Journal of Information Security and Applications 84 (2024) 103800

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

GMADV: An android malware variant generation and classification adversarial training framework☆

Shuangcheng Li [a], Zhangguo Tang [a,b,*], Huanzhou Li [a], Jian Zhang [a], Han Wang [a], Junfeng Wang [b]

[a] School of Physics and Electronic Engineering, Sichuan Normal University, Chengdu, 610101, China
[b] School of Cyber Science and Engineering, Sichuan University, Chengdu 610207, China

ARTICLE INFO

Keywords:
Android malware
RGB Markov image
GMM-GAN
Variant amplification

ABSTRACT

Android malware uses anti-reverse analysis and APK shelling technology, which leads to the failure of the classification method based on decompiled features and the reduction of the classification accuracy based on single file features. Moreover, the lack of samples in some families of Android malware makes the classification model based on sample learning ineffective. To solve the above problems, this paper proposes a two-layer general framework for Android malware classification and adversarial training named GMADV, which enhances classifier performance through adversarial training. In the sample classification layer, based on the transformation method of the Markov model, it is proposed for the first time to convert the three files in the APK into RGB Markov images, and use VGG13 to automatically extract features and classification; In the variant amplification layer, the idea of "regression for generation" is firstly proposed, and GMM-GAN based on Gaussian process is designed to amplify the diversity of samples within the family. The experimental results show that RGB Markov images have better classification performance than grayscale images. On the three datasets, the classification effect after amplification has been improved to varying degrees, and all F1_Score reaches 95 %. Compared with other methods, GMADV has stronger family sample amplification ability and greater adversarial intensity.

1. Introduction

With the vigorous development of information technology and network services, people's demand for application services has gradually shifted from computers to mobile devices. In the smartphone market, the usage of the Android system reaches 43.43 % [1]. While the Android system is recognized by the majority of users, it has become a criminal carrier for criminals. A huge amount of malware is mixed in the Android mobile market, and it hides a large number of malicious behaviors such as system monitoring, privacy theft, and remote control, infringing on the legitimate rights and interests of users. It is a major challenge to network security governance, and Android malware detection has become an indispensable link. People pay more and more attention to Android software detection, and a variety of detection technologies have been proposed for privacy protection.

At present, Android malware detection mainly presents two problems. On the one hand, in order to improve its concealment, the existing Android malware uses anti-reverse analysis to invalidate some detection technologies based on decompilation and feature extraction, or use APK shelling technology to encapsulate and confuse the code, adjust the logical order of the code and add redundant code, resulting in the reduction of the effect of the detection method based on automatic feature extraction from a single file. However, the method of feature extraction based on decompilation is widely used. For example, Arp et al. [2] first obtains the code obtained by decompilation of the dex file in the APK and the AndroidManifest.xml file, then extract as many features as possible, such as application permissions, API calls and network addresses, and embed them into the joint matrix and use SAM for classification, achieving an accuracy of 94 %. Li et al. [3] extracts Android app features from manifest files and source code, proposes a novel Factorization Machine model for Android malware detection. Efficiently handles long and sparse feature representations. The final

☆ This work was supported in part by the National Key Research and Development Program under Grant 2019QY1400; in part by the National Natural Science Foundation of China under Grant U2133208 and Grant 62101368; and in part by the Sichuan Youth Science and Technology Innovation Team under Grant 2022JDTD0014;

* Corresponding author.
E-mail address: tangzhangguo@sicnu.edu.cn (Z. Tang).

https://doi.org/10.1016/j.jisa.2024.103800

Available online 7 June 2024

accuracy reaches more than 99 %. Xu et al. [4] obtains bytecode through decompilation, and obtains different semantics through different combinations of its context, then by performing the combination of a Bytecode2Vec technique and a Long Short Term Memory neural network, achieving 97.74 % accuracy. The above methods have shown very good results in Android malware detection, effectively preventing the intrusion of Android malware, but the flaw is that they lose their original advantages in the face of Android malware with anti-reverse analysis capabilities.

On the other hand, Android malware has the characteristics of a small number of samples in some families, and the scarcity of samples leads to poor classification ability of deep models based on sample learning. Due to the strong family homology of Android malware, samples of the same family have a certain correlation. According to this feature, methods to improve detection by increasing the number and variants of Android malware have been proposed. Vasan et al. [5] by using traditional image processing techniques to expand additional samples without feature modification, such as rotation, flipping, scaling, and shifting, multiple additional image samples are derived from one sample, thereby generating new data. Singh et al. [6] proposed a malware image generation model MIGAN that explicitly embeds the domain knowledge of the dataset during training, which improves the performance of the classifier by performing data augmentation. Chen et al. [7] transformed the features of malware into image expressions, and used GAN to generate data from a small number of malicious families to balance and expand the original dataset. Compared with before and after data augmentation, the difference in f1-score accuracy reached 5 % ~ 20 %. Although the above methods use different GANs to achieve the amplification of Android malware variants, their essence is only the resurrection of the original samples, and the generated samples have almost the same feature representation and data statistical laws as the original samples, which limits the powerful effect of adversarial training.

In order to solve the related problems caused by the anti-reverse analysis of Android malware and the lack of samples in the family, this paper proposes a general framework GMADV for Android malware classification, including variant amplification layer and sample classification layer. The variant amplification layer amplifies family variant diversity through recombination of different sample feature ratios and Gaussian process regression; The sample classification layer converts three files with different formats into RGB Markov images, and uses VGG13 [8] for feature self-extraction and then classification. The GMADV framework strengthens the classifier based on adversarial training to improve the classification of Android malware. The main contributions of this paper are summarized as follows.

- It is proposed to convert multiple complex data of the original Android malware into visual RGB Markov images, which achieves a unified characterisation of different types of Android malware files, and solves the problem of difficult coupling due to the different sizes of different files and the inconsistent size of the converted images. The Markov model is based on the modeling concept of time series and probability, which can convert and utilize all the required files.
- The sample diversity enhancement method based on GMM-GAN, by introducing the concept of content image and style image, extracting the content features of Android malware according to the content loss, extracting the style features of other families according to the style loss, and mixing and recombining the two features to achieve Android malware. Lateral generation of malware variants, data camouflage and data obfuscation while expanding the data domain, to achieve high-intensity adversarial training.
- The idea of "regression for generation" is proposed. Based on Gaussian process regression, the link of sample generation is constructed through time series modeling, and the intermediate state is presented. In the face of small samples, using fewer style images is to achieve iterative generation of style images, longitudinally generate and expand the quantity of samples, and then use GMM-GAN to expand the inter-class variant diversity.

The remainder of this paper is structured as follows: In Section II, we present related work on Android malware classification. Section III details the workflow of the proposed adversarial training framework. In Section IV, we present the experimental results and analysis. The fifth section summarizes the work of this paper.

2. Related work

2.1. Method of feature extraction based on raw data

The method based on feature extraction mainly refers to decompiling or dynamically running the software to be detected to extract important features, and matching with the relevant features of existing malware to achieve the purpose of detection. Wang et al. [9] proposed a lightweight framework for Android malware identification by performing multi-level network traffic analysis to collect necessary network traffic features. By combining network traffic analysis and machine learning algorithms, their framework was evaluated on the Derbin dataset, which consists of 5560 malware samples. When the two detection mechanisms were combined, the detection rate reached 97.89 %. Zhou et al. [10] studied the characteristics of malicious traffic on the host, constructed traffic fingerprints, combined with machine learning algorithms, added an additional layer called obfuscation classifier to distinguish similar obfuscated traffic, and tested it using the cicandmal2017 dataset in the real world. The classification accuracy of malicious code reaches 95.2 %. Hu et al. [11] proposed a system named MIGDroid, which first constructs a method call graph at the smali code level, then divides the method call graph into weakly connected subgraphs, and calculates a threat score for each subgraph according to the called sensitive API. Scores are positively correlated with the likelihood of subgraphs being malicious code. The method of feature extraction is highly dependent on human prior knowledge. It has high requirements for the number of feature extraction and the important features of the screening, and the manual analysis is cumbersome and difficult to achieve.

2.2. Data characterization and end-to-end classification methods

Convolutional neural networks are widely used in the field of image recognition and can automatically extract important features that are unknown to people. Therefore, an end-to-end detection method is proposed to solve the problem that Android malware feature extraction is difficult and anti-reverse analysis cannot be decompiled. Hasegawa et al. [12] intercepted the four specification bytes of 512, 1024, 2048 or 4096 in the beginning or end of the APK file, and then used 1D convolutional CNN for classification, and confirmed that only the last 512–1k bytes of the APK file were used to obtain 95.40–97.04 % accuracy. Huang et al. [13] proposed a coloR-inspired Convolutional neural networks (CNN)-based Android malware Detection (R2-D2) system. The dex file was obtained by decompressing the APK and displayed in the form of bytecode. According to certain rules, the Hexadecimal is mapped from bytecode to RGB color code to obtain color image, and CNN is used for classification, and 92 %~99 % accuracy is achieved on different families. Bakour et al. [14] build the dex file, AndroidManifest.xml and resources.arsc files into grayscale image datasets with widths of 256, 64, and 64 respectively, and the height changes with the file size. In addition, the dex file, AndroidManifest.xml and resources. The combination of the three of arsc builds a grayscale image dataset with a width of 256. The height changes with the file size. Detection is performed by extracting local features and global features, this is the first time that a convolutional neural network model is trained based on this type of features and used in the android malware detection domain. Based on the grayscale image set combined with the three files, the global feature analysis is used to achieve 98.05 % accuracy. Ünver et al. [15] proposed

a malware classification model for detecting malware samples in the Android environment, the proposed model is based on converting some files from the source of the Android applications into grayscale images, some image-based local features and global features, including four different types of local features and three different types of global features, have been extracted from the constructed grayscale image datasets and used for training the proposed model. This type of features is used for the first time in the Android malware detection domain. Moreover, the bag of visual words algorithm has been used to construct one feature vector from the descriptors of the local feature extracted from each image. The extracted local and global features have been used for training multiple machine learning classifiers. The proposed method obtained a very high classification accuracy reached 98.75 %. Jin et al. [16] converts malicious files into images, designs a CNN-based autoencoder to learn the functional characteristics of malware, and realizes the classification and detection of malware and benign software by observing the reconstruction errors of the autoencoder, requiring only a few training data points, it achieves an accuracy rate of 93 %.

Although the above method reduces the workload, but through the method of interception and feature extraction of only a single file, the source of information acquisition is reduced, and there is a possibility of missing important information. In particular, many researchers choose dex files for conversion and classification, and APK packing technology can package Android malware into normal software by changing the dex file, so that the malware escapes detection. In addition, the size of the converted grayscale image will be different whether it is a single file or a coupling of multiple files, which affects the training and testing of the model, and thus affects the detection effect.

2.3. Classification method based on adversarial training

In recent years, Android malware anti-detection based on adversarial training has received increasing attention. Biggio et al. [17] demonstrated that by leveraging on vulnerabilities of clustering algorithms, an attacker can significantly impact the performance of malware clustering. Only a small fraction of poisoning samples is necessary to largely destroy the recovery of families in a dataset of real malware. Calleja et al. [18] designed and implemented a prototype tool called Iagodroid, which modifies malware samples towards a target family and classifies them into this target family, while preserving the original semantics of the malware samples. Iagodroid successfully misclassified 28 of the 29 representative malware families in the Derbin dataset. Grosse et al. [19] extended the fgsm and jsma algorithms to process binary features to generate adversarial malware samples that achieved a misclassification rate of 63 %. Stokes et al. [20] studied six different strategies to craft adversarial malware samples based on removing malicious features and augmenting benign ones.

The more popular method is to combine with deep learning and use GAN for adversarial training through data augmentation. Hu et al. [21] proposed a generative adversarial network (GAN)-based algorithm malgan, to generate adversarial malware, which is able to bypass black-box machine learning-based detection models. The advantage of our adversarial example generation algorithm is that malgan is able to reduce the detection rate to close to zero. Wang et al. [22] proposed the AdvAndMal general framework, using pix2pix to generate malware variants to expand the training set of the classifier, and the precision of the whole framework was improved from 0.976 to 0.989.

The adversarial training method based on deep learning can quickly realize the expansion of data samples, and can effectively alleviate the problems of insufficient training of classification models and weak detection ability caused by the scarcity of data, but there are also some problems. Although the augmentation of the Android malware data volume is achieved using GAN, it is essentially a simple replication generation of the same samples, and its confrontation only takes place at the level of the data volume, while no relevant changes are made to the distribution of the data in the data domain, which may lead to overfitting of the model and weakening of the model's generalisation ability.

3. The proposed method

3.1. Overall framework and process

The specific workflow of the Android malware adversarial training framework GMADV is shown in Fig. 1. It is divided into three specific stages: 1. Convert the files in the APK to RGB Markov images. 2. Diversity augmentation of RGB Markov images based on GMM-GAN. 3. Model training and classification.

Stage 1. Convert the files in the APK to RGB Markov images:

At this stage, this paper extracts three files which are closely related to the operation, behavior characteristics and resource calls of Android malware programs. In this paper, RGB Markov Image Conversion Algorithm is proposed to convert the extracted files into RGB Markov images of the same size.

Stage 2. Diversity augmentation of RGB Markov images based on GMM-GAN:

At this stage, this paper introduces the concepts of image content and image style to extract the content features of Android malware based on content loss and the style features of other families based on style loss, and then recombines these two features to realise the horizontal generation of variants of Android malware.

This paper proposes the idea of "regression for generation". Based on Gaussian process regression, the link of sample generation is constructed through time series modeling, and the intermediate state is presented. In the face of small samples, using fewer style images is to achieve iterative generation of style images, longitudinally generate and expand the quantity of samples.

Construct a GMM-GAN based on the above two aspects, to expand the inter-class variant diversity.

Stage 3. Model training and classification:

At this stage, this paper inputs the RGB Markov images converted by Android malware in original dataset and the variant images generated based on GMM-GAN into VGG13 for adversarial training to strengthen the classification performance of VGG13.

3.2. Data characterization of android malware

3.2.1. Markov models applied to malware classification

Yuan et al. [23] proposes a Markov image of malware binary, and gives the conversion principle. Each individual file in the Android malware APK is output as a continuous stream of bytes, and its random process can be expressed as:

$$B_n, n \in \{0, 1, ..., M-1\} \tag{1}$$

Among them, M represents the byte length of different files, B_n represents the value of the nth byte, that is $B_n \in \{0, 1, ..., 255\}$, if C_1 and C_2 are two adjacent bytes, according to the Markov random field theory, the probability of C_2 occurrence is only determined by The previous byte C_1 is determined, thereby obtaining the transition probability matrix of $P(C_2|C_1)$:

$$P(C_2|C_1) = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,255} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,255} \\ \vdots & \vdots & p_{c_1,c_2} & \vdots \\ p_{255,0} & p_{255,1} & \cdots & p_{255,255} \end{bmatrix} \tag{2}$$

Among them, P_{c_1}, P_{c_2} are transition probabilities, c_1 and c_2 are the actual values of C_1, C_2. Respectively, the calculation formula of P_{c_1}, P_{c_2} is as follows:

$$P_{c_1,c_2} = \frac{P(c_1, c_2)}{P(c_1)} = \frac{P(C_2|C_1)}{\sum_{j=0}^{255} P(C_j|C_1)} \tag{3}$$

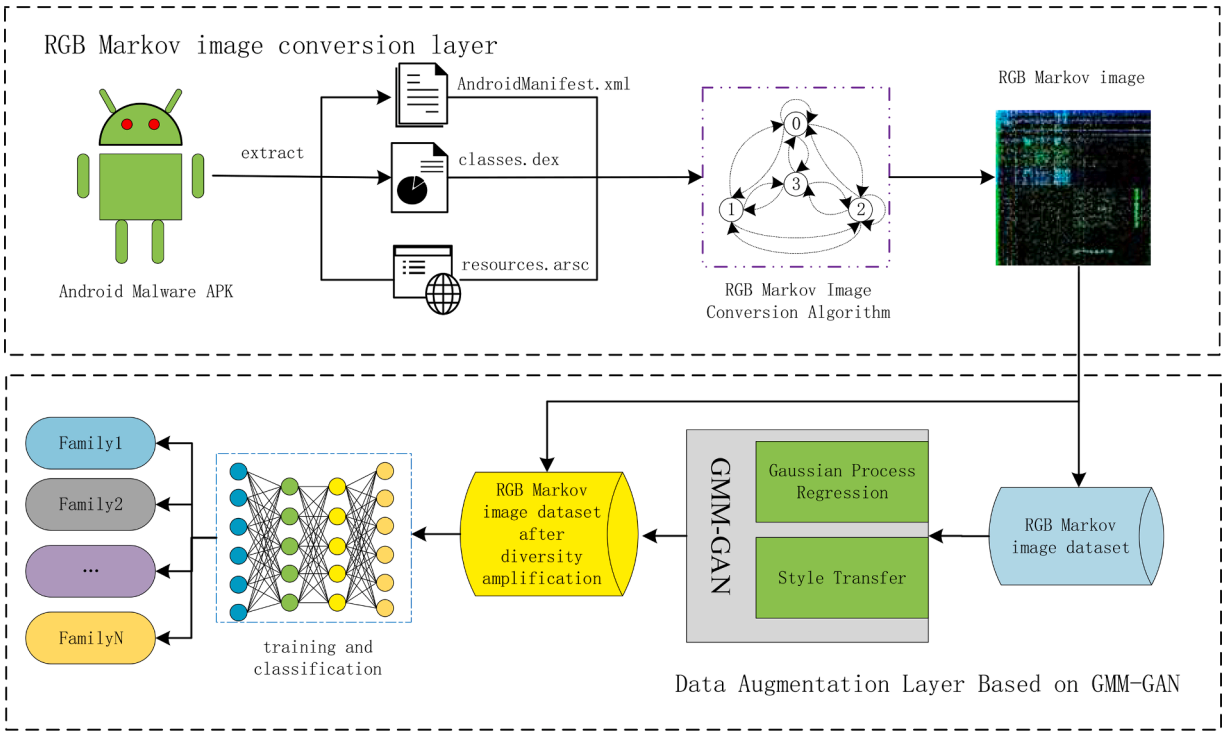


Fig. 1. GMADV Frame Structure.

$P(C_2|C_1)$Indicates the probability that C_1 appears after C_2. The corresponding single-dimensional Markov image can be obtained by rounding and coding the transition probability matrix.

3.2.2. RGB Markov images and channel filling

This paper expands on the original basis, using the three associated files in the APK for domain transformation and coupling into RGB Markov images.

The Android malware APK package mainly contains 7 files, whose file names and main functions are shown in Table 1. Because AndroidManifest.xml contains configuration information of Android applications, classes.dex stores java code executable files, and resources.arsc contains resource index information, they are closely related to the operation, behavior characteristics and resource calls of Android malware programs. Moreover, the internal calls of the program code have an associated timing logic and the use of file resources has an associated access frequency, which fits the idea of temporal and probabilistic modelling of Markov image transformations. So this paper chooses them as the input source file of the RGB Markov image channel. The transformation process is shown in Fig. 1: RGB Markov image conversion layer in GMADV Frame Structure.

Table 1
APK files and function.

File	The main function
Assets	Resource data that has not been compiled by AAPT
Libs	Store so files
META-INF	Signature file
Res	Project resources
AndroidManifest.xml	Configuration list file
classes.dex	Executable files that store java code
resources.arsc	Resource index table

3.3. Diversity amplification of RGB Markov samples based on GMM-GAN

3.3.1. GMM-GAN model

The model framework of the GMM-GAN proposed in this paper is shown in Fig. 2. First of all, this paper selects samples in a certain type of malicious family, extract its content characteristics, and build content losses. Secondly, select the normal samples of malicious samples or malicious behaviors in other families, extract their style characteristics, and build style losses. Finally, the content loss and style loss are given different weights. After addition, it returns to the generator as a total loss value. Through repeated training, new family samples with the original sample nature characteristics are generated. At the same time, the Gaussian process regression principle is used to predict the characteristics of the style, producing multiple new style for amplifying the RGB Markov image dataset.

3.3.2. Loss function

Content loss is used to evaluate the difference in content and structure between the generated image and the original image by comparing the two images in terms of feature activation at certain levels. By minimising this difference, the generated image is made as close as possible to the original image in terms of content and structure.

Style loss is used to measure the difference in style between the generated image and the reference style image. The style loss focuses on capturing and migrating the stylistic features of the image such as texture, colour distribution, etc. and is computed by comparing the Gram matrix of feature activations, which reflects the correlation between features. Style loss calculates the difference between the Gram matrix of feature activations and tries to minimise this difference so that the resulting image mimics the stylistic features of the reference style image.

The content feature loss calculation in this paper uses the L2 distance. The loss function is shown in formula (4):

$$C\_loss(C, G) = \Sigma(Z - C)^2 \tag{4}$$

The symbol C represents the content image, which is the original image

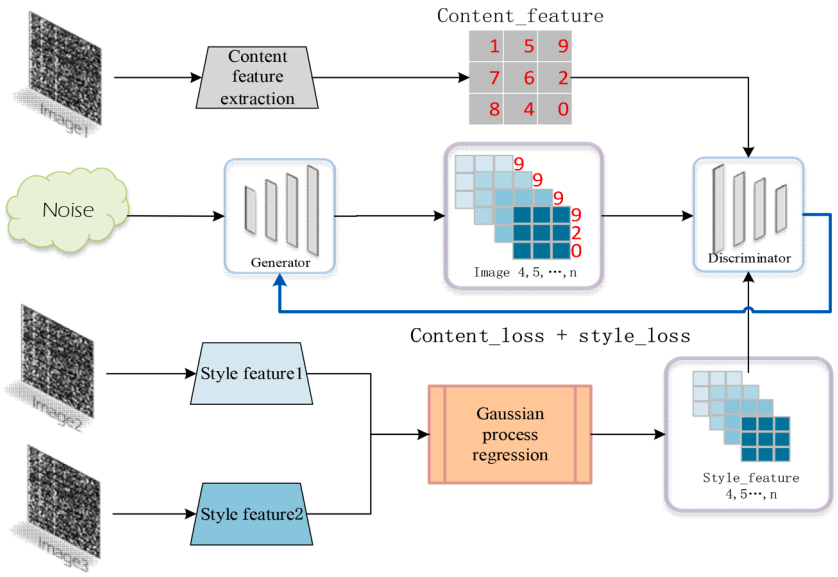


Fig. 2. Framework of GMM-GAN.

whose content is intended to be preserved. G represents the generated image, which is the image obtained after some transformation. Z represents the input vector, which is noise in this case.

When amplifying family sample data, this paper does not only generate the same samples, but also uses style transfer to amplify the diversity of family samples. The style of the image can be controlled through the Gram matrix [24], and the difference between the two image styles is measured by calculating the difference between their Gram matrix. The Gram matrix is specifically expressed as:

$$G_{kk'}^{[l]S} = \Sigma_{i=1}^{n_H^{[l]}} \Sigma_{j=1}^{n_W^{[l]}} a_{ijk}^{[l]S} a_{ijk'}^{[l]S} \tag{5}$$

Where $\alpha_{ijk}^{[l]s}$ is the output of the style image at the (i, j, k) position of the lth layer of the CNN, where(i, j, k) corresponds to the height, width and channel. The style loss function is:

$$S\_loss(S, G) = \Sigma_k \Sigma_{k'} \left(G_{kk'}^{[l]S} - G_{kk'}^{[l]G}\right)^2 \Big/ \left(2n_W^{[l]} n_H^{[l]} n_c^{[l]}\right) \tag{6}$$

At this point, the total loss function is:

$$L(G) = \alpha C\_loss(C, G) + \beta S\_loss(S, G) \tag{7}$$

The symbol S represents the style image, which is the reference image whose style we aim to mimic in the generated image. $C_loss(C, G)$represents the content loss,$S\_loss(S, G)$ represents the style loss, and $L(G)$ represents the total loss. Symbol α and β represent the weight of content loss and style loss. Different weights have different effects. The loss function is schematically shown in Fig. 3. In order to increase the content characteristics of the maximum extent while enhancing diversity. The ratio of content loss to style loss is used in this paper as 1000:1, and this value is tested in the experimental part.

3.3.3. Gaussian process regression and generative modeling

3.3.3.1. Gaussian process. Gaussian process is an important concept in probability theory and statistics. At the same time, as a machine learning algorithm, it is widely used in tasks such as automatic hyperparameter tuning, prediction and classification, and has achieved very good results. In this paper, different from the above approach, Gaussian process regression is used for generative models.

Gaussian process consists of two parts, Gaussian refers to Gaussian distribution, and process refers to random process. When the random variable is one-dimensional, it is called a one-dimensional Gaussian

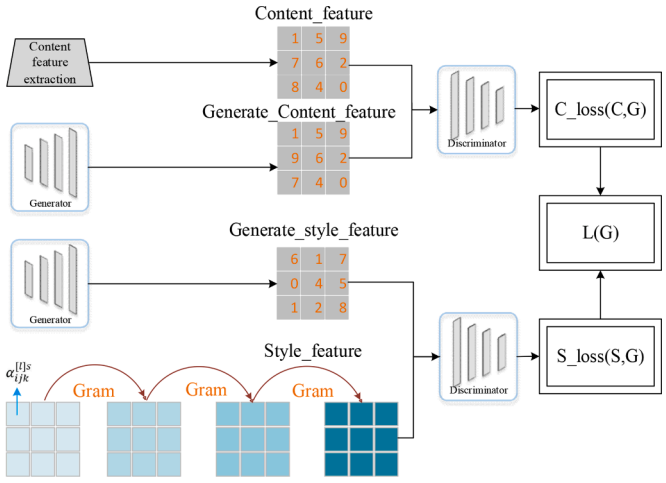


Fig. 3. Schematic of the loss function.

distribution, and the probability density function is:

$$p(x) = N(\mu, \sigma^2) \tag{8}$$

When a random variable rises to finite P dimensions, it is called a high-dimensional Gaussian distribution:

$$p(x) = N(\mu, \Sigma_{p\times p}) \tag{9}$$

Compared with the former, the Gaussian process is a stochastic process composed of infinitely many Gaussian random variables defined on a continuous domain. It can be understood that the Gaussian process is an infinite-dimensional Gaussian distribution. Its rigorous description is: for a continuous domain T, select n times on the continuous domain, $\{t_1,t_2,t_3,\ldots, t_n \in T\}$, so that an n-dimensional vector $\{\xi_1,\xi_2,\xi_3,\ldots, \xi_n\}$ obtained satisfies that it is an n-dimensional Gaussian distribution, then this $\{\xi_t\}$is a Gaussian process.

A p-dimensional Gaussian distribution, its distribution is determined by two parameters, one is the p-dimensional mean vector μ_p, which reflects the expectation of each dimension of random variables in the p-dimensional Gaussian distribution, and the other is the covariance matrix $\Sigma_{p\times p}$, It reflects the variance of each dimension itself and the covariance between different dimensions in a high-dimensional

distribution.

The same as the Gaussian process defined on the continuous domain T, it is an infinite-dimensional Gaussian distribution. It also needs to describe the mean value at each time point t. Because it is on the continuous domain, the dimension is infinite, so it is defined as a function of time t: $m(t)$.

The same is true for the covariance matrix. In the case of infinite dimensions, it is defined as a kernel function $k(s, t)$, where s and t represent any two moments, and the kernel function is also called the covariance function. The kernel function is the core of a Gaussian process, which determines the properties of the Gaussian process. The whole Gaussian process can be described as:

$$\xi_t \sim GP(m(t), k(t,s)) \tag{10}$$

3.3.3.2. Generative modeling. Gaussian process regression can be seen as a process of deriving the posterior based on priors and observations. The Gaussian distribution has a very good feature, that is, the joint probability, marginal probability, and conditional probability of the Gaussian distribution still satisfy the Gaussian distribution. For a set of observations, it has a vector X at all times, then the corresponding value is the Y of another vector of the same latitude, assuming there are 4 sets of observations, that is{(X[[1], Y[1])}, {(X[[2], Y[2])}, {(X[[3], Y[3])}, {(X[[4], Y[4])}, then all the remaining non-observation points, on the continuous domain, we defined as X^*, the value is defined as $f(X^*)$, first of all, the joint distribution satisfies the infinite-dimensional Gaussian distribution:

$$\begin{bmatrix} Y \\ f(X^*) \end{bmatrix} \sim \left(\begin{bmatrix} \mu(X) \\ \mu(X^*) \end{bmatrix}, \begin{bmatrix} k(X,X)k(X,X^*) \\ k(X^*,X)k(X^*,X^*) \end{bmatrix} \right) \tag{11}$$

$$f(X^*)|Y \sim N(\mu^*, k^*) \tag{12}$$

According to the properties of the Gaussian distribution, the conditional distribution in formula (13) is still a high-dimensional Gaussian distribution, which is the new covariance matrix:

$$\mathrm{k}^* = k(X^*,X^*) - k(X^*,X)^{-1}k(X,X^*) \tag{13}$$

In this paper, multiple samples in the same family are modeled as Gaussian processes, and the specific modeling is shown in Fig. 4.

In this paper, multiple samples are modeled as the data situation at different time points, t0, t2, t3, t5 are the observed values, which are the known samples at this time point. t1 and t4 are regression values, which are data samples obtained by regression through Gaussian process. According to the properties of the Gaussian process, t0, t1, t2, t3, t4, and t5 jointly obey the Gaussian distribution. As shown in Fig. 4, the distribution of images from image1 to image6 obeys the Gaussian distribution, and the distribution of images from image7 to image12 also obeys the Gaussian distribution. The images in the Fig. 4 are represented as two different Gaussian distribution, it also symbolises two different families. Each Gaussian distribution has its own k*, which means that each family has its own variant iteration. According to formula (12), formula (13) and formula (14), the covariance matrix can be obtained. The research in this paper finds that the style changes of the images are related to a certain extent, and the iteration of the Gram matrix can be realized through the iteration of the covariance matrix, which can be used for style type modulation. The new style loss function is:

$$S\_loss(S,G) = \Sigma_{\mathrm{k}}\Sigma_{\mathrm{k}'}\left(\mathrm{k}^* - G_{kk'}^{[l]G}\right)^2 \Big/ \left(2n_W^{[l]}n_H^{[l]}n_c^{[l]}\right) \tag{14}$$

3.3.4. Training algorithm of GMM-GAN

The generator and discriminator of GMM-GAN are composed of deconvolutional neural network and convolutional neural network respectively. VGG19 convolutional network is used as content feature extraction and style feature extractor. Its content layer includes block4_conv2 and block5_conv2, and the weighting coefficients are both 0.4; its style layer includes block1_conv1, block2_conv1, block3_conv1, and the weighting coefficients are all 0.15. After the content image and style image are input into the VGG19, the content feature map and the style feature map are obtained respectively. The content feature map remains unchanged, and multiple style feature maps are modeled by Gaussian process, and a new style feature map is obtained according to Gaussian regression. The noise is generated by the generator G and sent to the discriminator D for discrimination. The total loss L(G) is calculated and fed back to the generator. The above operations are repeated until the discriminator D and the generator G reach the Nash equilibrium. The specific training process of GMM-GAN is shown in Algorithm

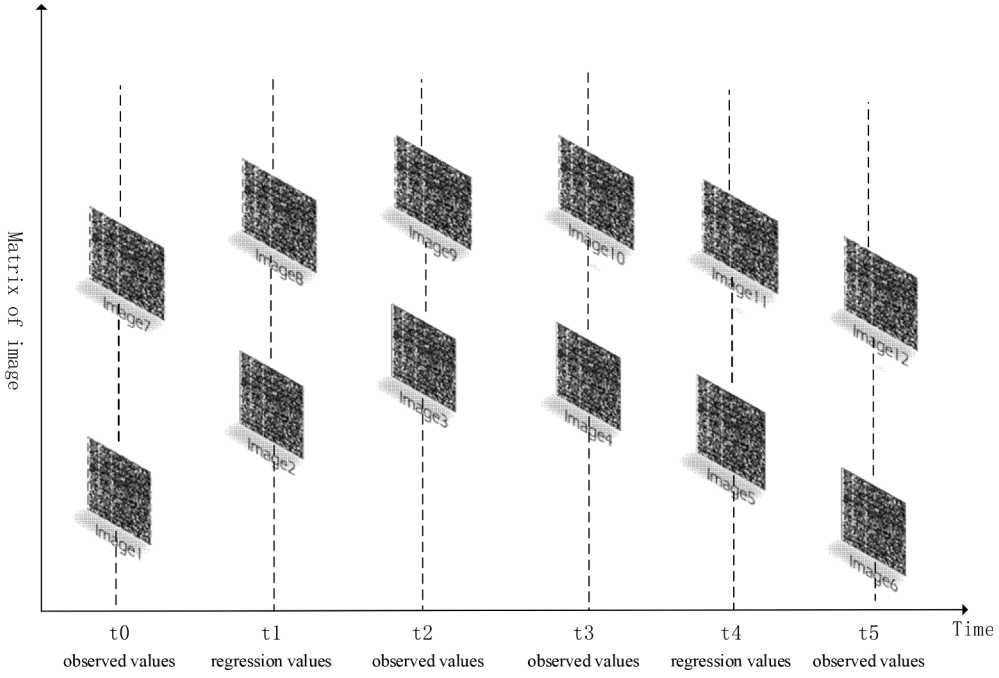


Fig. 4. Gaussian Process Modeling.

1:

Algorithm 1 GMM-GAN, our proposed algorithm

- ***Input:*** *learning rate (α) = 0.02; content_loss_factor(cf)=1000; style_loss_factor(sf)=1; steps_per_epoch=50;*
- ***Output:*** *images; generator parameters (θ_g); discriminator parameters(θ_d); content feature map(cfm); style feature map(sfm); gmm feature map (gfm); Gaussian Process Regression(GPR); Identity(I);*

1: ***while*** θ_g *has not converged* ***do***
2: ***For*** $I = 1,\ldots, m$ ***Do***
3: input RGB Markov image $x \sim P_{data}(x)$
4: input Noise sample $z \sim P(z)$
5: input style image $s \sim P(s)$
6: extract cfm from RGB Markov image $x \sim P_{data}(x)$ *based on content feature extraction layer*
6: extract sfm from style image $s \sim P(s)$ *based on style feature extraction layer*
7: ***For*** $j = 1,\ldots, n$ ***Do***
8: gfm(j)=GPR(sfm)
10: use gfm(j) to do the GAN loss function
11: update $S\_loss(S,G) = \Sigma_k \Sigma_{k'} \left(gfm(j) - G_{kk'}^{[l]G}\right)^2 / \left(2n_W^{[l]} n_H^{[l]} n_c^{[l]}\right)$
12: update $C\_loss(C,G) = \Sigma(Z - C)^2$
13: update $L(G) = cfC\_loss(C,G) + sfS\_loss(S,G)$
14: ***End for***
15: Update Noise sample z
16: ***End for***
*17:****End While***
*18:****Return*** *images*

4. Experimental evaluation

4.1. The datasets and evaluation indicators

4.1.1. Original dataset

A total of three datasets are used in this paper, namely AndroidBotnets, Derbin and CICAndMal2017 datasets. The overall situation of the datasets is shown in Table 2.

The AndroidBotnets dataset [25] is collected by the Canadian Institute of Cyber Security, combining some botnet samples from the Android Genome Malware Project, Malware Security Blog, VirusTotal, and examples provided by well-known anti-malware vendors. It contains a total of 14 families of Android botnet samples, this paper selects four families with more samples of AnserverBot, DroidDream, Geinimi and PJapps, and each family has 244 samples while maintaining the data balance. The above samples are converted into RGB Markov images, used to examine the effect of converting Android malware to RGB Markov images for classification.

The Derbin dataset [26] contains a total of 179 families of Android malware, of which 129 families have less than 10 samples, accounting for 72.07 %. Most families only contain one or two samples. There are 42 families with a sample size between 10 and 100, and only 8 families with a sample size of more than 100, accounting for only 4.47 %. The Derbin dataset has a large variety of families and a large difference in the number of families, which seriously limits the ability of deep learning to classify malicious Android software. In this paper, this data set is used to test the impact of the use of Gaussian process regression and style transfer on the classification effect of variant diversity expansion in the family.

The CICAndMal2017 dataset [27] is provided by the Canadian Institute of Cyber Security, which mainly includes four major categories, a total of 42 malware families and 4354 malware. Among them, the number of samples in the family is at most 17, and the number of samples in other families basically fluctuates around 10. The sparse number of samples in the family is a major obstacle to the classification research of Android malware. This paper uses this dataset to test the generality of the GMADV framework on different datasets.

Table 2
General view of the dataset.

Dataset	Number of family samples (n)	Number of families	The proportion of the number of families
Derbin	n<=10	129	72.07 %
	10<n<=100	42	23.46 %
	100<n	8	4.47 %
CICAndMal2017	n<=10	33	78.57 %
	10<n<=20	9	21.43 %
	20<n	0	0 %
AndroidBotnets	n<=50	3	21.43 %
	50<n<=100	6	42.86 %
	100<n	5	29.41 %

4.1.2. Augmented dataset

This paper firstly counts the number of Android malware APKs in the original data set, and converts the APKs into RGB Markov images according to a one-to-one correspondence ratio. Then use the GMM-GAN proposed in this paper to augment the RGB Markov image, the augmented data is shown in the RGB Markov image set in the table. Diversification of the same family of Android malware using multiple flavors. This part does not directly generate Android malware, but according to the RGB Markov image converted from the file in the APK, using the generation ability of GMM-GAN to generate a variety of RGB Markov images according to the quantity requirements, and then use it for Android malware Classification. The specific data augmentation of the three datasets is shown in Table 3, Table 4, and Table 5.

4.1.3. Evaluation indicators

This paper uses four indicators of precision, recall, precision and F1 score for evaluation. Next, the relevant definitions and calculation formulas of them are introduced in detail.

Accuracy: The percentage of correctly predicted samples in all samples to be predicted.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

Precision: The initial definition of precision is the proportion of samples classified as positive examples that are truly positive examples. In this paper, the proportion of samples classified as Android malware of a certain family is actually defined as the proportion of software of this family.

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

Recall: The initial definition of recall is how many positive examples are classified as positive examples. In this paper, it is embodied as the proportion of Android malware samples of a family that are correctly classified into samples of this family.

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

Table 3
Selection and expansion of the Derbin.

Family	Original dataset		Augmented RGB Markov image set	
	quantity	proportion	quantity	proportion
Adrd	91	2.17 %	1000	8.33 %
BaseBridge	330	7.87 %	1000	8.33 %
DroidDreame	81	1.93 %	1000	8.33 %
DroidKungFu	667	15.90 %	1000	8.33 %
FakeDoc	132	3.15 %	1000	8.33 %
FakeInstaller	925	22.06 %	1000	8.33 %
Geinimi	92	2.19 %	1000	8.33 %
GinMaster	339	8.08 %	1000	8.33 %
Iconosys	152	3.62 %	1000	8.33 %
Kmin	147	3.50 %	1000	8.33 %
Opfake	613	14.62 %	1000	8.33 %
Plankton	625	14.90 %	1000	8.33 %
Total	4194	100 %	12,000	100 %

Table 4
Selection and expansion of the CICAndMal2017.

Family	Original dataset		Augmented RGB Markov image set	
	Quantity	proportion	quantity	proportion
Dowgin	10	9.09 %	100	10 %
Mobidash	10	9.09 %	100	10 %
Koler	10	9.09 %	100	10 %
Svpeng	11	10.00 %	100	10 %
AndroidDefender	17	15.45 %	100	10 %
FakeApp	10	9.09 %	100	10 %
Biige	11	10.00 %	100	10 %
Jifake	10	9.09 %	100	10 %
Plankton	10	9.09 %	100	10 %
Nandrobox	11	10.00 %	100	10 %
Total	110	100 %	1000	100 %

Table 5
Selection and expansion of the AndroidBotnets.

Family	Original Dataset (Selected)		Augmented RGB Markov image dataset	
	quantity	proportion	quantity	proportion
AnserverBot	244	25 %	500	25 %
DroidDream	244	25 %	500	25 %
Geinimi	244	25 %	500	25 %
PJapps	244	25 %	500	25 %
Total	976	100 %	2000	100 %

F1_score: F1 score is a precision evaluation metric that correlates both precision and recall, taking into account both.

$$F1\_Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{18}$$

The accuracy rate can represent the overall classification effect of the model on Android malware, and the recall rate and precision rate can locally represent the classification effect of the model on different Android malware families. Because the variant diversity augmentation in this paper can help the model to better perform multi-classification and identify few-sample families, F1_score can better reflect the impact of the above changes on the classification of specific types of malware.

4.1.4. Experimental platform and configuration

The training and testing of the model in this paper is done in the Windows environment, and the configuration parameters are AMD Ryzen 7 5800H CPU 3.2 GHz, 16GB RAM and external GPU (NVIDIA GeForce RTX 3060). The relevant code is written in Python language and is based on the Tensorflow 2.4 framework.

4.2. Experimental results and analysis

4.2.1. Experiments on RGB Markov image representation and classification

4.2.1.1. RGB Markov image representation. Based on the Derbin malware dataset, this paper randomly selects 10 categories to convert to color Markov maps, including some but not limited to the categories in Table 3, and the image size is 256×256. Specifically as shown in Fig. 5.

In Fig. 5, the two images in the same column are converted from different samples of the same family. It can be compared that the RGB Markov images converted by Android malware in the same family have very similar texture feature distributions, and the texture features of RGB Markov images converted by different families are significantly different, which proves that it is reasonable and feasible to convert Android malware into RGB Markov images for classification.

4.2.1.2. Comparison of data representation classification based on multi-file and single-file. Based on the four families of malicious samples selected from the AndroidBotnets dataset, firstly, according to the steps of the data conversion layer in the GMADV framework, all samples are converted into RGB Markov images. Then the 244 samples are divided into training set and test set according to the ratio of 8:2 for training and testing. In addition, convert the dex files in all selected APKs into single-channel grayscale images with the same size as RGB Markov images, and put them into VGG13 for classification. The normalized confusion matrices obtained respectively are shown in Fig. 6. At the same time, this paper calculates the F1_score under the two experiments based on the confusion matrix, and the specific data is shown in Fig. 7.

It can be clearly seen from Fig. 6 that both RGB Markov images and grayscale images have better classification results. However, compared with grayscale images, the classification effect of multi-file RGB Markov image data representation is better. And according to the numerical statistics of F1_score in Fig. 7, the accuracy of RGB Markov images in multi-classification is also excellent. It shows that multiple files can better represent some unknown and important characteristics of Android malware compared to using a single file for classification. The features extracted based on AndroidManifest.xml file and resources.arsc play an important role in classification when the extracted features change due to possible shelling of dex file. And to a certain extent, it solves the problem of anti-adversity analysis of Android malware. The Markov model is based on the modeling characteristics of probability, which can convert and use all the selected files and retain more information. The data representation method of RGB Markov image normalization is more conducive to model training and classification, which improves the classification effect.

4.2.2. Diversity amplification experiment based on GMM-GAN

4.2.2.1. Family variant diversity amplification. According to the statistics of the original data set in the previous paper, there is a small number of Android malware in some families, which seriously limits the classification ability of the model. This paper uses the proposed GMM-GAN for data augmentation of Android malware while improving sample diversity within the family. The resulting image is shown in Fig. 8.

The original image is the original sample of the RGB Markov map to be multiplied, which is used to extract content features, and the style image is used to provide initial style features. The style transfer image is an image directly generated by using Origin image and Style image. The GMM-GAN image is generated by combining the original image and multiple styles of images generated after multiple iterations using Gaussian process regression. According to the image, the original image, the horizontally generated image by style transfer and the GMM image have very similar texture features, and at the same time have their own styles. This experiment proves that the proposed GMM-GAN can ensure the enhancement of sample data volume, and at the same time enrich the sample diversity based on Gaussian process regression.

4.2.2.2. Comparison of GMADV with other adversarial training methods. This experiment compares the classification effect of the original malware dataset with data augmentation using the AdvAndMal framework and the augmented dataset using GMADV to test the performance of GMADV. The number of amplified Derbin datasets used is shown in Table 3, which is divided into training set and test set according to 8:2. The original samples were used as much as possible in the test set, and augmented samples were used to supplement if the number was insufficient, and the rest were used as the training set. The evaluation indicators used are precision, recall, F1_Score and accuracy. The experimental data are shown in Table 6.

Table 6 summarizes the classification effects of the data set before amplification and the data set after amplification using AdvAndMal and GMADV respectively under the four indicators of precision, recall,

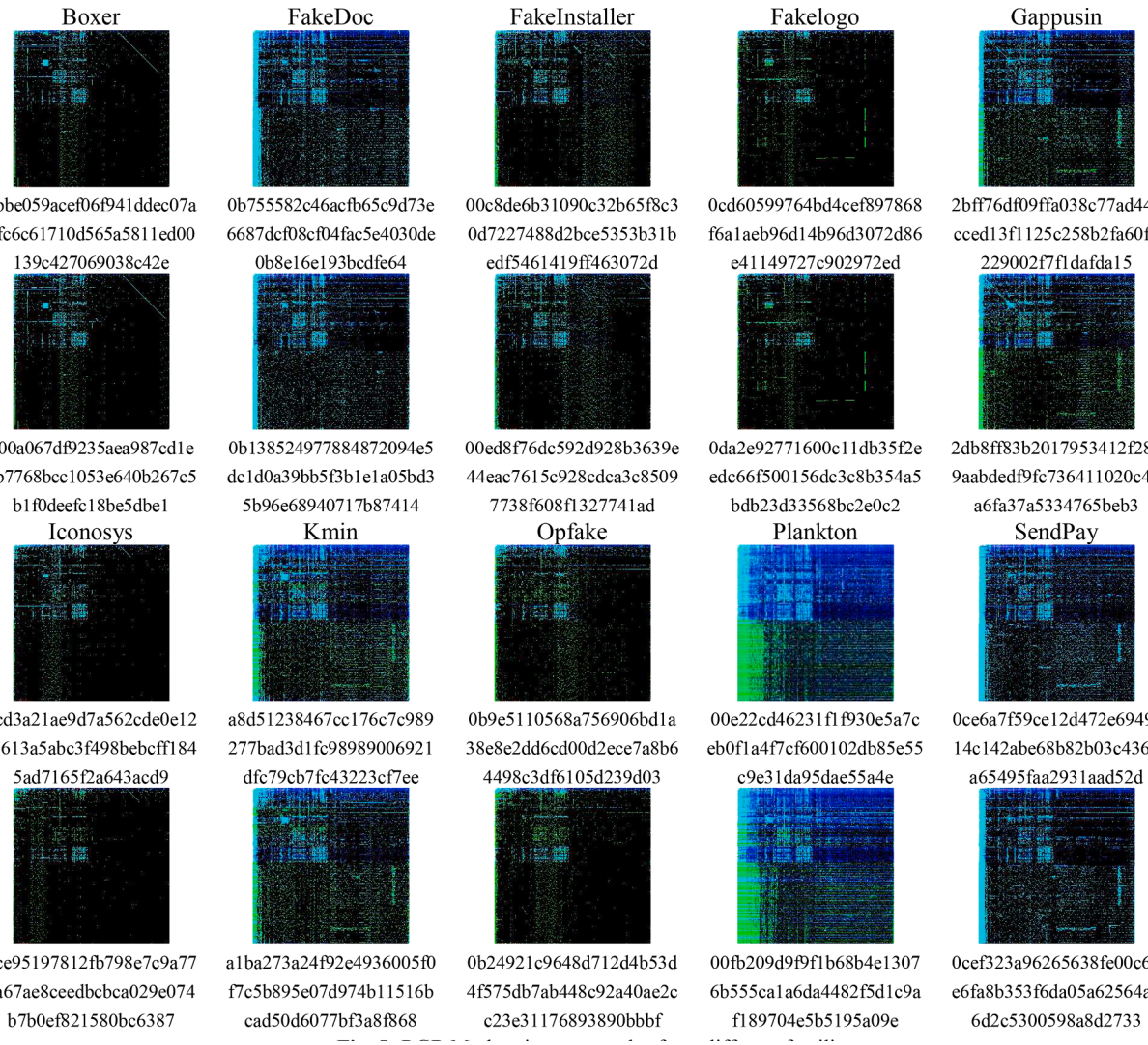
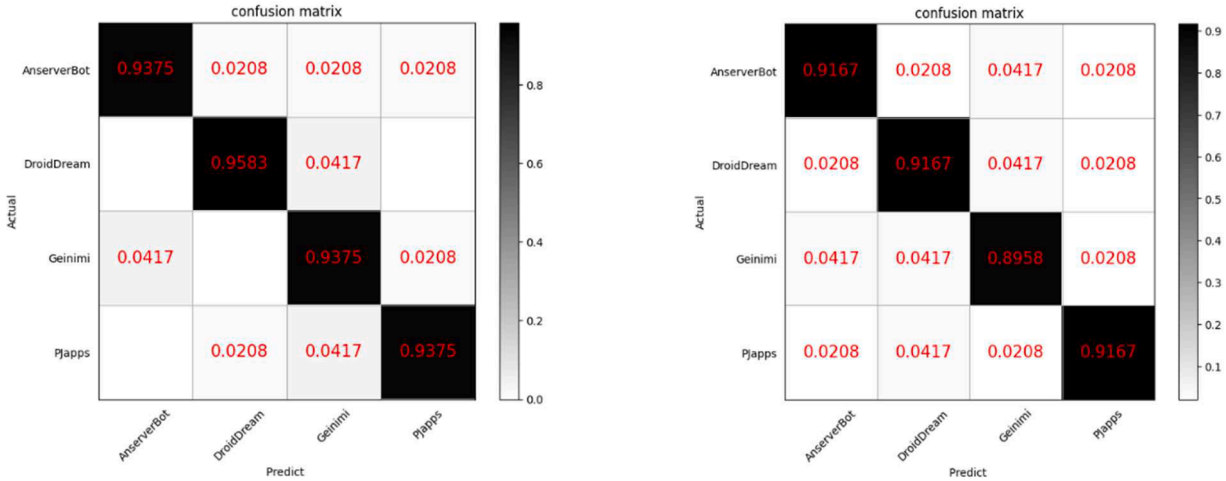


Fig. 5. RGB Markov image samples from different families.



(a) RGB Markov image (b) Grayscale image

Fig. 6. Confusion matrix based on RGB Markov image and Grayscale image.

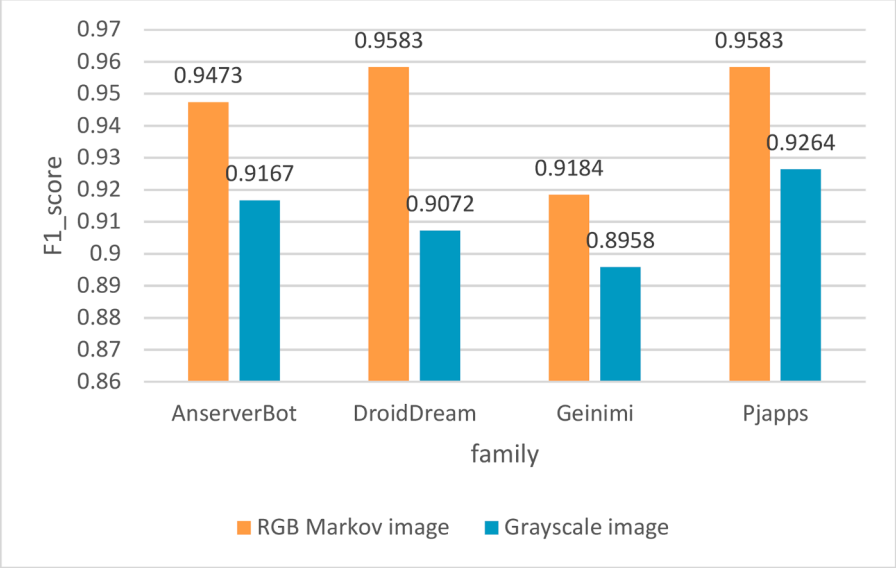


Fig. 7. Comparison of F1_score based on RGB Markov images and Grayscale images.

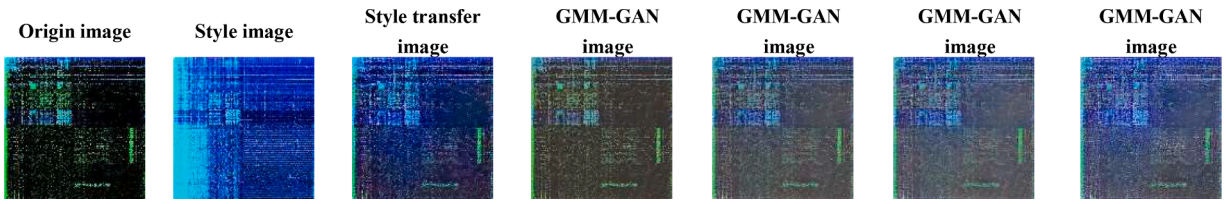


Fig. 8. Amplified images.

Table 6
Comparison of GMADV and AdvAndMal classification results.

Family	Original dataset			AdvAndMal			GMADV		
	Precision	Recall	F1_Score	Precision	Recall	F1_Score	Precision	Recall	F1_Score
Adrd	0.833	0.556	0.667	1	0.833	0.909	0.943	0.914	0.928
BaseBridge	0.868	0.908	0.868	0.909	0.923	0.909	0.928	0.937	0.932
DroidDreame	0.714	0.313	0.435	0.857	0.75	0.8	0.892	0.879	0.885
DroidKungFu	0.84	0.91	0.874	0.94	0.947	0.944	0.935	1	0.966
FakeDoc	0.857	0.923	0.889	0.962	0.962	0.962	0.970	0.974	0.972
FakeInstaller	0.978	0.962	0.97	0.984	0.968	0.975	1	0.979	0.989
Geinimi	0.8	0.444	0.571	1	0.722	0.839	0.976	0.934	0.955
GinMaster	0.63	0.603	0.617	0.836	0.824	0.83	0.875	0.861	0.868
Iconosys	0.806	0.967	0.879	0.829	0.967	0.892	0.962	1	0.981
Kmin	0.967	1	0.935	0.938	1	0.967	1	0.971	0.985
Opfake	0.96	0.976	0.968	0.953	0.984	0.968	0.957	1	0.978
Plankton	0.875	0.896	0.885	0.921	0.936	0.929	0.946	0.965	0.955
Average	0.841	0.788	0.801	0.929	0.901	0.912	0.949	0.951	0.950
Accuracy	0.88			0.935			0.992		

F1_Score and accuracy. It can be seen from the table that the precision of the classification model is low in the dataset before amplification due to the scarcity of samples, especially for some categories with a small proportion of samples, such as DroidDreame and Geinimi, compared with families with a large proportion of variants The classification effect is worse, which affects the classification effect of the deep network. After the expansion of the number of samples, the classification effect of the randomly selected 12 types of samples has reached a high value, and compared with Pix2pix, the average value of the three evaluation indicators of GMM-GAN has further increased. This experiment proves that the proposed GMM-GAN can ensure the increase of sample data volume, and at the same time enrich the diversity of samples in the family based on Gaussian process regression and style transfer. The classification accuracy using GMADV is also higher than that using AdvAndMal and using the original data set. This experiment proves that the proposed GMM-GAN not only enlarges the amount of sample data, but also enriches the diversity of samples in the family based on Gaussian process regression and style transfer. By expanding the original data domain boundary and increasing the intensity of adversarial training, the classification performance of the GMADV framework has been greatly improved.

4.2.2.3. Comparison with other different detection methods. This experiment mainly explores the accuracy and F1 score achieved by different classification algorithms using different feature extraction methods based on the same dataset. Among them, the datasets used by different

Table 7
Comparison with other different detection methods.

Class	Paper	Feature	Algorithm	Accuracy/F1-score
Raw data volume	[9]	Network traffic	Feature extraction and Data representation	97.89 % (accuracy)
	[14]	APK file	extracting local features and global features	98.05 %(accuracy)
	[28]	Opcode	MGOPDroid	96.35 % (accuracy)
	[29]	API	ProDroid	94.5 % (accuracy)
	[30]	Permission,Opcode	FAMD	97.40 %(accuracy)
	[31]	Static characteristics	JOWMDroid	98.08 % (accuracy)
Data augmentation	[7]	APK file	DCGAN	93.70 % (max F1-score)
	[22]	APK file	AdvAndMal	98.90 % (accuracy)
	Our	APK file	GMADV	99.20 %(accuracy)

methods in the experiment are all Derbin datasets. The experimental results are shown in Table 7.

The data in the table shows that when all the features used are APK file, the method proposed in this paper for adversarial training using data augmentation is superior to using the original data volume for classification. When classifying based on different features such as APK file, Network traffic, and Opcode, the accuracy of the data augmentation method proposed in this paper is also higher than that of classification methods such as using raw data for data representation. This proves that the method of using data augmentation for adversarial training is feasible and effective. In the same category of adversarial training methods, the amplification method GMADV proposed in this paper has a slightly higher accuracy than AdvAndMal and is significantly higher than DCGAN. This proves that sample diversity amplification to increase the intensity of adversarial training is better than simple replication of samples.

4.2.2.4. Exploration of model structure and parameter setting. Most of the neural network models are built on the basis of manual experience, and in order to obtain better results, this paper uses a gradual variation of the number of network layers and network parameters to be adjusted. In this paper, the number of layers is gradually increased when extracting style features and content features to achieve a more optimal network structure. Since the features extracted from the deeper layer are more relevant to the sample content and the features extracted from the shallower layer are more relevant to the sample style, this paper extracts the content features and style features from the deepest block5_conv2 and the shallowest block1_conv1 respectively. Meanwhile, this paper also debugs parameters such as Weighting coefficients for the content feature extraction layer, learning rate (α), epoch, and so on. Considering

Table 8
Model structure and parameter tuning results.

selection of content feature extraction layer	selection of style feature extraction layer	weighting coefficients for the content feature extraction layer	learning rate (α)	epoch	accuracy
block5_conv2	block1_conv1	0.3	0.02	50	95.47 %
		0.4	0.02	50	95.51 %
		0.5	0.02	50	95.34 %
	block1_conv1	0.3	0.02	50	96.43 %
	block2_conv1	0.4	0.02	50	96.56 %
		0.5	0.02	50	96.61 %
	block1_conv1	0.3	0.02	50	97.58 %
	block2_conv1	0.4	0.02	50	97.45 %
	block3_conv1	0.5	0.02	50	97.67 %
	block1_conv1	0.3	0.02	50	97.41 %
	block2_conv1	0.4	0.02	50	97.67 %
	block3_conv1	0.5	0.02	50	97.98 %
	block4_conv2				
block4_conv2	block1_conv1	0.3	0.02	50	98.13 %
block5_conv2		0.4	0.02	50	98.05 %
		0.5	0.02	50	98.27 %
	block1_conv1	0.3	0.02	50	98.44 %
	block2_conv1	0.4	0.02	50	98.62 %
		0.5	0.02	50	98.56 %
	block1_conv1	0.3	0.02	50	98.91 %
	block2_conv1	**0.4**	**0.02**	**50**	**99.20 %**
	block3_conv1	0.5	0.02	50	98.92 %
	block1_conv1	0.3	0.02	50	98.81 %
	block2_conv1	0.4	0.02	50	98.73 %
	block3_conv1	0.5	0.02	50	98.82 %
	block4_conv2				
block3_conv1	block1_conv1	0.3	0.02	50	98.36 %
block4_conv2		0.4	0.02	50	98.41 %
block5_conv2		0.5	0.02	50	98.24 %
	block1_conv1	0.3	0.02	50	98.37 %
	block2_conv1	0.4	0.02	50	98.15 %
		0.5	0.02	50	97.89 %
	block1_conv1	0.3	0.02	50	98.28 %
	block2_conv1	0.4	0.02	50	98.19 %
	block3_conv1	0.5	0.02	50	97.96 %
	block1_conv1	0.3	0.02	50	98.04 %
	block2_conv1	0.4	0.02	50	97.87 %
	block3_conv1	0.5	0.02	50	97.98 %
	block4_conv2				

the space, this paper shows some experiments of model building and hyper-parameter debugging, as shown in Table xx. It indicates that the model uses different selection of content feature extraction layer, selection of style feature extraction layer, weighting coefficients for the content feature extraction layer, learning rate (α) and epoch parameters to generate new samples, and then the new samples are used to augment the dataset for classification to get the classification accuracy. This experiment is performed based on Derbin dataset. The result is shown in Table 8.

Based on this experiment, this paper finally determines to use block4_conv2 and block5_conv2 as the content extraction layer, block1_conv1, block2_conv1 and block3_conv1 as the style extraction layer, the weighting coefficients for the content feature extraction layer is 0.4, weighting coefficients for the content feature extraction layer is 0.15, learning rate (α) is 0.02, and epoch is 50.

4.2.2.5. The influence of feature proportion and Gaussian process generation on accuracy. In the process of family variant diversity amplification, this paper further explores the extraction proportion of content features and style features and the influence of different sample generation amounts of GMM-GAN on the accuracy rate. Based on the Derbin dataset, the extraction proportions of content features and style features are set to 800:1, 900:1, 1000:1, 1100:1 and 1200:1, and represented by 800, 900, 1000, 1100, and 1200 respectively; Under the condition that the extraction proportion of content features and style features remain unchanged, the number of regression generation using a single content image and a single style image is set to 1, 3, 5, 7, and 9 in a total of 5 cases. In the process of training and classification, it is also divided into training set and test set according to 8:2. The original samples are used for testing. If the quantity is insufficient, the amplified samples are used for supplement, and if the quantity is redundant, they are used for training. Calculate their classification accuracy respectively. The results are shown in Fig. 9.

On the one hand, as the proportion of content features and style features increases, the generated samples maintain more gene homology with the original samples, and there are some variations. At this time, with the generation of variants, the classification accuracy gradually increases. When the proportion of content features to style features exceeds 1000:1, the generated variants are extremely similar to the original samples, the variability is reduced, and the classification accuracy improvement is slightly reduced. On the other hand, when the proportion of content features and style features remains unchanged, the classification accuracy also shows an upward trend with the increase of the number of new variants produced by GMM-GAN according to the Gaussian regression process, but it does not continue to increase with the increase of the number of variants. The experimental results show that the influence of the change in the proportion of content features and style features on the classification accuracy is significantly higher than the change in the number of GMM-GAN generation, and the maximum difference between its variants and the original samples is determined by the former, Gaussian Process regression increases the number of variants. Both achieved improved accuracy through variant amplification.

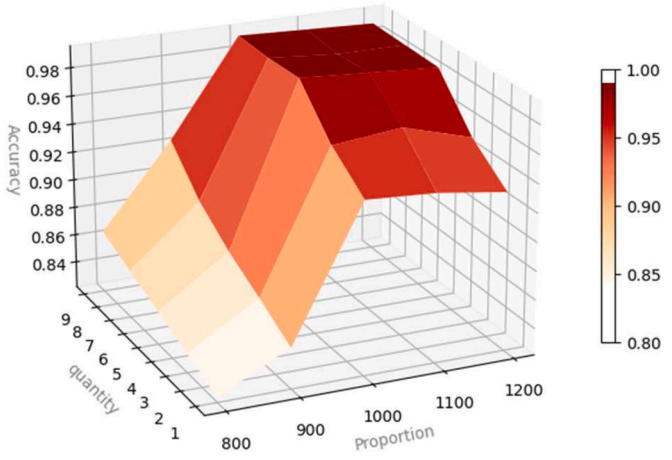


Fig. 9. Influence of characteristic proportion and amplification quantity on accuracy.

4.2.2.6. GMADV migration capability test. In order to detect the transferability of GMADV's family variant diversity amplification method, this paper not only uses the Derbin dataset, but also uses the CICAndMal2017 and AndroidBotnets datasets for testing. The amount of augmented data refers to the table of augmented data sets, and the proportion of training set and test set is 8:2. Using the three indicators of Precision, Recall, and F1_Score to measure, the average values of different family classifications are calculated, and the values are shown in Table 9.

The experimental data show that the GMADV adversarial training method shows good results on the above three data sets, and all F1_Score after adversarial training reaches 95 %, indicating that this method has good universality. Because the AndroidBotnets data set has few families selected, its classification results in F1_ Score is higher.

5. Conclusion

In order to solve the anti-reverse analysis of Android malware, taking into account the interference of APK shelling technology on the code, which reduces the effectiveness of using a single file for characterization to classify. This paper proposes to couple the three files into RGB Markov images of the same size for classification, using the characteristics of Markov model to solve the problem of difficult splicing caused by different file sizes before conversion. Experiments based on the four families of the AndroidBotnets dataset show that the F1_score values obtained by using multi-file classification are higher than those obtained by single-file classification.

In order to solve the problem of poor classification effect based on learning sample data features caused by the small number of samples in some families of Android malware, the GMM-GAN model proposed in this paper horizontally amplifies variants of the same family through recombination between different malicious samples, or through recombination of malicious samples and benign samples, at the same time, a generation idea based on Gaussian process regression is proposed and implemented, which vertically expands the variant samples within the family. On the three datasets, the classification index F1_Score using the augmented data has reached more than 96 %. Diversity amplification expands the boundary of the original family data domain, at the same time, it carries out data interleaving and data interference, and realizes high-intensity confrontation training.

Classifying malicious samples into their corresponding families, selecting representative samples, and providing family information and representative samples to security researchers can greatly improve work efficiency. The GMADV proposed in this paper effectively advances the above work. This paper makes a brave attempt to use Gaussian process regression for generation. In this exploration, we have not paid much attention to the relevant parameters of Gaussian process. In addition, how to use Gaussian process to directly generate variant samples is worth further research.

Table 9
Performance of GMADV on different datasets.

Dataset	Precision	Recall	F1_Score
Derbin	0.949	0.951	0.950
CICAndMal2017	0.953	0.956	0.955
AndroidBotnets	0.965	0.959	0.962

CRediT authorship contribution statement

Shuangcheng Li: Visualization, Writing – original draft. **Zhangguo Tang:** Methodology, Supervision. **Huanzhou Li:** Formal analysis, Project administration. **Jian Zhang:** Conceptualization, Data curation. **Han Wang:** Investigation, Software. **Junfeng Wang:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] STATCOUNTER, 2022, Operating System Market Share Worldwide. [Online]. Available: https://gs.statcounter.com/os-market-share.
[2] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: effective and explainable detection of android malware in your pocket," in Ndss, 2014, vol. 14, pp. 23–26.
[3] Li C, Mills K, Niu D, Zhu R, Zhang H, Kinawi HJIA. Android malware detection based on factorization machine. IEEE Access 2019;7:184008–19.
[4] Xu K, Li Y, Deng RH, Chen K. Deeprefiner: multi-layer android malware detection system applying deep neural networks. 2018 IEEE European symposium on security and privacy (EuroS&P). IEEE; 2018. p. 473–87.
[5] Vasan D, Alazab M, Wassan S, Naeem H, Safaei B, Zheng QJCN. IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture. Comput Netw 2020;171:107138.
[6] Singh A, Dutta D, Saha A. MIGAN: malware image synthesis using GANs. Proc AAAI Conf Artif Intell 2019;33(01):10033–4.
[7] Chen Y-M, Yang C-H, Chen G-C. Using generative adversarial networks for data augmentation in android malware detection. In: 2021 IEEE conference on dependable and secure computing (DSC). IEEE; 2021. p. 1–8.
[8] K. Simonyan and A.J. a. p. a. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv, vol. 1409, p. 1556, 2014.
[9] Wang S, et al. A mobile malware detection method using behavior features in network traffic. J Network Comput Appl 2019;133:15–25.
[10] Zhou J, Niu W, Zhang X, Peng Y, Wu H, Hu T. Android Malware Classification Approach Based on Host-Level Encrypted Traffic Shaping. In: 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). IEEE; 2020. p. 246–9.
[11] Hu W, Tao J, Ma X, Zhou W, Zhao S, Han T. Migdroid: detecting app-repackaging android malware via method invocation graph. In: 2014 23rd International Conference on Computer Communication and Networks (ICCCN). IEEE; 2014. p. 1–7.
[12] Hasegawa C, Iyatomi H. One-dimensional convolutional neural networks for Android malware detection. 2018 IEEE 14th international colloquium on signal processing & its applications (CSPA). IEEE; 2018. p. 99–102.
[13] Hsien-De Huang T, Kao H-Y. R2-d2: color-inspired convolutional neural network (cnn)-based android malware detections. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE; 2018. p. 2633–42.
[14] Bakour K, Ünver HMJNC, Applications. DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques. Neural Comput Appl 2021;33(18):11499–516.
[15] Ünver HM, Bakour K. Android malware detection based on image-based features and machine learning techniques. SN Appl Sci 2020;2:1–15.
[16] Jin X, Xing X, Elahi H, Wang G, Jiang H. A malware detection approach using malware images and autoencoders. In: 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). IEEE; 2020. p. 1–6.
[17] Biggio B, et al. Poisoning behavioral malware clustering. In: Proceedings of the 2014 workshop on artificial intelligent and security workshop; 2014. p. 27–36.
[18] Calleja A, Martín A, Menéndez HD, Tapiador J, Clark DJESwA. Picking on the family: disrupting android malware triage by forcing misclassification. Expert Syst Appl 2018;95:113–26.
[19] Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P. Adversarial examples for malware detection. In: European symposium on research in computer security. Springer; 2017. p. 62–79.
[20] Stokes JW, Wang D, Marinescu M, Marino M, Bussone B. Attack and defense of dynamic analysis-based, adversarial neural malware detection models. In: MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM). IEEE; 2018. p. 1–8.
[21] Hu W, Tan Y. Generating adversarial malware examples for black-box attacks based on GAN. In: International Conference on Data Mining and Big Data. Springer; 2022. p. 409–23.
[22] Wang C, Zhang L, Zhao K, Ding X, Wang XJS. AdvAndMal: adversarial training for Android malware detection and family classification. Symmetry 2021;13(6):1081.
[23] Yuan B, et al. Byte-level malware classification based on markov images and deep learning. Comput Secur 2020;92:101740.
[24] L.A. Gatys, A.S. Ecker, and M.J. a. p. a. Bethge, "A neural algorithm of artistic style," arXiv preprint arXiv, vol. 1508, p. 06576, 2015.
[25] Abdul Kadir AF, Stakhanova N, Ghorbani AA. Android botnets: what urls are telling us. In: International Conference on Network and System Security. Springer; 2015. p. 78–91.
[26] Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. In: 2017 International conference on information networking (ICOIN). IEEE; 2017. p. 712–7.
[27] Lashkari AH, Kadir AFA, Taheri L, Ghorbani AA. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST). IEEE; 2018. p. 1–7.
[28] Tang J, Li R, Jiang Y, Gu X, Li Y. Android malware obfuscation variants detection method based on multi-granularity opcode features. Future Generation Computer Systems 2022;129:141–51.
[29] Sasidharan SK, Thomas C. ProDroid—An Android malware detection framework based on profile hidden Markov model. Pervasive Mob Comput 2021;72:101336.
[30] Bai H, Xie N, Di X, Ye Q. Famd: a fast multifeature android malware detection framework, design, and implementation. IEEE Access 2020;8:194729–40.
[31] Cai L, Li Y, Xiong Z. JOWMDroid: android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. Comput Secur 2021;100:102086.

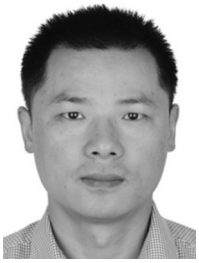
SHUANGCHENG LI was born in 1998. he obtained a bachelor's degree in Communication Engineering from Qingdao Agricultural University in 2020. At present, he is studying for a master's degree in electronic information at Sichuan Normal University. He is involved in several scientific research projects. His recent research interests include intrusion detection and pattern recognition.

ZHANGGUO TANG was Born in 1978. He received his bachelor's degree in electronic information engineering and master's degree in software engineering from the University of Electronic Science and Technology of China in 2005 and is currently studying for a doctor's degree in Cyberspace Security from Sichuan University. He has been an associate professor at Sichuan Normal University since 2012. He is the author of one book, more than 20 papers, and has undertaken more than 20 scientific

HUANZHOU LI was born in 1974. He received a bachelor's degree in applied electronic technology from Sichuan Normal University in 1992, a master's degree in computer application from the University of Electronic Science and Technology of China in 1999, and a doctor's degree in information security from Sichuan University in 2003. He has been working at Sichuan Normal University since 1996. He is a reserve candidate for the ninth batch of academic and technological leaders in Sichuan Province. He presided over more than 30 scientific research projects, participated in the drafting of three national standards, and published more than 50 academic papers. His recent research interests include Internet of things security and big data security.

JIAN ZHANG was born in 1976. She received a bachelor's degree in applied electronic technology from Sichuan Normal University in 1998, a master's degree in information and communication engineering from Sichuan University in 2005, and a doctor's degree in communication and information systems from Sichuan University in 2012. She has been working at Sichuan Normal University since 1998. She has presided over more than 10 scientific research projects and published more than 10 academic papers. Her recent research interests include image processing and intelligent security.

HAN WANG was born in 1997. In 2020, he obtained a bachelor's degree in electronic information engineering from Sichuan Technology and Business University and is currently pursuing a master's degree in network security at the School of Physics and Electronic Engineering, Sichuan Normal University. He is currently researching DNS traffic collection analysis, domain name detection, and classification, deep learning.



JUNFENG WANG received the M.S. degree in Computer Application Technology from Chongqing University of Posts and Telecommunications, Chongqing in 2001 and Ph.D. degree in Computer Science from University of Electronic Science and Technology of China, Chengdu in 2004. From July 2004 to August 2006, he held a postdoctoral position in Institute of Software, Chinese Academy of Sciences. From August 2006, Dr. Wang is with the College of Computer Science and the School of Aeronautics \& Astronautics, Sichuan University as a professor. He serves as an associate editor for several international journals. His recent research interests include network and information security, spatial information networks and data mining.