Conf. on Dependable and Secure Computing*, Aizuwakamatsu, Fukushima, Japan, pp. 1–8, 2021.

[14] C. Ding, N. Luktarhan, B. Lu and W. Zhang, "A hybrid analysis-based approach to android malware family classification," *Entropy*, vol. 23, no. 8, pp. 1009, 2021.

[15] J. Kim, Y. Ban, E. Ko, H. Cho and J. H. Yi, "MAPAS: A practical deep learning-based android malware detection system," *International Journal of Information Security*, pp. 1–14, 2022.

[16] K. Shaukat, S. Luo and V. Varadharajan, "A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks," *Engineering Applications of Artificial Intelligence*, vol. 116, pp. 105461, 2022.

[17] M. Kim, D. Kim, C. Hwang, S. Cho, S. Han *et al.,* "Machine-learning-based android malware family classification using built-in and custom permissions," *Applied Sciences*, vol. 11, no. 21, pp. 10244, 2021.

[18] T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2018.

[19] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal *et al.,* "A3CM: Automatic capability annotation for android malware," *IEEE Access*, vol. 7, pp. 147156–147168, 2019.

[20] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian *et al.,* "Similarity-based android malware detection using hamming distance of static binary features," *Future Generation Computer Systems*, vol. 105, pp. 230–247, 2020.

[21] A. Tekerek and M. M. Yapici, "A novel malware classification and augmentation model based on convolutional neural network," *Computers & Security*, vol. 112, pp. 102515, 2022.

[22] J. Xu, W. Fu, H. Bu, Z. Wang and L. Ying, "SeqNet: An efficient neural network for automatic malware detection," arXiv preprint arXiv:2205.03850, 2022.

[23] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *2012 IEEE Symp. on Security and Privacy*, San Francisco, CA, USA, pp. 95–109, 2012.

[24] M. Khushi, K. Shaukat, T. M. Alam, I. A. Hameed, S. Uddin *et al.,* "A comparative performance analysis of data resampling methods on imbalance medical data," *IEEE Access*, vol. 9, pp. 109960–109975, 2021.

[25] H. S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models," arXiv preprint arXiv:1804.04637, 2018.

[26] D. Yuxin, W. Guangbin, M. Yubin and D. Haoxuan, "Data augmentation in training deep learning models for malware family classification," in *2021 Int. Conf. on Machine Learning and Cybernetics*, Adelaide, Australia, pp. 1–6, 2021.

[27] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck *et al.,* "Drebin: Effective and explainable detection of android malware in your pocket," *Ndss*, vol. 14, pp. 23–26, 2014.

[28] F. Wei, Y. Li, S. Roy, X. Ou and W. Zhou, "Deep ground truth analysis of current android malware," in *Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, Bonn, Germany, pp. 252–276, 2017.

[29] J. Akosa, "Predictive accuracy: A misleading performance measure for highly imbalanced data," in *Proc. of the SAS Global Forum*, Orlando, FL, USA, vol. 12, pp. 1–4, 2017.

[30] H. -J. Ye, H. -Y. Chen, D. -C. Zhan and W. -L. Chao, "Identifying and compensating for feature deviation in imbalanced deep learning," arXiv preprint arXiv:2001.01385, 2020.

[31] S. Türker and A. B. Can, "Andmfc: Android malware family classification framework," in *2019 IEEE 30th Int. Symp. on Personal, Indoor and Mobile Radio Communications*, Istanbul, Turkey, pp. 1–6, 2019.

[32] K. Sohn, H. Lee and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[33] E. Raff and C. Nicholas, "Malware classification and class imbalance via stochastic hashed lzjd," in *Proc. of the 10th ACM Workshop on Artificial Intelligence and Security*, Dallas, Texas, USA, pp. 111–120, 2017.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.,* "Generative adversarial networks," *Communications of the ACM*, New York, NY, USA, vol. 63, no. 11, pp. 139–144, 2020.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," arXiv preprint arXiv:1312.6114, 2013.

[36] M. D. Wong, E. Raff, J. Holt and R. Netravali, "Marvolo: Programmatic data augmentation for practical ml-driven malware detection," arXiv preprint arXiv:2206.03265, 2022.

[37] W. Blanc, L. G. Hashem, K. O. Elish and M. H. Almohri, "Identifying android malware families using android-oriented metrics," in *IEEE Int. Conf. on Big Data*, Los Angeles, CA, USA, pp. 4708–4713, 2019.

[38] N. Daoudi, K. Allix, T. F. Bissyandé and J. Klein, "A deep dive inside drebin: An explorative analysis beyond android malware detection scores," *ACM Transactions on Privacy and Security*, vol. 25, no. 2, pp. 1–28, 2022.

[39] T. Lu, X. Liu, J. Chen, N. Hu and B. Liu, "AFCGdroid: Deep learning based android malware detection using attributed function call graphs," in *Journal of Physics: Conf. Series*, Inner Mongolia, China, vol. 1693, pp. 012080, 2020.

[40] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[41] L. Li, T. F. Bissyandé, H. Wang and J. Klein, "Cid: Automating the detection of api-related compatibility issues in android apps," in *Proc. of the 27th ACM SIGSOFT Int. Symp. on Software Testing and Analysis*, Amsterdam, Netherlands, pp. 153–163, 2018.

[42] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto *et al.,* "Droidsieve: Fast and accurate classification of obfuscated android malware," in *Proc. of the Seventh ACM on Conf. on Data and Application Security and Privacy*, Scottsdale, Arizona, USA, pp. 309–320, 2017.

[43] Y. Kim, "Convolutional neural networks for sentence classification. CoRR abs/1408.5882 (2014)," arXiv preprint arXiv:1408.5882, 2014.

[44] Androguard, 2022. https://github.com/androguard/androguard.

[45] M. Grandini, E. Bagli and G. Visani, "Metrics for multi-class classification: An overview," arXiv preprint arXiv:2008.05756, 2020.

[46] C. Parker, "An analysis of performance measures for binary classifiers," in *2011 IEEE 11th Int. Conf. on Data Mining*, Vancouver, BC, Canada, pp. 517–526, 2011.

[47] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?," arXiv preprint arXiv:1511.05101, 2015.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, Lake Tahoe, Nevada, USA, pp. 3111–3119, 2013.

[49] S. Lai, L. Xu, K. Liu and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI Conf. on Artificial Intelligence*, Austin, Texas USA, 2015.

[50] K. Shaukat, S. Luo, S. Chen and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," in *2020 Int. Conf. on Cyber Warfare and Security (ICCWS)*, Islamabad, Pakistan, pp. 1–6, 2020.