Performance evaluation of deep neural network on malware detection: visual feature approach

V. Anandhi1,2 · P. Vinod3 · Varun G. Menon1,2 · Korankotte Manoj Aditya1,2

Received: 24 January 2022 / Revised: 2 July 2022 / Accepted: 26 July 2022 / Published online: 18 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Nowadays, several malicious applications target computers and mobile users. So, malware detection plays a vital role on the internet so that the device is secure without any malicious activity affecting or gathering the useful content of the user. Researches indicate that the vulnerability of adversarial attacks is more in deep neural networks. When there is a malicious sample in a family, there will not be many changes in the variants, but there will be more signatures. So, a deep learning model, DenseNet was used for detection. The adversarial samples are created by other types of noise, including the Gaussian noise. We added this noise to a subset of malware samples and observed that for Malimg, the modified samples were precisely identified by the DenseNet, and the attack cannot be done. But for BIG2015, we found that there was some marginal decrease in the performance of the classifier, which shows that the model performs well. Further, experiments on the Fast Gradient Sign Method (FGSM) were conducted, and it was observed that a significant decrease in classification accuracy was detected for both datasets. We understand that deep learning models should be robust to adversarial attacks.

Keywords Deep neural network · Deep learning · Adversarial attack · DenseNet · Noise · Fast gradient sign method

1 Introduction
Malware causes catastrophic damage to a computer or network of computers. The malicious software is designed to cause hardware failure, slow down the system speed, restrict access to our own files, pop up unwanted programs, send spam messages from the affected system to any network, etc. [1].

At the end of March 2020, when COVID-19 started spreading worldwide, people started working from home and students started attending virtual classrooms. Internet usage has increased drastically. Due to this, security has become a real issue as the network area is widespread. As seen in [2], it is clear that malware threats have increased day by day and reached a maximum in the mid of March. In [3], the Lab report specifies 12,755,097 COVID-19 related malicious files were detected from June 2020 to June 2021. The attackers would like to leverage this pandemic situation as an opportunity to attack organizations and the computers used by professionals at home too. Many systems are connected online. This gives a chance to the attacker to infiltrate the information.

Machine learning is a major application of artificial intelligence. Machine learning is best used in medical diagnosis, image processing applications, image prediction, image classification, etc. The computational systems built on machine learning algorithms can learn from past experience or previous data. Deep learning is a subset of machine learning that works on different layers to create a deep neural network that can learn and make decisions intelligently by itself.

Classical methods for the detection of malware are slow [4]. Traditionally malware detection is performed through signature-based methods, where signatures are unique string/bytes used to identify malicious payload. In this approach a signature repository of known malware is
created and used to predict samples using exact/in-exact string matching techniques. Even though signature-based methods are fast, they pose certain limitations such as (a) size of signature database exponentially increases with appearance of new malware samples (b) technical expertise is required to build signatures and (c) pattern matching techniques fail to detect new/obfuscated/encrypted malware files.

Malware detection can be performed using a static and dynamic approach. The malware identity is extracted in the static approach, which requires human expertise. Moreover, the approach is bypassed when obfuscated techniques are used. In the dynamic approach, the code is executed, and the features are analyzed in the run time environment. Dynamic analysis is less vulnerable to obfuscated techniques.

Pascal et al. [5] propose an analysis of the latest malware and detection techniques. The study also gives details on static and dynamic malware analysis based on behavior-level machine learning and deep learning methods. Their paper focus on malware detection in Windows and Android OS.

In [6, 7], it is observed that the images which belong to the same family of the data set are visually similar. Even though, the samples do not change functionally, they differ in their code structure. This result in different signatures for variants belonging to identical families. However, the generic code representative of the family is preserved, which can be carved out using the Convolution Neural Network (CNN). CNN has been found prominent for image classification tasks. Perhaps the fundamental problem in CNN is that, as we go deeper into the network, the performance will degrade due to the vanishing gradient problem. This is because the gradient declines as the network goes deeper, and the layers do not learn. So, a deep learning model called DenseNet is employed in our work to detect malicious samples. A DenseNet uses skip connections, thereby the \( n^{th} \) layer will receive input from all previous \( n - 1 \) layers, which results in more interconnections. These connections form dense paths which learn better comparing traditional neural network models. The advantages of DenseNet architecture are overcoming the vanishing gradient problem, propagation of features through the network, and decreasing the parameter count.

Malware detection is very crucial with its presence in the webspaces. Anti malware systems protect computers and interconnected devices from malicious programs and cyber attacks. This detection method keeps the hackers by not attacking the computer and prevents the information from getting misused by others [8]. Recent researches [34, 35] indicates a threat to the existing deep learning algorithms. Adversaries can fool the models by infusing the attributes of legitimate files into the feature space of malicious executable. This forces machine learning-based classifiers to wrongly predict the malware instance as legitimate, bringing down the overall performance of the detection system. Existing machine learning models are rarely evaluated on adversarial examples before deployment. Hence in this article, we attempt to build detection models and additionally evaluate their performance on poisoning and evasion attacks.

The contributions of this paper are

- A malware detection system was developed using DenseNet architecture which resulted in 99.98\% accuracy. The model was constructed using large collection of executables from public benchmark dataset.
- The proposed detection model predicts unseen/novel samples with prediction time comparatively lesser to the results proposed in the state-of-the-art approaches.
- Security analysis of the detection model was evaluated by generating adversarial examples using Fast Gradient Sign Method and additive noise. We showed that the DenseNet model was found to be resilient to the poison and evasion attack.

In this paper, the summarization of the adversarial attacks to the state-of-the-art is presented. It also provides the performance characteristics of the adversarial attack and the defense techniques measures. The organization of the paper is as follows: In Sect. 2, the related works are discussed. In Sect. 3, a background of adversarial attacks and the threat models are given. In Sect. 4, the detailed evaluation of the proposed attack method is presented. In Sect. 5, the results and discussions are summarized and in Sect. 6, the conclusion of the paper with future works is discussed.

## 2 Related work

In this section, a brief review of the works carried out on machine learning, deep learning, and other similar methods is done. Also, an explanation of the related works on the main attacks in image classification methods is presented.

### 2.1 Detection based on machine learning

Peiravian and Zhu [9] proposed a machine-learning algorithm to extract the permission from profile information of every App. The APIs can be extracted from unpacking the App file. The App is identified as malicious or not by using features they used to characterize each App.

Xu et al. [10] introduces a framework for the detection of malware focusing on kernel aspect and memory access. This provides hardware assistance with better automation.
and by reducing the user’s input on malware signatures. The key work is that malware changes the flow of data structures and access program memory which in turn increase the process of automation.

Baptista et al. [11] introduces a novel approach for the detection of malware which is based on visualization techniques and self-organizing networks. Their accuracy measure for malware content was experimentally determined in .pdf and .doc files as 91.7% and 94.1% respectively.

Bakour et al. [12] extracts three varieties of global features and four varieties of local features from grayscale images. These grayscale images are involved in training the model. This way of feature generation was used in Android device’s malware detection. They have extracted these seven different features for training using standard classifiers used in machine learning methods.

Yifei et al. [13] propose a visual technology-based malware detection on the BIG2015 dataset. A deep neural network architecture called SERLA was used to improve the network performance. They also used data augmentation to improve the detection and classification employing three-channel RGB images.

### 2.2 Detection based on deep learning

Yuxin and Siyi [14] propose a deep belief network that uses unlabeled data to train the layered model. They have compared the evaluation measures of the network with state-of-art detection models and found that it classifies better with the features of the data sample. If more features are selected automatically, it classifies better.

Hardy et al. [15] proposes an intelligent malware detection using stacked autoencoders, which are the building blocks of deep neural networks. Training is performed by unsupervised learning and fine-tuning by supervised learning methods.

Oliveira and Sassi [16] propose a combination of manual and automatic feature extraction by using deep learning architectures. The dataset composed of the Android application Omniodroid. Each application was analyzed using both static and dynamic data analysis. Their work includes DNNs, CNNs and transformer networks for detection purposes.

Hung and Kao [17] extracted the hexadecimal values and developed RGB images to build CNN based Android classification model. The system works effectively in known and unknown Android malware. Yang and Wen [18] converted the APK contents to grayscale images. With the obtained GIST image descriptor, these features were used to create a Random Forest classifier. Karimi and Moattar [19] identified the ransomware using opcode sequences of length 2, and subsequently created grayscale images. Finally they classified malware files using Linear Discriminant Analysis (LDA).

### 2.3 Others—using similarity methods

The similarity measure is a method of measuring relative differences in data samples to each other whereas, the dissimilarity measure tells us how much the data objects are different. Mainly these measures are used in clustering methods where one cluster has similar data samples. All other dissimilar data samples are grouped into other clusters. These similarity methods are used in classification methods, where the data objects are labeled based on the features’ similarity. A numerical value is given for the similarity measure. The measure is higher for alike data samples else the measure is 0, which indicates low similarity.

Gabel and Godhardt [20] proposes automatic acquisition of similarity measure by performing k-nearest neighbor classification/regression on the UCI Machine Learning Repository. In the 19 domains considered, there are 11 domains in which the performance were approximately equivalent or superior to the previously existing methods.

Mathisen et al. [21] described how to automate the two similarity measures. The two similarity measures were worked on several varied datasets and the results showed that the classifier gives low training time for the state of art performance.

### 2.4 Adversarial attack

Demetrio et al. propose black-box attacks that preserve the functionality of benign content. The attacks focused on static Windows malware detectors and not on dynamic analysis and the evaluation has only a few queries and very small payloads [22, 23].

Benjamin et al. designed DNN such that an adversary fools an image classifier easily by adding noise to the original pixels. The technique was a feature-generation one that evaluates the performance of the model which responds to noisy images [24].

The comparison of the related works are shown in Table 1. Our proposed work differs from the state-of-the-art approach in the following aspects: (a) proposed a malware detection system using DenseNet having the potential to detect malicious programs with a high accuracy and F1-measure. However, the prior work focused on conventional machine learning algorithms and Convolutional Neural Networks which suffers from vanishing gradient problems and (b) evaluation of the robustness of classification models by generating adversarial examples. These tainted samples were used to craft poisoning and evasion attacks. Inspired by the image processing domain
we modeled an adversary having the capacity to create adversarial samples by employing additive noise techniques and Fast Gradient Sign method.

3 Background

Adversarial Machine Learning [25] exploits the classification methods by taking control over the model and trying to develop a malicious attack. However, there is no one complete answer for Deep Learning models nor a completely accepted justification for the adversary images.

3.1 Attack framework

The framework starts with images as input to a learning model and finally classifies the given image as legitimate or malicious [26]. The model code is written as bytes of data. The input space is a list of whole numbers, where $W \subset \{0, ..., 255\}$. The feature extraction can be expressed as $W \rightarrow X$ where $X$ is a 2 dimensional vector space. The prediction function in the output layer is denoted as $f : X \rightarrow R$ where $R$ denotes $\{0, 1\}$, which is equivalent to benign or malware.

Let $T$ be the noise vector with the added noise characteristics. In this scenario, the attacker tries to create adversarial malware to each malware input and give it as input to the model. The attacker applies a practical approach, by transforming that altered input without corrupting its previous behavior. We encode the functionality-preserving methods as a function itself. The output is also a file with the original pattern as the input but with an entire varied representation.

3.2 Threat model

A threat model is classified into different categories [27]. Our paper is based on the knowledge of the attacker. Considering into account the attacker’s knowledge, there are three different attacks: white-box attack, black-box attack, and grey-box attack. In a white-box attack, the attacker has direct access/control to the model whereas in a black-box attack, the attacker does not have access to the model nor knowledge about any data concerned with the defense method. In a grey-box attack, the attacker has direct access to the model but does not have access to any data concerned with the defense mechanism. In our paper, we use the grey-box attack model. This type of attack can be used to evaluate defenses and classifiers where the data

Table 1 Comparison of the related work

| S.No | Author                  | Proposed work                                                                 | Remark                                      |
|------|-------------------------|-------------------------------------------------------------------------------|---------------------------------------------|
| 1    | Peiravian and Zhu [9]   | A feature based system which uses SVM, Decision tree and bagging learning algorithms | Accuracy is 96.39%, Precision is 94.9%, Recall is 94.1% |
| 2    | Xu et al. [10]          | Detection of malware on kernel and memory by analyzing data structures        | Accuracy is 86.5%, False positive rate (FPR) is 32.7% for Logistic regression |
| 3    | Baptista et al. [11]    | Detection of malicious content based on binary visualization and self-organizing networks | Accuracy is 94.1% and False Positives is 2.4% |
| 4    | Bakour et al. [12]      | Grayscale feature generation used with AdaBoost classifier                   | Accuracy is 98.75% and computational time is 0.018 sec |
| 5    | Yuxin and Siyi [14]     | A deep belief network that uses unlabeled data to train the layered model    | Accuracy is 96% and F1 score is 95.8%       |
| 6    | Hardy et al. [15]       | An intelligent malware detection using stacked autoencoders                  | Accuracy of 96.85% in training phase and 95.64% in testing phase |
| 7    | Han et al. [4]          | Static and dynamic API sequences are extracted, and hybrid method used for detection | Detection and classification accuracy is 97.89% and 94.39% respectively |
| 8    | Nataraj et al. [6]      | Data visualized as grayscale images                                          | Accuracy is 98%                             |
| 9    | Yiwei et al. [13]       | Data augmentation technology on the BIG 2015 dataset                         | Accuracy is 98.31%                          |
| 10   | Oliveira and Sassi [16] | Combination of manual and automatic feature extraction                       | Accuracy is 90.9%, Precision is 94.4%, Recall is 86.6% |
| 11   | Hwang and Kao [17]      | RGB images to train CNN based                                                | Accuracy is 97.3%, F1-score 97.25%          |
| 12   | Manzhi Yang and Quoayon Wen [18] | APK contents to grayscale images using GIST                                      | Accuracy is 95.51%                         |
| 13   | Karimi and Moattar [19] | Convert opcode sequence to grayscale using LDA                               | Accuracy is 97.3%, TPR is 92%              |
is not available but the classification model is known. These attacks have a high degree of threat when compared to the other attacks.

During the training stage, we use poisoning attacks where the attacker has full control of the learning model. In this attack, the training dataset is corrupted or changed with adversarial examples to obtain a model which is compatible with the original dataset.

4 Proposed method

In our paper, we propose the evaluation of robustness of the detection model implemented using DenseNet. We have implemented using two adversarial attacks, namely Additive noise and Fast gradient sign method.

In the additive noise method, gaussian, localvar, speckle, salt noise, pepper noise, salt and pepper noise were added to a subset of malware samples and we observe that for the malware dataset the modified samples were precisely identified by the DenseNet model. In the second attack, the datasets are given to the DenseNet through the Fast gradient sign method to generate new adversarial images.

Experiments were conducted on the two public benchmarks dataset, Malimg [6] and Microsoft Malware dataset (BIG2015) [33]. We choose to perform the study on these dataset, as both the dataset have class imbalance, which helps to determine the robustness of the model. The overall framework of our proposed method is shown in Fig. 1.

4.1 Data preprocessing

Fig. 1 illustrates the grayscale image generation process from the one-dimensional array. Hexadecimal codes from malicious executables are extracted using Hex editor [28]. Initially, a frequency matrix is created which record occurrence of consecutive bytes. Later, from the frequency matrix a probability matrix of size 256×256 is derived (refer Eqs. 1, 2). The elements of the probability matrix retain probabilities of two consecutive bytes. Each time two bytes are read from the hexdump. The first byte is used to index the row and the next 8 bits are used to index a column of the probability matrix. At the intersection, a pixel is obtained by multiplying the probability with 255 and scaling to nearest integer. This process is repeated until the length of the malware file, which eventually produces a 256×256 malware image. The model can be described by the following mathematical equation where the probability of byte $X_{i+1}$ is related to byte $X_i$.

\[
P(X_{i+1}|X_0, X_1 \cdots X_i) = P(X_{i+1}|X_i)
\]

\[
P(a, b) = P(b|a) = \frac{f(a, b)}{\sum_{a=0}^{a=255} f(a, b)}.
\]

where $f$ is the frequency of consecutive pixels $a$ and $b$ obtained from frequency matrix. Subsequently a probability of consecutive bytes are computed by dividing with maximum occurrence of two bytes in a row.

4.2 Densely connected networks (DenseNet)

CNN is widely used in the recognition and classification of malware samples represented in the form of images [29–31]. In general, a CNN architecture contains a certain number of Conv2D followed by MaxPool2D layers. The output of convolution and maxpool operation is a new image. The convolution layer is capable of extracting patches in the images using a certain number of filters, and the maxpooling layer produces a subsampled version of images retaining the important attributes useful for the identification of malware from a large collection of executable images. CNN operations can be summarized as:

\[
CNN : x_l = O(x_{l-1})
\]

where $x_l$ denotes the feature map obtained from the $l^{th}$ layer, $O$ is the non-linear operation and $l$ is the index of the specific layer. Perhaps researchers are widely using CNN for malware classification and detection problems, CNN is not suitable in scenarios where wider and deeper networks are required to improve the performance of the detectors. Specifically, CNN faces problems related to exploding or vanishing gradients. To mitigate the aforesaid issues, shortcut connections and summation of feature maps can be employed. To reduce the limitations of CNN, we propose to adopt DenseNet [32], where each layer receives extra inputs from all the preceding layers and passes on its own feature maps to all the other layers. To be precise, all feature maps $x_0, x_1, \cdots, x_{l-1}$ from previous layers are concatenated and propagated to the subsequent layer to generate new feature maps. In general, $f$ feature maps will be obtained after each $O_l$ operation as shown below:

\[
x_1 = O_1(x_0)
\]
\[
x_2 = O_2(concat[x_1, x_0])
\]
\[
x_3 = O_3(concat[x_2, x_1, x_0])
\]
\[
\vdots
\]
\[
x_f = O_f(concat[x_{f-1}, x_{f-2}, \cdots, x_1, x_0])
\]

Despite the fact that each layer output $j$ number of feature maps, it would mean that the network in general will have a
Fig. 1 Proposed system architecture of Malware Detection Model
large number of inputs. Typically, this is handled by introducing a bottleneck layer which is implemented by adding a 1×1 convolution layer before a 3×3 convolution layer. This achieves a reduction in feature maps and minimises computational cost.

4.3 Attacks using additive noise

To simulate adversarial attacks on the deep learning based malware scanner, we implemented two categories of attack (i) additive noise and (ii) Fast Gradient Sign Method. In this Section we briefly introduce the additive noise techniques to construct adversarial examples. Gaussian noise is a noise that has a probability density function (pdf) equal to the normal distribution, where the noise is independent and identically distributed and has zero-mean and variance $N$. This noise is an Additive noise represented by a series of outputs $Y_i$ at discrete time index $i$. $Y_i$ is the sum of the input $X_i$ and noise, $Z_i$. The Gaussian output is given by

$$Y_i = X_i + Z_i,$$

(5)

Local variance noise is the zero-mean white Gaussian noise with an intensity-dependent variance. Speckle noise is a granular noise that exists in an image and affects the quality of the image. It is a dot of color on the image normally produced by random generation of spots over an image and an image of low contrast is obtained. Salt noise is the addition of some bright pixels all over the image. Pepper noise is the addition of some dark pixels all over the image. Salt and pepper noise is the addition of both white and dark pixels all over the image, where few pixels are replaced by 255 or 0.

4.4 Fast gradient sign attacks

The Fast Gradient Sign Method [38] is the effective used method to develop adversarial images. FGSM works on the input image. Predictions are made on the image using a trained deep neural network. Compute the loss of the prediction based on the true class label. Calculate the gradients of the loss to the input image. Compute the sign gradient of the image. Finally, construct the output adversarial image for further analysis.

For a particular input image, the method uses the loss gradient and the input image to create new images [39]. The new image thus obtained is called an adversarial image. Let the original input image be $x$, original input label be $y$, $J$ be the adversarial loss, $e$ be the multiplier and is considered to be small, and $\theta$ be the parameter of the model. This $adv_x$ is the adversarial image that can be viewed by the following equation.

$$adv_x = x + e \cdot sign(\nabla J(\theta, x, y))$$

(6)

The label corresponding to $adv_x$ and $x$ are different. The method evaluates the adversarial loss function with respect to every input. The resultant adversarial image is the image used for the attacks and it looks similar to human observation, but the neural network classifies it correctly.

5 Experimental results and discussions

5.1 System requirements

The minimum requirements to execute our model are based on Intel i3 7th generation processor, 8 GB DDR4 RAM, and 500 GB hard disk storage. Our model was created with Intel i7 8th generation processor, 16 GB DDR4 RAM, 8 GB VRAM, and 1 TB hard disk storage. The preprocessing unit is entirely coded in Python 3.5. We use TensorFlow and Keras as the backend to create a Dense network.

5.2 Dataset collection

The dataset used in the experiment constitutes of executables which are malware or benign. In this experiment, the malware executables were taken from two datasets, Malimg [6] and Microsoft Malware dataset (BIG2015) [33]. The benign executables were obtained from different sources [40] and analyzed with the VirusTotal service. This service consists of many anti-malware systems for establishing whether the given input file is malicious or not.

Malimg dataset contains 9339 malware samples from 25 families. In this dataset, each executable is represented in the form of a gray-scale image. Additionally we also experimented using the BIG2015 dataset contains 10860 labeled malware files belonging to 9 different families. Each file contains the hexadecimal representation of executables without the PE header. Raw hexadecimal values of malicious binaries are transformed into the images.

5.3 Evaluation metrics

The evaluation measures determined for the proposed methods are Precision, Recall or Test Positive Rate (TPR), F1 measure, Accuracy, False Positive rate (FPR) [6]. These metrics are calculated using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). $TP$ represents the number of malware samples correctly identified as malware. $TN$ represents the number of benign samples accurately identified as legitimate. $FP$ represents the number of benign samples misclassified as malware. $FN$ indicates the number of wrongly classified malware images. The evaluation metrics are defined using Eqs. (7–11).
\[
\text{Precision}(P) = \frac{TP}{TP + FP} \quad (7)
\]
\[
\text{Recall}(R) = \frac{TP}{TP + FN} \quad (8)
\]
\[
\text{F1-measure}(F) = 2 \frac{R \cdot P}{R + P} \quad (9)
\]
\[
\text{Accuracy}(A) = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)
\]
\[
\text{FPR} = \frac{FP}{FP + TN} \quad (11)
\]

- **Experiment 1** Performance evaluation of DenseNet for Malware Detection (Benign versus malware).
- **Experiment 2** Performance of DenseNet model on Evasion Attack with Adversarial samples generated using Additive Noise.
- **Experiment 3** Results of classification Model on a Poison Attack.
- **Experiment 4** Evaluation of DenseNet model on FGSM Attack.

### 5.4 Performance evaluation of DenseNet for malware detection (Benign versus malware)

The model created was based on binary classification between benign samples versus malware samples. The dataset is divided into two portions. The first portion consists of the training set which consists of 80% of the dataset and the other portion consists of 20% of the dataset called the test set. The training is done based on the train set created and the prediction is done based on the provided test set. As the ratio of the training set and test set are different we can evaluate the model performance and evaluate the parameters more accurately. The proposed model classifies whether a sample is a malware or benign based on the image obtained from it.

We noticed similar results with all experiments done on all datasets. With the image size of 256×256, with a dataset split of 80:20 on our model with 3 convolution and 3 pooling layers along with a fully connected NNs, obtained 96.79% accuracy and 96.79% F1-score for the BIG2015 dataset and 99.47% accuracy and 99.47% F1-measure for the malimg dataset as illustrated in Table 2. There was a 4.66% misclassification among the malware samples in the BIG2015 dataset and a 0.60% misclassification in the malimg dataset.

Fig. 2 gives the graphical representation of the three evaluation parameters, accuracy, F1 score, and precision for both the benchmark dataset.

### 5.5 Performance of DenseNet model on evasion attack with adversarial samples generated using additive noise

The second model created was based on binary classification, whether a sample is simply a benign sample or an obfuscated malware sample. From the malware dataset collected, the test samples were added with the six types of noises and the new test samples were generated.

Six obfuscation techniques were used for the adversarial attack. From obfuscated malware, the test samples are given to the DenseNet model. On testing, it was found that the model classified good with no significant change in malware misclassification. Table 3 indicates the adversarial attack on BIG2015 and Table 4 indicates the adversarial attack on malimg. *Indicates the number of misclassified samples, that is malware misclassified as benign. The noises included in the images were gaussian, localvar, speckle, salt noise, pepper noise, salt and pepper noise.
State-of-the-art data models can achieve better performance on image classification models. The classical models are not robust to the human visual system and have various forms of image distortions. These can be done by adding noise as mentioned in [34] and [35]. The distortions can be changed by increasing the noise levels in the noise models.

Table 5 gives a malware image of Adialer family as the first figure followed by (b) to (g) are the different types of noises mixed with the first image. These are variants of Adialer family injected with Gaussian noise, Local variance noise, Pepper noise, Salt noise, Speckle noise, and Salt and pepper noise. On observation, it is found that slight noise was added to the original image, Comparing with Fig (a), there is a difference in the patterns visually too in Fig. (b) to (g).

### 5.6 Results of classification Model on a poison attack

The third model created was based on classifying whether a sample is a benign sample or an obfuscated malware sample. We obfuscated a subset of the training malware dataset and used it to train a new DenseNet model. The obfuscated malware is extracted from six different obfuscated samples and the final model lowers the misclassification rate of the obfuscated malware samples. TN is the number of wrongly classified malware images. Here, it is obtained as 136 samples in BiG2015 and 17 in maling dataset was misclassified malware samples. TP signifies the number of correctly identified malware samples. In our case, we used 3129 malwaresamples in BiG2015 and 2802 malware samples in maling test dataset.

### 5.7 Evaluation of DenseNet model on FGSM attack

The fourth model is based on classifying whether a sample is a benign or malware sample after the white-box attack. There are many types of attacks, but here the focus is on the FGSM attack, which is a white box attack whose main aim is to guarantee misclassification. In a white box attack, an attacker has full access to the model being attacked. The working of the FGSM uses the gradient loss of the neural network to make the loss maximum by adjusting the input to result in an adversarial example. For a given input image, the method computes the loss gradient along with the input image to generate another image. It should be ensured that the loss is maximum. This new image is taken as the new obfuscated malware dataset and given to the model for classification as shown in Table 6.

In the fourth model, with an image size of $256 \times 256$, with a dataset split of 80:20, with 3 convolution and 3 max pool layers along with a fully connected layer and an epsilon value of 1.5, the proposed method obtained 95.50% accuracy, 95.49% F1-measure and a total time of 30.23 ms and for an epsilon value of 0.6, the accuracy obtained is 95.50%, 95.49% F1-measure and a total time of 26.81 ms for the BiG2015 dataset. For the Maling dataset, parameters obtained were 90.58% accuracy, 90.36% F1-measure, and a total time of 25.83 ms with the epsilon value of 1.5 and for the epsilon value of 0.6, the accuracy was 90.58%, 90.36% F1-measure and a total time of 24.35 ms. Here it is observed that there is a 1.5% decrease in accuracy in the

### Table 3 Adversarial attack on BiG2015

| Metrics       | Gaussian | Local var | Salt | Pepper | Speckle | Salt and pepper |
|---------------|----------|-----------|------|--------|---------|-----------------|
| Before attack | 4.6654 (152)* | 4.65 (152)* | 4.65 (152)* | 4.65 (152)* | 4.67 (153)* | 4.65 (152)* |
| % Misclassification | 4.65 (152)* | 4.65 (152)* | 4.65 (152)* | 4.65 (152)* | 4.67 (153)* | 4.65 (152)* |
| Avg Time (ms)    | 0.221     | 0.005     | 0.006 | 0.006  | 0.006   | 0.304           |

Number of malware samples in test set = 3258. Values in table $\alpha(\beta)^* \text{ indicate } \alpha \text{ as } \% \text{ of misclassification and } \beta \text{ as misclassified instances.}$

### Table 4 Adversarial attack on Maling dataset

| Metrics       | Gaussian | Local var | Salt | Pepper | Speckle | Salt and pepper |
|---------------|----------|-----------|------|--------|---------|-----------------|
| Before attack | 0.6067 (17)* | 0.606 (17)* | 0.6067 (17)* | 0.6030 (17)* | 0.6047 (17)* | 0.6060 (17)* |
| % Misclassification | 0.606 (17)* | 0.606 (17)* | 0.6067 (17)* | 0.6030 (17)* | 0.6047 (17)* | 0.6060 (17)* |
| Avg Time (ms) | 0.2809 | 0.0055     | 0.0059 | 0.0055 | 0.2897 | 0.2090 |

Number of malware samples in test set = 2802. Values in table $\alpha(\beta)^* \text{ indicate } \alpha \text{ as } \% \text{ of misclassification and } \beta \text{ as misclassified instances.}$
Table 5 Adialer family image file with six noises

(a) Adialer image  
(b) Gaussian noise  
(c) Local variance  
(d) Pepper noise  
(e) Salt noise  
(f) Speckle noise  
(g) Salt and pepper
BIG2015 dataset and a 9% decrease in accuracy for the Malimg dataset. It is also observed for lower epsilon values, less time is sufficient in a white-box attack on the model.

FGSM [36] operates on the gradients of the neural network to build an adversarial attack. FGSM is a particular type of algorithm, which updates weights along the direction of the loss gradient.

The model have obtained the results as shown in Table 6, how FGSM is used to fool the trained neural network model arriving at incorrect predictions.

Fig. 3 shows the comparison of the three attacks on BIG2015 dataset. The misclassification is in the range of 4 to 8%, which is a major misclassification. If a very large dataset is considered, the percentage of malware samples classified as benign is comparatively less, which accounts for the performance of the DenseNet model. Fig. 4 shows the comparison of the three attacks on malimg dataset. Evasion attacks are common ones in the adversarial learning models. In the evasion situation, the attack happens during the test phase. In Poisson attacks, the changes are in the training phase, where the model infer incorrectly the malware as benign. For evasion and poisson attacks, the misclassification is negligible in the proposed model. But for FGSM, it identifies 21.8% malware samples as benign which is considerably high.

Table 7 indicates different noise levels added in benign and adversarial samples and the measures were calculated in BIG2015 dataset. The samples are randomly selected from the training set. Later it is relabelled as benign. But smart attackers will take the sample and submit it to the classifier. It is found that the classifier performs well on the noise-added datasets. The FPR without attack is obtained as 2% but after attack is 1.645%. The reason for the marginal decrease in FPR is that, the malicious samples statistical distribution becomes benign and it brings down the samples that are classified as malware.

Table 8 shows different noise levels induced malware folders in malimg dataset. From the table, it is observed that the classifier, misclassifies fewer samples when the noise added increases. This is the countermeasure where the performance measures are better than those compared with adversarial samples. The FPR obtained is 2.210% which is slightly higher than the FPR without attack

Table 9 shows different noise levels indicates different noise levels added in benign and adversarial samples and the measures were calculated in BIG2015 dataset. The samples are randomly selected from the training set. Later it is relabelled as benign. This is performed on the malimg dataset too. The FPR without attack for malimg is obtained as 0.514% and after attack is 0.565%.

| Table 6 Performance evaluation using FGSM |
|-------------------------------------------|
| Parameters                          | BIG2015 | Malimg |
|---------------------------------------|---------|--------|
| Epsilon                              | 0.6     | 1.5    | 0.6   | 1.5   |
| Accuracy (%)                         | 95.51   | 95.50  | 90.58 | 90.58 |
| F1 Measure (%)                       | 95.50   | 95.49  | 90.36 | 90.36 |
| Misclassification (%)                | 7.51    | 7.51   | 21.80 | 21.80 |
| Time (ms)                            | 29.35   | 30.23  | 24.35 | 25.83 |

Fig. 3  Performance of DenseNet on BIG2015 dataset

Fig. 4  Performance of DenseNet on Malimg dataset
Table 10 induced noise malware folders in malimg dataset. For both the dataset, the samples chosen are randomly taken and 10%, 20%, and 30% of the noise added and either put in benign or malware families and then given to the classifier. The FPR for malimg on retraining data is obtained as 0.283% for 10%, and slightly higher for the remaining two cases. From the table, it is observed that the classifier, misclassifies less samples when the noise added increases compared to Table 8.

6 Conclusion and future work

In this work, a comprehensive view of the images derived from executables of different programs is analyzed and how appropriate the malware is identified is also discussed. This is achieved by assessing the performance of the DenseNet model. The study showed the finest results with grayscale images of size $256 \times 256$ of the MALIMG dataset even after obfuscation. The experiments carried out on two
benchmark datasets resulted in an F1-score of 96.81% for BIG2015 and 99.45% for malimg dataset. Carrying out an adversarial attack on the above model resulted in the misclassification of certain malware samples.

As part of continuing research on this topic, we have also investigated the Fast Gradient Sign method. These methods can be used to create perturbations in the test set to bring down the accuracy further. The FGSM results on the dataset show that the accuracy dropped from the 90% by about 2%. Further, for the models to correctly classify in spite of the perturbations, the models were retrained using the subset of the train malware dataset. This method decreased the misclassification rate of malware samples. Our study specifically determined the F1 measure, Accuracy, FPR and prediction time. Since DenseNet alleviates the vanishing-gradient problem, and the FGSM model exploits the gradient of the model, we can observe a higher performance in the detection of DenseNet model.

In future, we plan to extend dataset by including more samples from alternate malware repositories. Also, we aim to perform the malware family classification using DenseNet model. Further, we also plan to study if the machine learning models have capability to classify malicious executables obfuscated using diverse obfuscation transformations. Finally, we plan to construct deep learning based malware classification system using visualization techniques which is agnostic to file types, as research in detection of malware on Android devices [37] are gaining large attention amongst the researchers.

### Funding
The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

### Declarations

#### Competing Interests
The authors have no relevant financial or non-financial interests to disclose.

### References

1. Ahmad, Mariwan. (2020). Malware in Computer Systems: Problems and Solutions. IJID (International Journal on Informatics for Development), https://doi.org/10.14421/ijid.2020.09101
2. https://www.sophos.com/en-us/mediablog/technical-papers/sophos-2021-threat-report.pdf
3. https://www.mcafee.com/enterprise/en-us/lp/covid-19-dashboard.html
4. Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z., Mao, L.: MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. Comput. Secur. 83, 208–233 (2019)
5. Maniriho, P., Mahmood, A.N., Chowdhury, M.J.: A study on malicious software behavior analysis and detection techniques: taxonomy, current trends, and challenges. Future Gener. Comput. Syst. 130, 1–18 (2022)
6. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S.: Malware images: visualization and automatic classification. In: Proceedings of 8th International Symposium, Visualization for Cyber Security, VizSec (2011)
7. Wang, C., Mark Liao, H., Wu, Y., Chen, P., Hsieh, J., Yeh, I.: CSPNet: A New Backbone that can Enhance Learning Capability of CNN, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1571–1580 (2020)
8. Katzenbeisser, S., Kinder, J., Veith, H.: Malware Detection: Encyclopedia of Cryptography and Security, pp. 752–755. Springer, Boston (2011)
9. Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and API calls. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300–305 (2013)

### Table 10 Performance evaluation of Malimg on adversarial retraining

| % of samples | Confusion Matrix | Results (%) | Time (ms) |
|--------------|------------------|-------------|-----------|
| 10 % of retraining samples | 3880 11 2782 | Accuracy = 99.54 Loss = 2.72 Precision = 99.54 Recall = 99.54 F-measure = 99.54 FPR = 0.283 | Total = 29.626 Avg = 0.004 |
| 20 % of retraining samples | 3878 13 2779 | Accuracy = 99.46 Loss = 2.83 Precision = 99.46 Recall = 99.46 F-measure = 99.46 FPR = 0.334 | Total = 32.512 Avg = 0.005 |
| 30 % of retraining samples | 3875 16 2780 | Accuracy = 99.44 Loss = 2.90 Precision = 99.44 Recall = 99.44 F-measure = 99.44 FPR = 0.411 | Total = 32.868 Avg = 0.005 |
10. Xu, Z., Ray, S., Subramanyan, P., Malik, S.: Malware detection using machine learning based analysis of virtual memory access patterns. In: Design, Automation & Test in Europe Conference & Exhibition, pp. 169–174 (2017)
11. Baptista, I., Shiaeles, S., Kolokotronis, N.: A novel malware detection system based on machine learning and binary visualization. In: IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6 (2019)
12. Ünver, H.M., Bakour, K.: Android malware detection based on image-based features and machine learning techniques. SN Appl. Sci. 2, 1299 (2020)
13. Jian, Y., Kuang, H., Ren, C., Ma, Z., Wang, H.: A novel framework for image-based malware detection with a deep neural network. Comput. Secur. 109, 102400 (2021)
14. Yuxin, D., Siyi, Z.: Malware detection based on deep learning algorithm. Neural Comput. Appl. 3(2), 461–472 (2017)
15. Hardy, W., Chen, L., Hou, S., Ye, Y., Li, X.: DL4MD: a deep learning framework for intelligent malware detection. In: Proceedings of the International Conference on Data Science (2016)
16. Schranko de Oliveira, Angelo; Sassi, Renato José (2020): Chi-squared KL divergence for visualizing adversarial examples. International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 344–347 (2017). https://doi.org/10.1109/ICCCA.2017.7951936
17. Karimi, A., Moattar, M.H.: Android ransomware detection using reduced opcode sequence and image similarity. In: 2017 7th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 229–234 (2017). https://doi.org/10.1109/ICCKE.2017.8167881
18. Gabel, T., Godehardt, E.: Top-down induction of similarity measures using similarity clouds. In: Hüllermeier, E., Minor, M. (eds.) Case-Based Reasoning and Development, pp. 149–164. Springer, Cham (2015)
19. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Prog. Artif. Intell. 9(2), 129–143 (2020)
20. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Explaining vulnerabilities of deep learning to adversarial malware binaries. In: Proceedings of the Third Italian Conference on CyberSecurity (ITASEC) (2019)
21. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Functionality-preserving black-box optimization of adversarial windows malware. IEEE Trans. Inf. Forensics Secur. 16, 3469–78 (2021)
22. Sun, Li, Taylor, D., Wu, Q., Mahoney, M.W.: Noise-respose analysis of deep neural networks quantifies robustness and fingerprints structural malware. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM) 2021, pp. 100–108. Society for Industrial and Applied Mathematics (2021)
23. Demetrio, L., Coull, S.E., Biggio, B., Lagorio, G., Armando, A., Roli, F.: Adversarial examples: a survey and experimental evaluation of practical attacks on machine learning for windows malware detection. ArXiv abs/2008.07125 (2020)
24. Machado, Gabriel R., Eugenio Silva and R. Goldschmidt. Adversarial Machine Learning in Image Classification: A Survey Towards the Defender’s Perspective, ArXiv abs/2009.03728, 2020
25. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., Keshet, J.: Deceiving end-to-end deep learning malware detectors using adversarial examples (2018). arXiv:1802.04528
26. Ford, N., Gilmer, J., Carlini, N., Cubuk, D.: Adversarial Examples are a Natural Consequence of Test Error in Noise, arXiv:1901.10513 (2019)
27. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014)
28. Keerthi, V., Venkatesh, V.: Reshaping Security: A Computability Perspective, ArXiv abs/2009.03728, 2020
29. Carlini, N., Papernot, N.: Towards the Adversary, ArXiv abs/2009.03728, 2020
30. Schranko de Oliveira, Angelo; Sassi, Renato José (2020): Chi-squared KL divergence for visualizing adversarial examples. International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 344–347 (2017). https://doi.org/10.1109/ICCCA.2017.7951936
31. Karimi, A., Moattar, M.H.: Android ransomware detection using reduced opcode sequence and image similarity. In: 2017 7th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 229–234 (2017). https://doi.org/10.1109/ICCKE.2017.8167881
32. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Explaining vulnerabilities of deep learning to adversarial malware binaries. In: Proceedings of the Third Italian Conference on CyberSecurity (ITASEC) (2019)
33. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Functionality-preserving black-box optimization of adversarial windows malware. IEEE Trans. Inf. Forensics Secur. 16, 3469–78 (2021)
34. Sun, Li, Taylor, D., Wu, Q., Mahoney, M.W.: Noise-respose analysis of deep neural networks quantifies robustness and fingerprints structural malware. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM) 2021, pp. 100–108. Society for Industrial and Applied Mathematics (2021)
35. Demetrio, L., Coull, S.E., Biggio, B., Lagorio, G., Armando, A., Roli, F.: Adversarial examples: a survey and experimental evaluation of practical attacks on machine learning for windows malware detection. ArXiv abs/2008.07125 (2020)
36. Machado, Gabriel R., Eugenio Silva and R. Goldschmidt. Adversarial Machine Learning in Image Classification: A Survey Towards the Defender’s Perspective, ArXiv abs/2009.03728, 2020
37. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., Keshet, J.: Deceiving end-to-end deep learning malware detectors using adversarial examples (2018). arXiv:1802.04528
38. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014)
39. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., Keshet, J.: Deceiving end-to-end deep learning malware detectors using adversarial examples (2018). arXiv:1802.04528
40. https://en.softonic.com/s/benign-images
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
V. Anandhi is currently the Head of the Electronics and Communication department at the SCMS School of Engineering and Technology, India. She is also research scholar in Image Security at APJ Abdul Kalam Technological University, Kerala. She completed her B. E Electronics and Communication Engineering from Tamil Nadu College of Engineering, Coimbatore and M.E Applied Electronics from Kongu Engineering College, Erode. Her research interests are in malware analysis, image security.
P. Vinod is presently Professor at the Department of Computer Applications at Cochin University of Science & Technology, Cochin, Kerala, India. Prior to this he was Professor in Department of Computer Science & Engg. at SCMS School of Engineering & Technology, Kerala, India. He holds a Ph.D from Department of Computer Engineering, Malaviya National Institute of Technology, Jaipur, India. He was Post Doc at Department of Mathematics, University of Padua, Italy and also a Post-Doctoral researcher at Malaviya National Institute of Technology, Jaipur, India. He has more than 70 research articles published in peer reviewed Journals and International Conferences. He is a reviewer of a number of security journals, and has also served as programme committee member in the International Conferences related to Computer and Information security. His current research is involved in the development of malware scanners for mobile applications using machine learning techniques. Vinod’s area of interest is Adversarial Machine Learning, Malware Analysis, Context aware privacy preserving Data Mining, and Natural Language Processing.

Varun G. Menon is currently Professor and Head of the Department of Computer Science Engineering at SCMS School of Engineering and Technology, India. He is a Distinguished Speaker of the Association of Computing Machinery (ACM) and a Senior Member of IEEE. He has completed his Ph.D. in Computer Science and Engineering and holds an M. Tech degree in Computer and Communication with University First Rank. He also holds an M.Sc. in Applied Psychology, an MBA in Human Resource Management and a Diploma in Training and Development. His research interests include Sensor Technologies, Internet of Things, Green IoT, Wireless Communication, Fog Computing and Networking.

Korankotte Manoj Aditya is a Bachelor student at the Department of Computer Science Engineering at SCMS School of Engineering and Technology, India. His research interests are in Machine Learning, Malware Analysis.