MALWARE DETECTION AND CLASSIFICATION USING GENERATIVE ADVERSARIAL NETWORK

Krishna Kumar[1], Hardwari Lal Mandoria[1], Rajeev Singh[1], Shri Prakash Dwivedi[1], Paras[2]

[1]Department of Information Technology, College of Technology, G. B. Pant University of Agriculture and Technology Pantnagar, Uttarakhand, India
[2]Department of Electronics and Communication Engineering, College of Technology, G. B. Pant University of Agriculture and Technology Pantnagar, Uttarakhand, India

ABSTRACT

The Generative Adversarial Networks (GANs) are playing a crucial role in deep-learning-based malware classification to overcome the dataset imbalance and unseen malware. The Generative AI is preferably used in many applications, such as improving image resolution and generating audio, video, and text. The cybercriminals are also using the Generative AI for generating the malware and deepfake videos to harm the targeted person or device. By generating the synthetic data, it makes the deep learning model more robust to detect such types of unseen and adversarial attacks. This work utilizes GANs for generating adversarial malware samples to train a classification and detection model, improving the model's ability to identify sophisticated malware variants. The performance of the proposed Conditional Generative Adversarial Network (CGAN) model is evaluated on a multiclass malware grayscale image dataset and a binary class malware RGB image dataset. The performance of proposed model is compared with current state-of-the-art. Results indicate a significant improvement in classification accuracy and a reduction in training time and false positives, showcasing GAN's potential in the dynamic cybersecurity landscape.

KEYWORDS

Malware detection, Generative Adversarial Networks, deep learning, classification, adversarial learning, cybersecurity.

1. INTRODUCTION

Malware remains a significant challenge to global cybersecurity, with attackers continuously refining their methods, making it difficult for traditional detection systems to identify zero-day exploits and obfuscated malware[1]. The rapid expansion of cloud computing, IoT, sensor devices, and Industry 4.0 has significantly increased the risk of cyberattacks[2]. This digital growth, along with the rise of big data analytics for business decisions, highlights the growing dependency on computing resources[3], [4]. The use of generative AI technologies, such as ChatGPT, poses new challenges for conventional cybersecurity techniques, as cybercriminals exploit these tools to perpetrate fraud, creating fake images, audio, and videos[5]. Consequently, researchers are focusing on enhancing security measures, particularly against zero-day, ransomware, and APT attacks[6]. Generative Adversarial Networks (GANs) turn out to be very useful to overcome the unbalanced dataset problem and improves the performance of model is terms of malware detection accuracy. Deep learning advancements, particularly in Generative Adversarial Networks (GANs), offer a promising new direction by enhancing the detection of emerging threats. This study explores how GANs can be effectively applied to malware detection

DOI: 10.5121/ijcsit.2024.16508

and classification. GANs not only identify malware but also learn from adversarial samples to build a more resilient detection model.

1.1. Generative Adversarial Network (Gan)

Generative Adversarial Network is a deep learning network model proposed by Goodfellow [7]. It is based on the minimax two-player game that consists of two basic components: the generative model and the discriminative model D. The generator produces synthetic samples $p(z)$from a latent vector $p_g$over the data x, while the discriminator differentiates between fake $p_z(z)$and real sample $p_{data(x)}$. Both the generator and discriminator are iteratively fine-tuned through training to create a highly optimized GAN-based detection system (Figure 1).

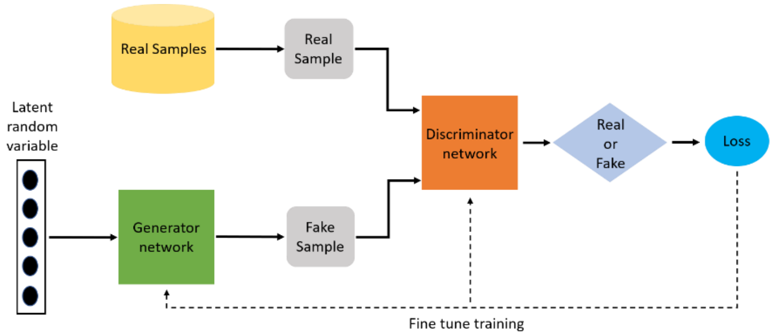
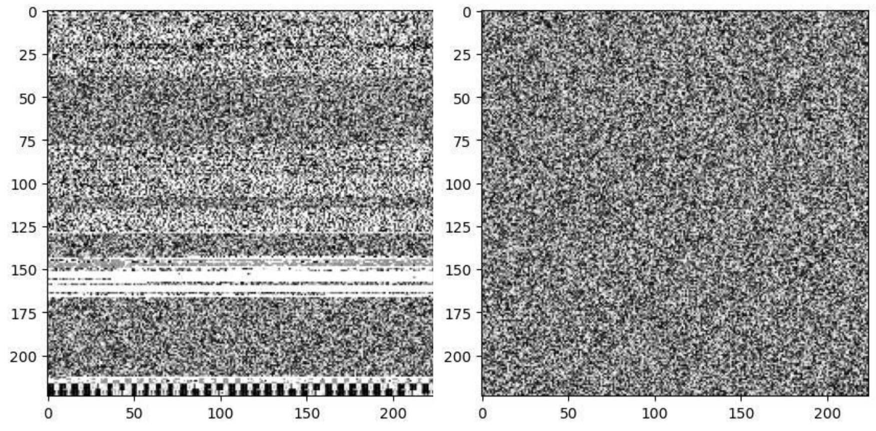


Figure 1. Basic GAN Architecture [8]

The $G$and Dboth are trained simultaneously; we update parameters for $G$to minimize $lo(1 - D(G(z))$and update parameters for D to minimize $logD(x)$. Both models are trained up to when no further improvements can be done because of p_g = p_{data}. The $D$and G are using the value function V(G,D) as [7]:

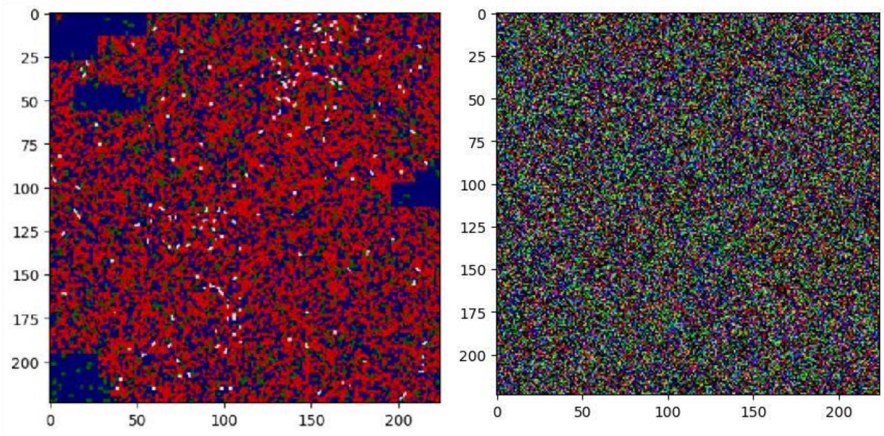
$$min_G max_D(D, G) = \mathbb{E}_{x\sim pdata(x)}[\log D(x)] + \mathbb{E}_{z\sim pz(x)}[\log(1 - G(z)] \qquad (1)$$

The generator learns to generates fake image and discriminator trained to identify real and generated images. The inclusion of adversarial malware samples in the DL model makes it more robust for novel and unseen malware. The generator learns to generate the fake malware samples called the unsupervised learning. On the other hand, the discriminator identifies the malware images as real or fake using the supervised learning because we also provide the label information along with the image samples from both sides, the generator and the real dataset. The generated image with respect to the real image for grayscale malware image dataset and RGB image dataset are displayed in Figure 2 and Figure 3.



Real Grayscale Malware Sample **Generated Fake Malware Sample**

Figure 2. Real and generated fake grayscale malware sample



Real RGB Malware Sample **Generated Fake Malware Sample**

Figure 3. Real and generated fake RGB malware sample

1.2. Conditional Generative Adversarial Network (Cgan)

The Conditional Generative Adversarial Network (CGAN) is the extension of the model for condition-based learning where we provide the data x with respective label information y for the both generator$G$ and discriminator D. For this, the two-player minimax game equation is described as [9]:

$$min_G max_D(D, G) = \mathbb{E}_{x\sim pdata(x)}[\log D(x|y)] + \mathbb{E}_{z\sim pz(x)}[\log(1 - G(z|y)] \quad (2)$$

There is many cybersecurity threats and challenges that can only be reduced by employing the unsupervised or semi-supervised deep learning techniques [6]. The novel and unseen malwares cannot be effectively detected by the traditional intrusion detection system[10]. The Discriminator plays a crucial role to discriminate the real and fake samples. It also helps to learn model to detect the unseen malware samples.

2. RECENTRESEARCHWORKS

Several studies have explored deep learning approaches for malware detection, such as CNNs and RNNs[11]. However, GANs have seen limited application in this domain. Prior works utilizing GANs primarily focus on generating adversarial samples for evasion attacks or data augmentation. There is a strong requirement to work on designing intrusion detection system that are able to detect adversarial attacks. The researchers are working on a GAN-based technique for malware classification, as mentioned below.

Nagarjun and Stamp [12] introduced an Auxiliary-Classifier GAN (AC-GAN) model for malware classification, which was trained and tested on two malware image datasets: Malimg and MalExe. They experimented with three different image sizes—32×32, 64×64, and 128×128. The AC-GAN model achieved its highest classification accuracy of 95% on these datasets, demonstrating its effectiveness in identifying malware based on image-based data representation. Chui et al. [13] introduced a lightweight GAN model to address imbalanced malware classification issue. They utilized two efficient architectures—ShuffleNetV2 and MobileNetV3, both integrated with a GAN framework for malware detection. Their approach achieved classification accuracies of 94.2% and 95%, respectively, showcasing the potential of lightweight deep learning models in handling malware classification tasks while maintaining performance and efficiency.

Won et al. [14] introduced four GAN models—DCGAN, LSGAN, WGAN-GP, and E-GAN— for classifying zero-day malware attacks. Their study used the Malimg dataset, focusing on 8 malware classes with shared family names, such as Allaple.A and Allaple.L, comprising 10,868 samples. Among the models, E-GAN achieved the highest classification accuracy at 96.35%, demonstrating its effectiveness in handling malware classification within these specific categories.

Reilly et al. [15] introduced robust GAN models for malware classification using byteplot and space-filling curve conversion techniques. The vanilla byteplot model achieved a classification accuracy of 95.76%, while the vanilla Z-order model reached 93.12%. Their study highlighted that incorporating adversarial images during training could significantly enhance the robustness of the classification models, making them more resilient against evolving malware threats.

Lu and Li [16] proposed a GAN-based approach to enhance deep learning models for malware classification. By incorporating GAN-generated synthetic malware samples, they significantly improved the performance of a deep residual network. Initially, the residual network classified malware images with 84% accuracy. However, after augmenting the training set with GANgenerated samples, the test classification accuracy increased by 6%, reaching 90%, demonstrating the effectiveness of using GANs for enhancing deep learning-based malware detection systems.

These studies collectively highlight the increasing focus on applying GANs for malware classification, emphasizing their capacity to address challenges like limited labelled data and generating diverse malware samples. The research underscores GANs as a promising approach in improving cybersecurity, particularly by enhancing malware analysis and classification performance. By handling imbalanced datasets and producing adversarial samples, GANs provide a robust mechanism for building more effective and resilient malware detection systems.

3. PROPOSED METHODOLOGY

section describes the benchmark datasets for the performance evaluation and GAN model details.

3.1. Dataset Description

There are two separate datasets used separately for malware classification and malware detection. The **Dataset-1 (**Malimg [17]) that contains 9,339 grayscale malware image samples of 25 distinct malware classes. The image samples are varying shape across the classes that are resized to a standard size of 224×224 pixels to ensure consistency across the model training and testing processes. The distribution of malware samples across 25 classes is displayed in Figure 4.

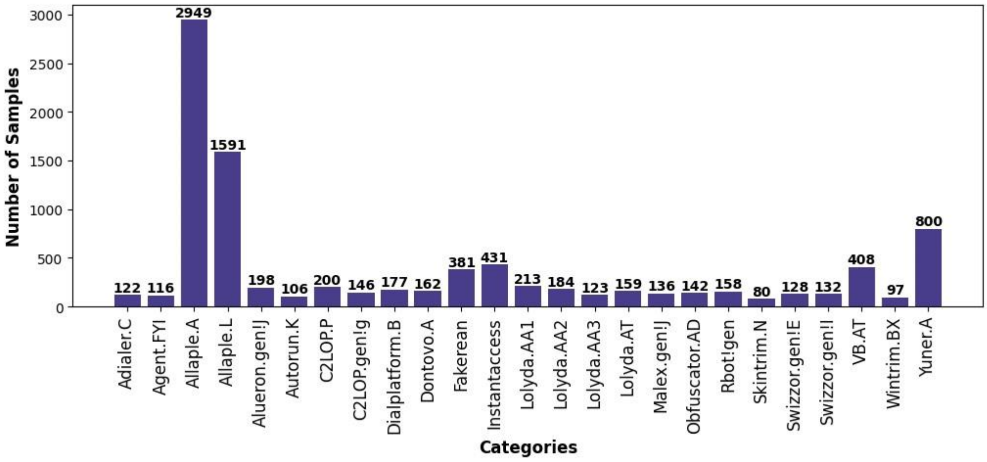


Figure 4. Dataset-1 Sample Distribution

Dataset-2 [18] contains the total 48,240 malware image samples and source code files out of which only 24,109 binary class malware and normal images are used for binary classification model. The image dataset contains 11,919 malicious images samples and 12,190 normal image samples. These samples are employed to test the model's ability to detect anomalies and accurately differentiate between malware and benign data.

3.2. Dataset Pre-Processing

The Image samples from the grayscale image dataset are resized into the shape of 224×224×1 as the required input image dimensions for the malware classification model. The dataset is divided into the 80:20 ratio for the training and testing purpose. The input image pixels are normalized in the range of [-1, 1] stored in a *numpy* array. Dataset-2 consists of binary class and having the 3 channel images. These image samples are normalized and reshaped into the shape of 224×224×3 required input image dimension for the malware detection model.

3.3. Generative Adversarial Network (Gan) Model

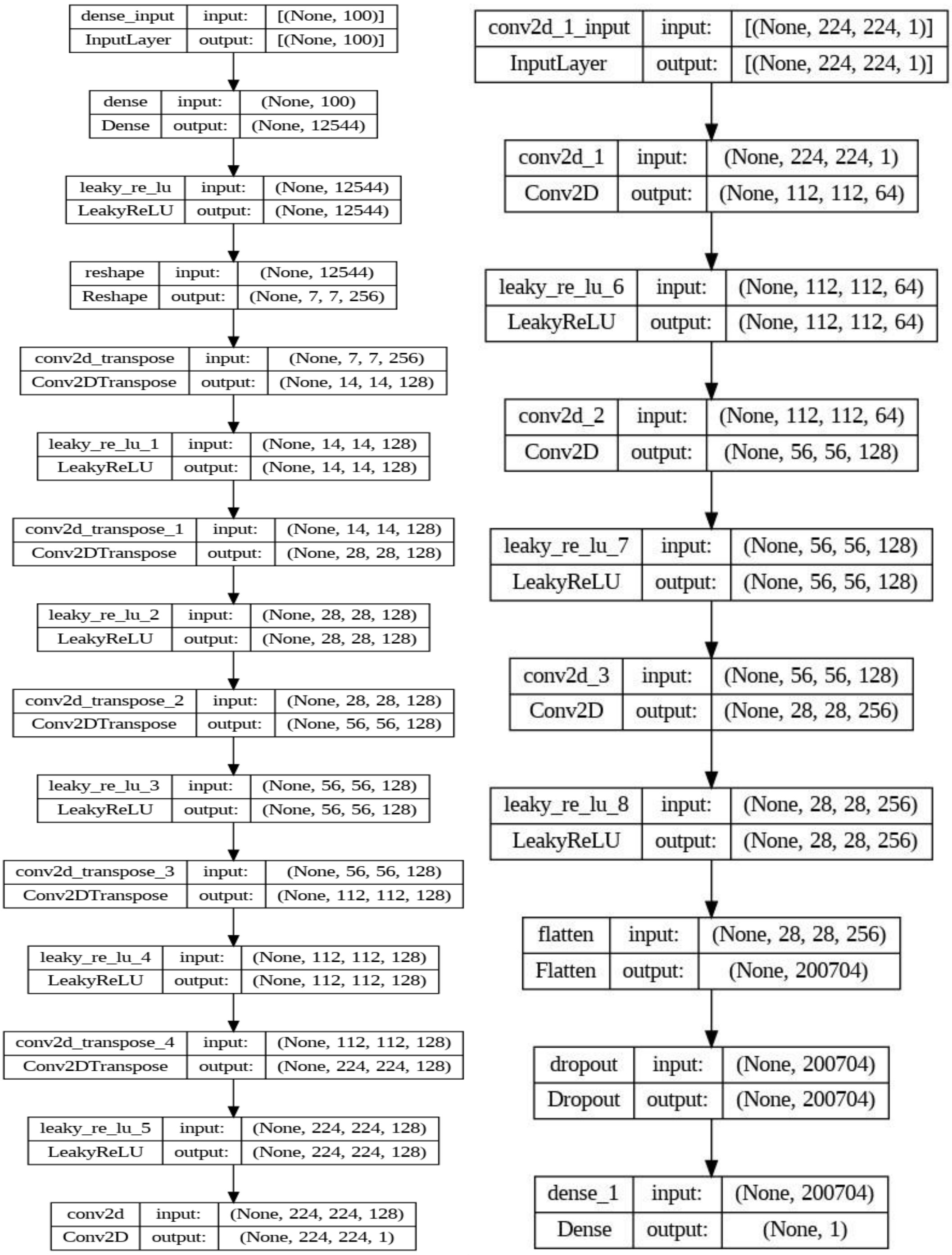
The GAN model consists of two DL model generator and discriminator.

3.3.1. Building the Generator Model

The generator model generates similar input data as the real data sample from the dataset. Here, we have a grayscale image of a malware dataset of size 224×224 pixels. For this, the generator needs to generate the same size of image from the given latent dimension of length 100. It starts with the fully connected layer of a 7×7×256 input vector with an input dimension value of 100. This layer generates the 256 images of size 7×7. Now the next layers use the *Convolution2D Transpose* layer and *LeakyRelu* layer to up sample the image size as 14×14, 28×28, 56×56, 112×112, and final layer uses the activation function *tanh* to get fake image of size 224×224. The model architecture for generator G is given in Figure 5(a).

3.3.2. Building the Discriminator Model

The discriminator model takes the 224×224 input image and discriminate the image as real or fake. In this experimental work, the discriminator model contains the three sets of layers: the *2D Convolution* layer and the *LeakyRelu* layer. The final set of layers contains a *flatten* layer, followed by a *dropout* layer with a dropout value of 0.4, and a final dense layer of 1 output value with a *sigmoid* activation function. The model architecture for discriminator D is shown in Figure 5(b).


dense_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
dense | input: | (None, 100)
Dense | output: | (None, 12544)
leaky_re_lu | input: | (None, 12544)
LeakyReLU | output: | (None, 12544)
reshape | input: | (None, 12544)
Reshape | output: | (None, 7, 7, 256)
conv2d_transpose | input: | (None, 7, 7, 256)
Conv2DTranspose | output: | (None, 14, 14, 128)
leaky_re_lu_1 | input: | (None, 14, 14, 128)
LeakyReLU | output: | (None, 14, 14, 128)
conv2d_transpose_1 | input: | (None, 14, 14, 128)
Conv2DTranspose | output: | (None, 28, 28, 128)
leaky_re_lu_2 | input: | (None, 28, 28, 128)
LeakyReLU | output: | (None, 28, 28, 128)
conv2d_transpose_2 | input: | (None, 28, 28, 128)
Conv2DTranspose | output: | (None, 56, 56, 128)
leaky_re_lu_3 | input: | (None, 56, 56, 128)
LeakyReLU | output: | (None, 56, 56, 128)
conv2d_transpose_3 | input: | (None, 56, 56, 128)
Conv2DTranspose | output: | (None, 112, 112, 128)
leaky_re_lu_4 | input: | (None, 112, 112, 128)
LeakyReLU | output: | (None, 112, 112, 128)
conv2d_transpose_4 | input: | (None, 112, 112, 128)
Conv2DTranspose | output: | (None, 224, 224, 128)
leaky_re_lu_5 | input: | (None, 224, 224, 128)
LeakyReLU | output: | (None, 224, 224, 128)
conv2d | input: | (None, 224, 224, 128)
Conv2D | output: | (None, 224, 224, 1)
conv2d_1_input | input: | [(None, 224, 224, 1)]
InputLayer | output: | [(None, 224, 224, 1)]
conv2d_1 | input: | (None, 224, 224, 1)
Conv2D | output: | (None, 112, 112, 64)
leaky_re_lu_6 | input: | (None, 112, 112, 64)
LeakyReLU | output: | (None, 112, 112, 64)
conv2d_2 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 56, 56, 128)
leaky_re_lu_7 | input: | (None, 56, 56, 128)
LeakyReLU | output: | (None, 56, 56, 128)
conv2d_3 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 28, 28, 256)
leaky_re_lu_8 | input: | (None, 28, 28, 256)
LeakyReLU | output: | (None, 28, 28, 256)
flatten | input: | (None, 28, 28, 256)
Flatten | output: | (None, 200704)
dropout | input: | (None, 200704)
Dropout | output: | (None, 200704)
dense_1 | input: | (None, 200704)
Dense | output: | (None, 1)

(a) (b)

Figure 5. (a) Generator model architecture, and (b) Discriminator model Architecture

3.3.3. Defining GAN Model

A GAN model is created by combining the two separate models, generator and discriminator, in a sequential model using the *Keras* library (Figure 6).

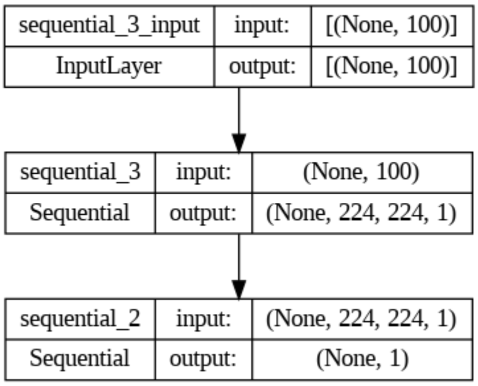


Figure 6. GAN model Architecture

The GAN model has two sequential components, first the generator that takes input dimension of size 100 to generate the 224×224 size fake image and the discriminator takes the input image of size 224×224-pixel value to generate binary class output 0 or 1. Here, 0 represents the fake and 1 represents the real image. Then the GAN model is trained on the set of real and fake images to adjust the model parameters up to the condition where generator is competitive for discriminator.

3.4. CGAN Model for Malware Classification

The Conditional Generative Adversarial Network (CGAN) [9] model, extends the capability of discriminator to classify the number of classes of a given dataset. For this a dense layer is added as a final layer with the *softmax* activation function for multiclass malware classification. The model created for multiclass malware classification is named as ‘CGAN Model1’ (Figure 7(a)).

3.5. CGAN Model for Malware Detection

A CGAN model named ‘CGAN Model2’ is created for malware detection task. The GAN model is extended by adding a *flatten* layer and after this one *dense* layer with activation function *relu* and a final output layer for binary class classification with *sigmoid* activation function (Figure 7(b)).

3.6. Performance Evaluation Metrics

The performance metrics used to evaluate the CGAN model is described following.

3.6.1. Accuracy

Accuracy measures the overall correctness of the model by calculating the proportion of correctly predicted instances including both positive and negative out of all predictions.

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

3.6.2. Precision

Precision is the proportion of predicted true positive samples and total samples predicted as positive by the model also called positive predicted value.

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

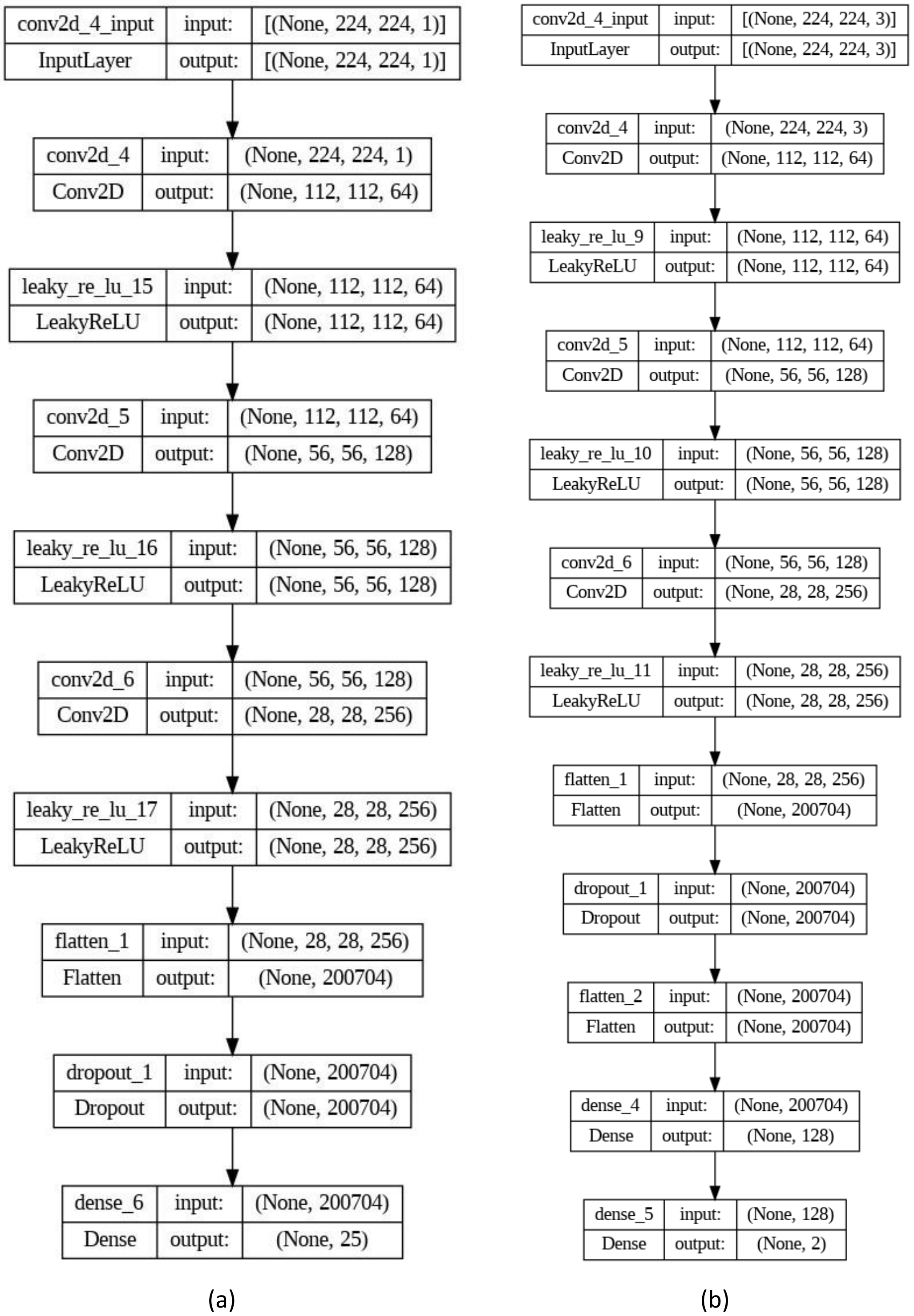

conv2d_4_input | input: | [(None, 224, 224, 1)]
InputLayer | output: | [(None, 224, 224, 1)]
conv2d_4 | input: | (None, 224, 224, 1)
Conv2D | output: | (None, 112, 112, 64)
leaky_re_lu_15 | input: | (None, 112, 112, 64)
LeakyReLU | output: | (None, 112, 112, 64)
conv2d_5 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 56, 56, 128)
leaky_re_lu_16 | input: | (None, 56, 56, 128)
LeakyReLU | output: | (None, 56, 56, 128)
conv2d_6 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 28, 28, 256)
leaky_re_lu_17 | input: | (None, 28, 28, 256)
LeakyReLU | output: | (None, 28, 28, 256)
flatten_1 | input: | (None, 28, 28, 256)
Flatten | output: | (None, 200704)
dropout_1 | input: | (None, 200704)
Dropout | output: | (None, 200704)
dense_6 | input: | (None, 200704)
Dense | output: | (None, 25)
(a)
conv2d_4_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
conv2d_4 | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 112, 112, 64)
leaky_re_lu_9 | input: | (None, 112, 112, 64)
LeakyReLU | output: | (None, 112, 112, 64)
conv2d_5 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 56, 56, 128)
leaky_re_lu_10 | input: | (None, 56, 56, 128)
LeakyReLU | output: | (None, 56, 56, 128)
conv2d_6 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 28, 28, 256)
leaky_re_lu_11 | input: | (None, 28, 28, 256)
LeakyReLU | output: | (None, 28, 28, 256)
flatten_1 | input: | (None, 28, 28, 256)
Flatten | output: | (None, 200704)
dropout_1 | input: | (None, 200704)
Dropout | output: | (None, 200704)
flatten_2 | input: | (None, 200704)
Flatten | output: | (None, 200704)
dense_4 | input: | (None, 200704)
Dense | output: | (None, 128)
dense_5 | input: | (None, 128)
Dense | output: | (None, 2)
(b)

Figure 7. (a) Classification Model CGAN_1, and (b) Detection Model CGAN_2

3.6.3. Recall

Recall is the proportion of correctly predictive positive samples over the total positive samples and also called the true positive rate.

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

3.6.4. F1-score

The mathematical formula to calculate F1-score is given below.

$$F1\ score = \frac{2 * (recall * precision)}{recall + precision}$$

3.6.5. Receiver Operating Characteristic (ROC-AUC) Curve

The Receiver Operating Characteristic (ROC-AUC) curve shows the probability of model for classifying the true positive and false positive sample for the threshold values of range 0 and 1.

3.6.6. Confusion Matrix

The confusion matrix provides the model's predicted value for each sample. It makes easier to understand the performance of classification and detection model in respect to identify true positive, false positive, true negative and false negative samples of each class.

3.7. Hardware and Software Requirement

The experimental setup consists of the computational resources and software tools and libraries to create, train and evaluate the GAN model. The Google Collaboratory, is a cloud-based platform that provides required computational resources as per the requirement. This experimental work is performed by using an A100 GPU, equipped with 83.5 GB of system RAM, 40 GB of GPU RAM, and 201.3 GB of disk storage. This virtual computing platform has installed Python 3(version 3.10.12), TensorFlow (version 2.15.0), and the NumPy library.

4. EXPERIMENTAL ANALYSIS

The GAN-based detection model was evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC curve.

4.1.Malware Classification

After training the GAN model or the real and fake malware image samples, the model is modified for multiclass classification. The models are compiled with two different optimizers that are *Adam* and *Stochastic Gradient Descant (SGD)* named as 'CGAN Model1' and 'CGAN Model2' respectively. The CGAN models as multiclass classifier is trained on the training dataset of grayscale image dataset (dataset-1) up to the 50 epochs. The training validation accuracy and training validation loss curve for the classification models shown in Figure 8.

The performance of both classification models is evaluated on the test dataset. The test accuracy of CGAN Model1 and CGAN Model2 is achieved as 96.96% and 95.67% respectively. The performance matrix for CGAN Model1 is illustrated in Table 1.

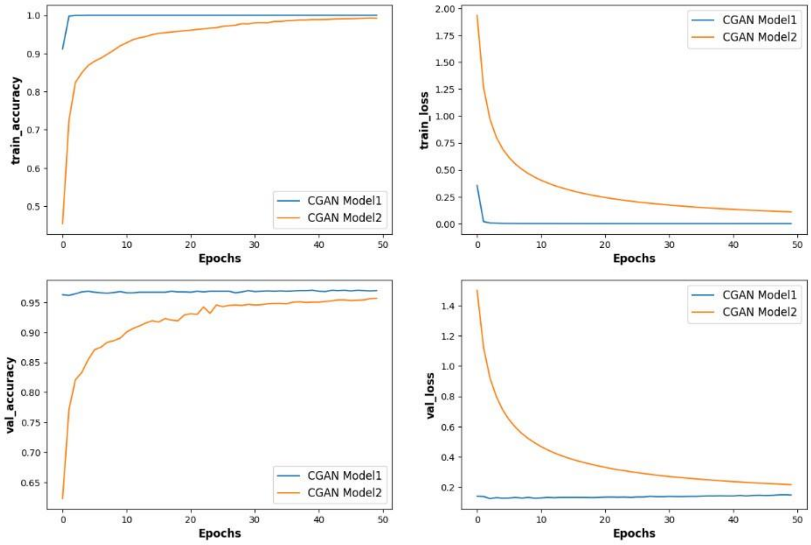


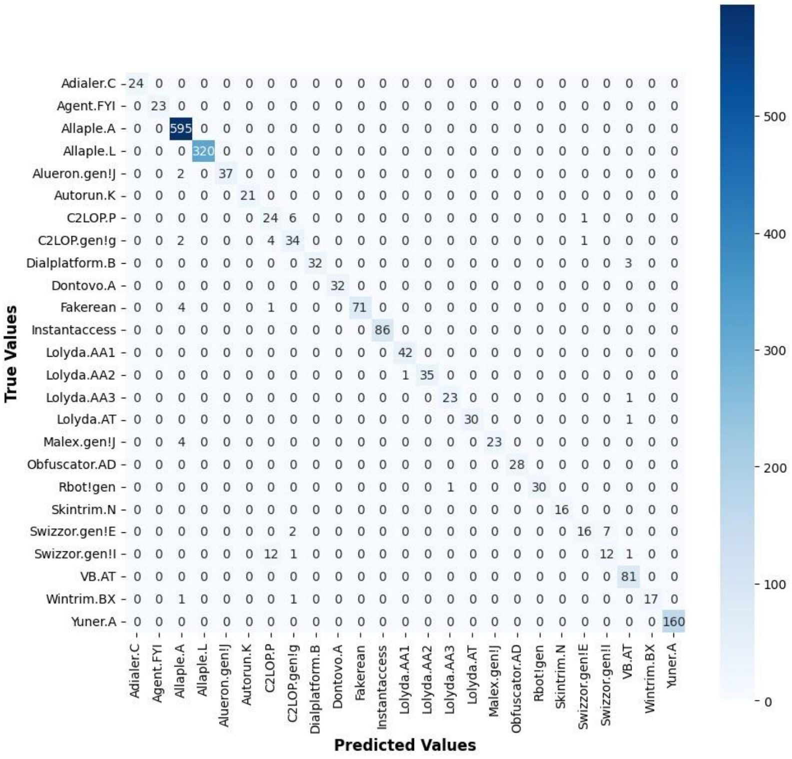
Figure 8. Accuracy-loss curve for Malware Classification Models

Table 1. Performance Matrix for Malware Classification Model CGAN Model1.

Malware Classes	Precision	Recall	F1-score	Samples
Adiler.C	1.00	1.00	1.00	122
Agent.FYI	1.00	1.00	1.00	116
Allaple.A	0.98	1.00	0.99	2,949
Allaple.L	1.00	1.00	1.00	1,591
Alueron.gen!J	1.00	0.95	0.97	198
Autorun.K	1.00	1.00	1.00	106
C2LOP.P	0.59	0.77	0.67	200
C2LOP.gen!g	0.77	0.83	0.80	146
Dialplatform.B	1.00	0.91	0.96	177
Dontovo.A	1.00	1.00	1.00	162
Fakerean	1.00	0.93	0.97	381
Instantaccess	1.00	1.00	1.00	431
Lolyda.AA1	0.98	1.00	0.99	213
Lolyda.AA2	1.00	0.97	0.99	184
Lolyda.AA3	0.96	0.96	0.96	123
Lolyda.AT	1.00	0.97	0.98	159
Malex.gen!J	1.00	0.85	0.92	136
Obfuscator.AD	1.00	1.00	1.00	142
Rbot!gen	1.00	0.97	0.98	158
Skintrim.N	1.00	1.00	1.00	80
Swizzor.gen!E	0.89	0.64	0.74	128
Swizzor.gen!I	0.63	0.46	0.53	132
VB.AT	0.93	1.00	0.96	408

Wintrim.BX	1.00	0.89	0.94	97
Yuner.A	1.00	1.00	1.00	800
Accuracy			**0.97**	**1869**
Macro avg	**0.95**	**0.92**	**0.93**	**1869**
Weighted avg	**0.97**	**0.97**	**0.97**	**1869**

The confusion matrix highlights that CGAN Model1 accurately classifies most classes, except for ‘Swizzor.gen!I’ (Figure 9). It performs well in distinguishing similar classes, such as ‘Allaple.A’, ‘Allaple.L’, and various ‘Lolyda’ variants (‘Lolyda.AA1’, ‘Lolyda.AA2’, ‘Lolyda.AA3’, ‘Lolyda.AT’). The ROC-AUC curve for malware classification, showing that the AUC value for CGAN Model1 surpasses that of CGAN Model2, indicating better overall classification performance and reduction in false positive rate (Figure 10).



	Adialer.C	Agent.FYI	Allaple.A	Allaple.L	Alueron.gen!J	Autorun.K	C2LOP.P	C2LOP.gen!g	Dialplatform.B	Dontovo.A	Fakerean	Instantaccess	Lolyda.AA1	Lolyda.AA2	Lolyda.AA3	Lolyda.AT	Malex.gen!J	Obfuscator.AD	Rbot!gen	Skintrim.N	Swizzor.gen!E	Swizzor.gen!I	VB.AT	Wintrim.BX	Yuner.A
Adialer.C	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Agent.FYI	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.A	0	0	595	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.L	0	0	0	320	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alueron.gen!J	0	0	2	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Autorun.K	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2LOP.P	0	0	0	0	0	0	24	6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C2LOP.gen!g	0	0	2	0	0	0	4	34	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Dialplatform.B	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
Dontovo.A	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fakerean	0	0	4	0	0	0	1	0	0	0	71	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Instantaccess	0	0	0	0	0	0	0	0	0	0	0	86	0	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA1	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA2	0	0	0	0	0	0	0	0	0	0	0	0	1	35	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	1	0	0
Lolyda.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	1	0	0
Malex.gen!J	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0
Obfuscator.AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0
Rbot!gen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	30	0	0	0	0	0	0
Skintrim.N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0
Swizzor.gen!E	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	16	7	0	0	0
Swizzor.gen!I	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0
VB.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	81	0	0
Wintrim.BX	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0
Yuner.A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	160

Figure 9. Confusion Matrix of Malware Classification Model CGAN Model1

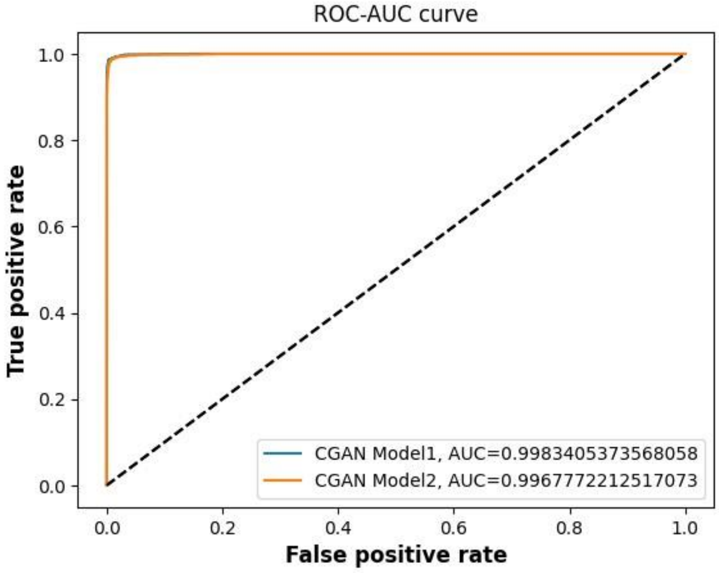


Figure 10. ROC-AUC curve for Malware Classification Models

4.2. Malware Detection

The CGAN model was adapted for binary classification to detect malware. CGAN Model1 was compiled using the *Adam* optimizer, while CGAN Model2 used the *Stochastic Gradient Descant (SGD)*optimizer, both using *binary_crossentropy* as the loss function. These models were trained on Dataset-2 for 50 epochs. The resulting accuracy and loss trends over the training process are illustrated in Figure 11, demonstrating how each model performed across epochs during training.

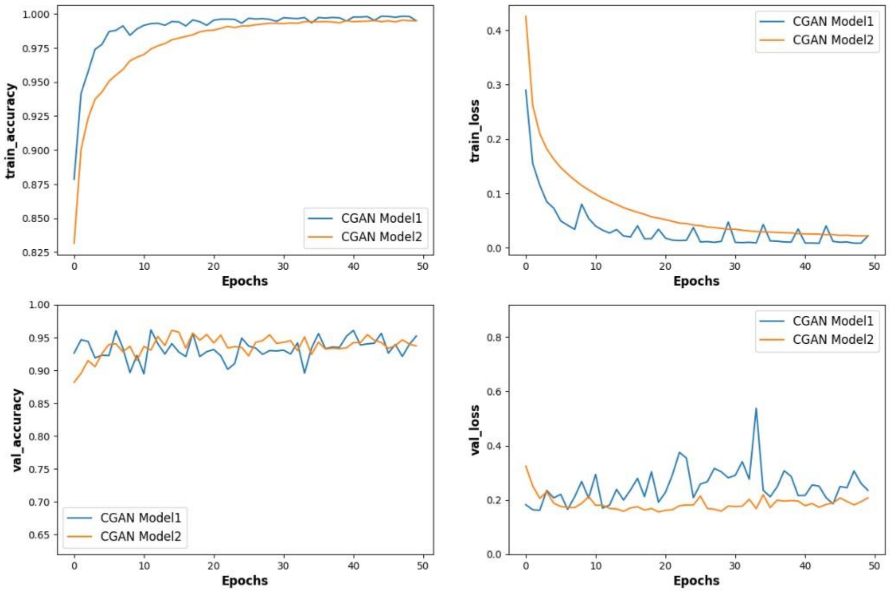


Figure 11. Accuracy-loss curve for Malware Detection Models

For the malware detection, the CGAN Model1 achieves the detection accuracy of 95.19% and the GAN Model2 achieves the detection accuracy of 93.69% on the test dataset. The performance matrix of the malware detection model CGAN Model1 shown in Table 2.

Table 2. Performance Matrix for CGAN Model1 for malware Detection.

Malware Classes	**Precision**	**Recall**	**F1-score**	**Samples**
Malicious Images	0.93	0.98	0.95	2383
Normal Images	0.98	0.93	0.95	2438
Accuracy			**0.95**	**4821**
Macro avg	**0.95**	**0.95**	**0.95**	**4821**
Weighted avg	**0.95**	**0.95**	**0.95**	**4821**

The confusion matrix for the malware detection model CGAN Model1, shows the significant reduction in the false positive rate as compared to false negative rate (Figure 12). The performance of CGAN Model 1 is better than the CGAN Model2 as higher ROCAUC value is achieved by CGAN Model1(Figure 13).

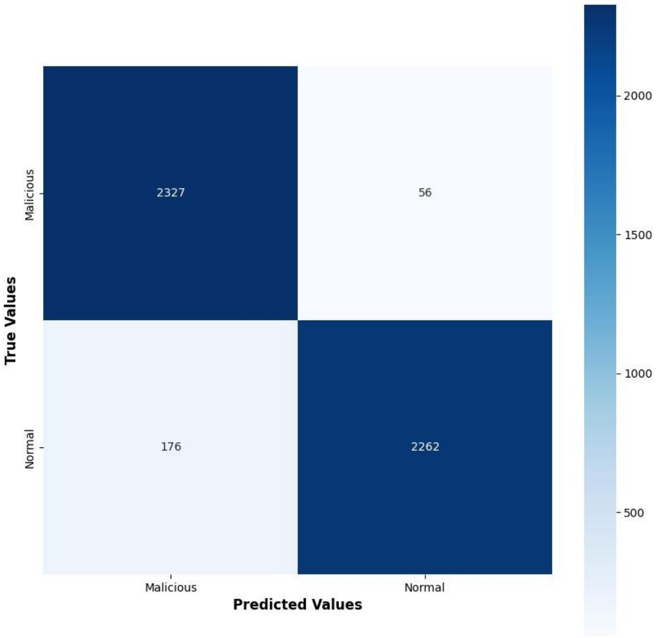


Figure 12. Confusion matrix of Malware Detection Model CGAN Model1

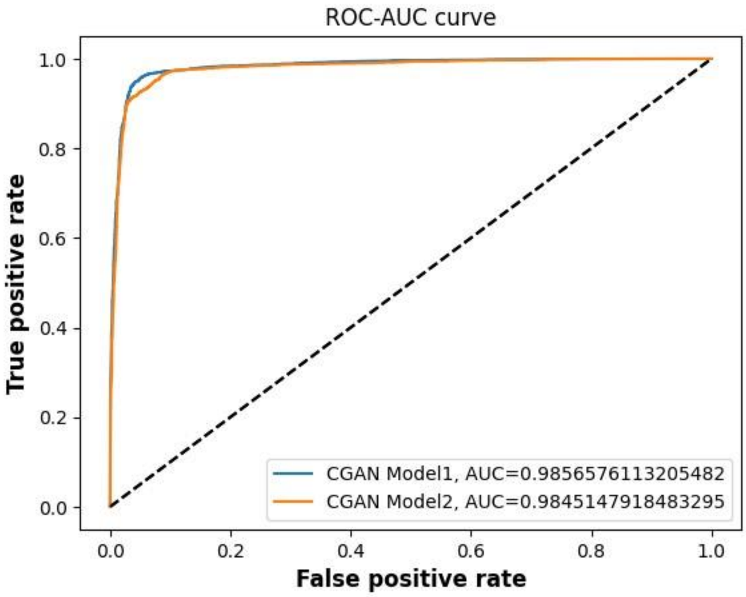


Figure 13. ROC-AUC Curve Malware Detection Models

5. Results And Discussion

The experimental results show that the Conditional Generative Adversarial Network (CGAN) using the *Stochastic Gradient Descant (SGD)* optimizer converges gradually as compared to the *Adam* optimizer. The performance of the CGAN model using the *Adam* optimizer is better. From Table 3, we can see that the proposed CGAN models are taking very little time to train and converge in 50 epochs. From the grayscale malware image classification takes less training time as compared to the RGB image-based binary classification. Therefore, the generative models are more efficient to train large image datasets.

Table 3. Performance Comparison of proposed models.

Proposed GAN Models	**Optimizer**	**Dataset Type**	**Image Type and Size**	**Training Time**	**Validation Accuracy**
CGAN Model1	Adam	Malimg	Grayscale, 224×224	1 min 6 s	96.96%
CGAN model2	SGD	Malimg	Grayscale, 224×224	1 min 48 s	95.67%
CGAN Model1	Adam	IEEEDataPort	RGB, 224×224	6 min 43 s	95.19%
CGAN Model2	SGD	IEEEDataPort	RGB, 224×224	6 min 27 s	93.69%

5.1. Performance Comparison with State-Of-The-Art

The performance of a GAN model for malware detection and classification is compared with the state-of-the-art in Table 4. The proposed malware classification model ‘CGAN Model1’ using *Adam* optimizer on the Malimg dataset outperformed all the compared models with the highest classification accuracy of 96.96%.

Table 4. Performance Comparison with State-of-The-Art.

References and Year	**Dataset**	**Model**	**Accuracy**
[16] (2019)	Malimg	DCGAN	90%
[12] (2022)	Malimg, MalExe	AC-GAN	95%
[13] (2023)	Malimg	ShuffleNetV2, and MobileNetV3	94.2% 95%
[14] (2023)	Malimg	E-GAN	96.35%
[15] (2023)	VirusTotal	Vanilla Byteplot Model	95.76%
Proposed GAN model for Malware Classification	**Malimg**	**CGAN Model1**	**96.96%**
	Malimg	**CGAN Model2**	**95.67%**
Proposed GAN model for Malware Detection	**IEEEDataPort**	**CGAN Model1**	**95.19%**
	IEEEDataPort	**CGAN Model2**	**93.69%**

6. Conclusion and Future Scope

The Generative Adversarial Network (GAN) is more time and resource efficient with the large image dataset and also able to generate synthetic input samples. The capability of a generative model is extended as a CGAN model for malware detection and classification to improve the accuracy and robustness by reducing the false positive rate. The GAN-based approach provides an effective means of generating adversarial samples that enhance the classifier's ability to detect sophisticated and novel malware. The GAN-based models are effective for adversarial attacks, obfuscated malware, and novel and zero-day attacks. The future research directions include realtime classification of malwares, cross-platform malware detection like mobile devices and IoT devices, integrating GAN with other DL models, detecting adversarial attacks, and application of transfer learning in GAN models.

Acknowledgements

The authors are thankful to the Ministry of Social Justice &Empowerment, India, for providing financial support as an RGNF-SC fellowship for the research work.

References

[1] Prachi Chauhan, Hardwari Lal Mandoria, and Alok Negi, “Security and Privacy Defensive Techniques for Cyber Security Using Deep Neural Networks (DNNs),” in *Advanced Smart Computing Technologies in Cybersecurity and Forensics*, I., 2021, pp. 11–22.

[2] M. Rai and H. L. Mandoria, “A Study on Cyber Crimes, Cyber Criminals and Major Security Breaches,” *International Research Journal of Engineering and Technology*, no. July, 2008.

[3] K. Kumar and A. Dwivedi, “Big Data Issues and Challenges in 21 st Century,” *International Journal on Emerging Technologies (Special Issue NCETST-2017)*, vol. 8, no. 1, pp. 72–77, 2017, [Online]. Available: www.researchtrend.net

[4] A. Dwivedi, R. P. Pant, S. Pandey, and K. Kumar, “Internet of Things’ (IoT’s) Impact on Decision Oriented Applications of Big Data Sentiment Analysis,” in *Proceedings - 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU 2018*, 2018. doi: 10.1109/IoTSIU.2018.8519922.

[5] H. Shahnawaz, S. C. Gupta, C. Mukesh, and H. L. Mandoria, “A proposed model for intrusion detection system for mobile adhoc network,” in *2010 International Conference on Computer and Communication Technology, ICCCT-2010*, 2010. doi: 10.1109/ICCCT.2010.5640420.

[6] "Cybersecurity Threats, Forensics, and Challenges_Springer".

[7] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014. doi: 10.1007/978-3-658-40442-0_9.

[8] A. Dash, J. Ye, and G. Wang, "A Review of Generative Adversarial Networks (GANs) and Its Applications in a Wide Variety of Disciplines: From Medical to Remote Sensing," *IEEE Access*, vol. 12, 2024, doi: 10.1109/ACCESS.2023.3346273.

[9] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets Mehdi," *arXiv:1411.1784v1 [cs.LG] 6 Nov 2014 Conditional*, 2018.

[10] S. Pujari, H. L. Mandoria, R. P. Shrivastava, and R. Singh, "To Identify Malware Using Machine Learning Algorithms," in *Communications in Computer and Information Science*, 2022. doi: 10.1007/978-3-031-10551-7_9.

[11] M. Rai and H. L. Mandoria, "Network Intrusion Detection: A comparative study using state-of-the-art machine learning methods," in *IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques, ICICT 2019*, 2019. doi: 10.1109/ICICT46931.2019.8977679.

[12] R. Nagaraju and M. Stamp, "Auxiliary-Classifier GAN for Malware Analysis," in *Advances in Information Security*, vol. 54, 2022. doi: 10.1007/978-3-030-970871_2.

[13] K. T. Chui, B. B. Gupta, V. Arya, R. Bansal, and F. Colace, "A Lightweight Generative Adversarial Network for Imbalanced Malware Image Classification," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Nov. 2023. doi: 10.1145/3647444.3652455.

[14] D. O. Won, Y. N. Jang, and S. W. Lee, "PlausMal-GAN: Plausible Malware Training Based on Generative Adversarial Networks for Analogous Zero-Day Malware Detection," *IEEE Trans Emerg Top Comput*, vol. 11, no. 1, pp. 82–94, Jan. 2023, doi: 10.1109/TETC.2022.3170544.

[15] C. Reilly, S. O'Shaughnessy, and C. Thorpe, "Robustness of Image-Based Malware Classification Models trained with Generative Adversarial Networks," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jun. 2023, pp. 92–99. doi: 10.1145/3590777.3590792.

[16] Y. Lu and J. Li, "Generative Adversarial Network or Improving Deep Learning Based Malware Classification."

[17] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *ACM International Conference Proceeding Series*, 2011. doi: 10.1145/2016904.2016908.

[18] B. Saridou, J. Rose, S. Shiaeles, and B. Papadopoulos, "48,240 Malware samples and binary visualisation images for machine learning anomaly detection (2021)," 2022.

AUTHORS

Krishna Kumar received the BTech in Information Technology in 2013 and the MTech in Computer Science and Engineering in 2016. Currently, he is pursuing a PhD in information technology from the Department of Information Technology, College of Technology, G. B. Pant University of Agriculture and Technology, Pantnagar, Uttarakhand, India. His research interests focus on Cyber Security, Machine Learning, and Cyber Forensics.

Hardwari Lal Mandoriacompleted his doctoral degree from Maulana Azad National Institute of Technology/ Barkatullah University, Bhopal (Madhya Pradesh) in Computer Science & Engineering in 2003. He is a Professor in Department of Information Technology, College of Technology, G. B. Pant University of Agriculture and Technology, Pantnagar, Uttarakhand, India. He has professional experience of more than 38 years in teaching and research in the field of Computer Science and Engineering including two years of foreign teaching experience under UNDP at Ethiopia. He has published more than 100 research papers in various National and International peer-reviewed journals. He has written one book, many book chapters and successfully guided 03 PhD and 21 MTech students. His research expertise covers a wide range of areas, including Computer Networks, Information Security, Wireless

Communication Networks, Cyber Security & Forensics.

Rajeev Singhreceived his M.Tech. degree in computer science & engineering from Indian Institute of Technology, Roorkee, India in 2008 and his Ph.D. degree from National Institute of Technology, Hamirpur, India in 2014. Currently, he is working as a Professor with the Department of Computer Engineering, Govind Ballabh Pant University of Agriculture & Technology, Uttarakhand, India. His research interest includes Computer Networks & Network Security, Information Systems Network Security Modeling and Simulation Wireless LANs Programming Languages.

Shri Prakash Dwivedireceived his M. E. degree in Computer Science & Automation from Indian Institute of Science (IISc) and his PhD degree in Computer Science & Engineering from Indian Institute of Technology (Banaras Hindu University), Varanasi. Currently, he is working as an Assistant Professor with the Department of Information Technology, G. B. Pant University of Agriculture & Technology, Pantnagar, India. His research interest includes Algorithms, Graph Matching and Pattern Recognition.

Paras received his BTech, MTech, and PhD degrees in Electronics & Communication Engineering from G. B. Pant University of Agriculture & Technology, Pantnagar, India. Currently, he is working as a Professor in the Department of Electronics & Communication Engineering at G. B. Pant University of Agriculture & Technology, Pantnagar, India. He has more than 21 years of professional experience in teaching and research. He has guided 14 MTech students and one PhD student. His primary research interests include Antennas, Metamaterials and Artificial Neural Networks.