

# Robust Android Malware Detection with CTGAN

An M.Tech project report submitted in the 4th semester of

**Master of Technology**

in  
Cryptology and Security

by

**Bivash Sarkar**

(Roll No. CrS2203)

Under the Supervision of

**Shri Sanchit Gupta**

Scientist 'F', SAG, DRDO

**Shri Debrup Chakroborty**

Professor, ISI Kolkata

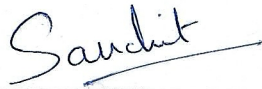
Cryptology and Security Department

**ISI Kolkata**

2nd July, 2024

## CERTIFICATE

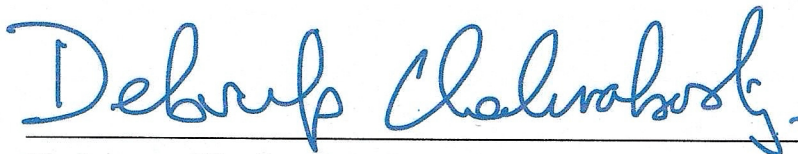
This is to certify that the dissertation entitled “**Robust Android Malware Detection with CTGAN**” submitted by **Bivash Sarkar** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.



---

**Sanchit Gupta**

Scientist 'F', Scientific Analysis Group,  
Defence Research and Development Organisation.



---

**Debrup Chakraborty**

Professor, Cryptology and Security Research Unit,  
Indian Statistical Institute, Kolkata.

## ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisors, Sri Sanchit Gupta and Sri Debrup Chakraborty for their invaluable guidance, support, and encouragement throughout this project. Their expertise and insights have been instrumental in shaping the direction of this study.

I am also grateful to Indian Statistical Institute, Kolkata and Defence Research and Development Organization for providing the necessary resources and facilities to conduct this research.

**Bivash Sarkar**

# *Abstract*

In this paper our main objective is to make a robust malware detector system by enhancing its ability to detect malicious applications. Machine learning based model has been used to detect or classify malware and benign samples, while the malware attackers have strong motivation to attack such ML based algorithms. Malware attackers usually have no access to the detailed structures and parameters of the machine learning models used by malware detection systems, and therefore they can only perform black-box attacks. With the proliferation of malware threats, the development of robust detection methods is imperative. Generative Adversarial Networks (GANs) have recently emerged as a promising avenue for generating synthetic data, offering potential applications in augmenting datasets for malware detection. This paper presents a comparative analysis of contemporary GANs with Conditional Tabular GANs (CTGAN) in the context of detecting malware and benign samples generated through GANs. Through extensive experimentation on diverse datasets, including both benign and malicious samples, we demonstrate that CTGAN outperforms contemporary GAN architectures in generating synthetic data that closely resembles real-world malware behaviors. Our evaluation metrics encompass various aspects of detection accuracy, including precision, recall, F1-score, TPR, AUC-ROC, confusion matrix and generator and discriminator loss. Additionally, we analyze the robustness of the generated samples against state-of-the-art malware detection techniques. The results indicate that CTGAN exhibits superior performance in producing synthetic malware instances that challenge existing detection methods like MaLGAN and LSGAN, thereby showcasing its potential for enhancing the efficacy of malware detection systems. CTGAN enhances adversarial training with around 20% when compared with untrained detector. This study contributes to the advancement of GAN-based approaches in cybersecurity and underscores the significance of leveraging synthetic data generation techniques for improving malware detection capabilities.

# Chapter 1

## Introduction

Android malware detection plays a critical role in cybersecurity due to the widespread adoption of Android devices globally. Some of the commonly known attacks are data breaches, financial fraud, mitigation of privacy risk etc. So it is very much necessary to prevent such attacks by detecting the malware applications properly in our system. Machine learning based malware detector can be intruded by the attacker using black-box adversarial attack. Adversarial training emerges as a promising strategy in the quest for robust malware detection models. Unlike traditional training methods that rely solely on benign and malicious samples, adversarial training integrates adversarial examples into the training process. These examples are carefully crafted perturbations of input data designed to deceive the model, thereby enhancing its resilience against evasion attacks. By exposing the model to such adversarial inputs during training, it learns to recognize and appropriately handle subtle variations in malware instances, thus fortifying its defenses against evolving threats.

Furthermore, the integration of Ian Goodfellow's Generative Adversarial Networks (GANs) (1) into the training pipeline presents a novel avenue for enhancing model robustness. GANs, comprising a generator and a discriminator network engaged

---

in a min-max game, excel at generating synthetic data that closely resembles real-world samples. In the context of malware detection, the contemporary GANs fall short within the context of tabular data and so the application of GANs, particularly Conditional Tabular GANs (CTGAN), facilitates the generation of diverse and realistic malware instances. By augmenting the training dataset with synthetic samples produced by CTGAN, the model gains exposure to a broader spectrum of malware variants, thereby enhancing its generalization capabilities and improving overall performance.

The primary objective of this research is to investigate the efficacy of adversarial training with CTGAN-generated samples in enhancing Android malware detection. This entails two key parameters: first, assessing the fidelity of generated adversarial samples compared to contemporary GANs to ensure the robustness and realism of the synthetic data; and second, evaluating the impact of integrating CTGAN-generated samples on the malware detection performance of the model. By rigorously examining these aspects, this study aims to contribute valuable insights into the potential of adversarial training and synthetic data augmentation in bolstering cybersecurity defenses against Android malware threats.

# Chapter 2

## Literature Review

Android malware has become a pressing concern in the domain of cybersecurity, particularly with the proliferation of mobile devices and the widespread adoption of the Android operating system. The vast user base and open nature of Android provide fertile ground for malicious actors to exploit vulnerabilities, posing significant risks to users' personal information, financial data, and overall device integrity. This increasing threat landscape has driven researchers to devote considerable effort to developing effective malware detection techniques. The first notable work in adversarial examples was by Szegedy et al (2) which proposes slight addition of perturbations on original dataset to perform adversarial attacks. This was then further used for using adversarial attacks beyond image classification. Generally, there are three main methods for creating adversarial examples: gradient-based, optimization-based, and Generative Adversarial Networks (GAN)-based. The first two methods face three significant challenges: (1) they require access to the white-box architecture and continuous knowledge of model parameters, (2) their optimization process is slow and can only optimize perturbations for one specific instance at a time, and (3) they produce adversarial examples with low perceptual quality.

This research efforts rely heavily on training data, which can be resource-intensive and may not always represent the full spectrum of potential malware. To counter this limitation, researchers have turned to Generative Adversarial Networks (GANs) for malware detection. GANs are particularly useful in training on unbalanced datasets, which is common in cybersecurity where the number of malicious samples can be significantly lower than benign ones(3). One prominent application of GANs in this field is MalGAN(4), which proposes a GAN-based algorithm to generate adversarial malware examples. These adversarial examples are designed to bypass black-box machine learning-based detection models, highlighting potential weaknesses in existing defenses and prompting the development of more resilient detection systems. To enhance the robustness of malware detection systems, researchers have proposed several advancements. Yuan et al. (5) introduced GAPGAN, an advanced adversarial attack framework designed to generate adversarial examples specifically targeting binary-based malware detection systems through the use of Generative Adversarial Networks (GANs). This framework operates by appending carefully crafted adversarial perturbations to the original malware binaries, ensuring that the malicious functionality of the binaries remains intact. By doing so, GAPGAN effectively bypasses detection mechanisms, revealing vulnerabilities in existing malware detection systems and highlighting the need for more robust defensive strategies. One such advancement is Mal-LSGAN(6) which uses Least Square (LS) loss function and new activation function combinations, Mal-LSGAN achieves a higher Attack Success Rate (ASR) and a lower True Positive Rate (TPR). To counter shortcomings like mode collapse and training instability, LSGAN-AT (7), which comprises two modules: the LSGAN module and the Adversarial Training (AT) module was proposed. The LSGAN module generates more effective and smoother adversarial malware examples by utilizing new network structures and Least Square (LS) loss to optimize boundary samples. The AT module uses these adversarial examples for adversarial



training, thereby creating a more robust Malware Detector (RMD) based on machine learning . A notable limitation of current GAN-based approaches, such as MalGAN and LSGAN-AT, is their performance with tabular data. Many malware detection techniques focus on the binary and sequential data typically associated with application behavior and network traffic. However, malware can also present itself in tabular format, which includes structured data sets such as permissions, API calls, and system events. These structured data forms are crucial for understanding the context and correlations between different malware characteristics.

<b>Author(s)</b>	<b>Title</b>	<b>Key Contributions and Findings</b>
Ian Goodfellow et al.	Generative Adversarial Nets	Introduction of GANs, demonstrating their ability to generate realistic images and adversarial training process.
Radford et al.	Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks	Introduced DCGAN, showing that CNNs can be used in GANs to improve the quality of generated images.
Arjovsky et al.	Wasserstein GAN	Proposed WGAN, addressing the training instability of GANs by using the Wasserstein distance.

<b>Author(s)</b>	<b>Title</b>	<b>Key Contributions and Findings</b>
Miyato et al.	Spectral Normalization for Generative Adversarial Networks	Introduced spectral normalization to stabilize GAN training and improve the quality of generated samples.
Karras et al.	Progressive Growing of GANs for Improved Quality, Stability, and Variation	Presented a method for progressively growing GANs, leading to higher resolution and more stable training.

MalGAN, InfoGAN, LSGAN and WGAN generally used for generating synthetic images, not for tabular dataset, also for a dataset with non Gaussian or multimodal distribution they can not generate effective adversarial samples.

# Chapter 3

## Methodology

Our dataset here is tabular dataset containing 6000 samples of Android applications and 160 feature columns. It is taken from Virusshare (8) and benign dataset from playstore. The feature columns are binary- each entity of the feature vector represents an API, for an particular example if the API feature is present then the entity is 1 otherwise 0. There is a few API whose presence occurs the Benign sample to a Malicious sample. The training procedure begins with the collection of a diverse dataset comprising both benign and malicious samples. This dataset is split into training, validation, and test sets. The training phase involves feeding the input data into the model and updating the model parameters through backpropagation to minimize the categorical cross-entropy loss function. An Adam optimizer is utilized for this purpose due to its adaptive learning rate capabilities, which facilitate efficient convergence.

In our proposed framework we use CTGAN (conditional tabular GAN) to generate adversarial samples. CtGAN consists conditional generator. Unlike the generator of MalGAN, LSGAN or WGAN conditional generator generates fake samples conditioned on a class level. CTGAN can generate more effective synthesized tabular

dataset, specially the table containing discrete columns, than of MalGAN, LSGAN, WGAN and InfoGAN.

First, we train the generator for generating adversarial samples and check the accuracy of the detector on these samples. When the the generator is trained ‘good enough’ that is they can bypass the detector most of the time, we stop training and get those adversarial samples. Then we label them all as ‘malware’ and combine with the original training dataset. We train our model (Malware detector) with this combined data that is we adversarially train the malware detector system.

During retraining, the model is exposed to a mix of benign, malicious, and adversarial samples in each batch. This exposure helps the model to recognize and correctly classify adversarial examples, thereby improving its resilience against adversarial attacks. The performance of the retrained model is continuously monitored using a validation set that includes adversarial samples to ensure that the model’s ability to detect genuine malware is not compromised.

For a certain reasons we will see that CTGAN performs better in enhancing the malware detector and restricts the drop out of TPR than the pre existing methods using MalGAN or LSGAN-AT. We are discussing these here,

1. It is specifically designed to handle tabular datasets, which are common in many real-world applications, including malware detection. In contrast, other GAN variants like MalGAN, LSGAN, WGAN, and InfoGAN are generally tailored for generating synthetic images or continuous data. CTGAN outperforms other GANs in the context of adversarial training for tabular data.

2. CTGAN is designed to model the distribution of tabular data effectively. It handles the challenges of mixed data types, missing values, and complex feature correlations better than other GANs, which are primarily designed for image data. It can also capture the distribution of discrete and continuous features, preserving the relationships and dependencies between features that are critical in malware detection datasets.
3. In this CTGAN Conditional GANs (9), are constructed by feeding categorical attributes to both the generator and discriminator. They generate the rest of the features based on these categorical input attributes. This helps to rebalance the imbalanced categorical data, by generating samples for the minority categories.
4. The adversarial samples generated by CTGAN are realistic and varied, providing a comprehensive dataset for adversarial training. This helps the malware detector learn to recognize and resist a wide range of adversarial attacks, improving its robustness. For CTGAN, the generator loss is adapted to incorporate the conditional distributions of the data. The loss function for CTGAN's generator can be expressed similarly to the standard GAN, but it also conditions on specific columns in the tabular data. A majority of applications for conditional GAN are concerned with synthesizing table by giving the label for the adversary that should be generated. Nonetheless, in the case of tabular data, this could be the shape of data on a multimodal distribution and can be used to inject information as prior knowledge to the generator.

# Chapter 4

## Results

A good malware detector should detect the malicious samples with more accuracy. So we concern about the performance of a detection model through it's malware detection rate or *True Positive Rate* (TPR). We can say an Adversarial attack has been successfully executed if the adversarial samples generated by GAN can decreases the TPR of the detector by a significant level without knowing any internal architecture of the ML based detector (Black-box). But after retraining our model with the adversarial samples generated by the generator the model's TPR has been increased from the previous when the model is attacked by the adversarial samples without retraining.

So the performance of a GAN is depending on two things:

Firstly, how the adversarial samples generated by this GAN can penetrate the detector and secondly how good the adversarial samples generated by it can train the model and make an enhanced robust model against the malicious samples. After facing adversarial attacks we need to retrain our malware detector. But here we should do adversarial training, i.e we need to train the detector with the adversarial examples. This is called adversarial training.

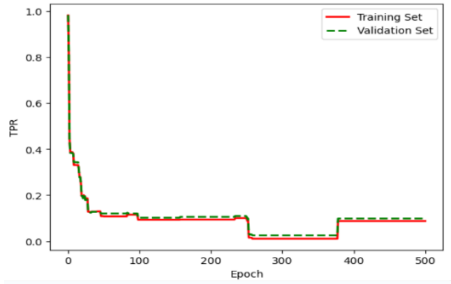
**Table 4.1:** TPR(in %) of the models before and after adversarial training by CTGAN

Classifier	TPR without adversarial training	TPR after adversarial training
LR	0.062%	5.72%
SVM	1.77%	29.79%
DT	0.0027%	0.04%
RF	0.43%	11.57%
MLP	0.003%	26.49%
DNN	2.68%	39.70%

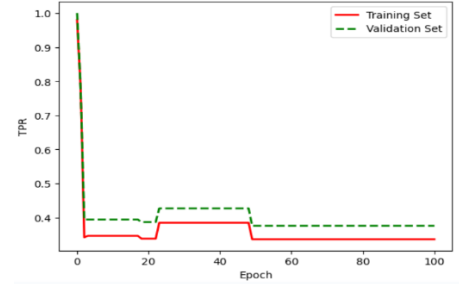
Here we note down the TPR of the malware detector using various machine learning and deep learning based classifier's performance against the Mal-CTGAN generated adversarial samples. The TPR here are calculated in percentage form. The second column contains the TPRs of the detectors without adversarial training and the third column contains TPRs against CTGAN generated samples after the adversarial training. We use ML powered Malware detectors, which were based on six well known ML algorithms for classification: Logistic Regression (LR), Support Vector Machine (SVM) with RBF kernel, Multi-Layer Perceptron (MLP), Decision Tree (DT), Random Forest (RF) and Deep Neural Network (DNN).

Here we generate 500 adversarial samples from the generator of CTGAN for all classifiers and use these to adversarially train the model along with the original training dataset, with labelling all these adversarial samples as 'malware'. Then we calculate the TPR on the test dataset.

After adversarial training we see that the detector being enhanced to detect the malicious samples. Training with the CTGAN generated adversarial samples to all the classifiers, almost all of them can detect the malicious samples with significantly high accuracy before the adversarial training.



**Figure 4.1:** Before adversarial training



**Figure 4.2:** After adversarial training

Now we are going to compare the CTGAN generated adversarial trained model against the pre existing models used for adversarial training. To compare the performance of the CTGAN with each of the other GANs like MalGAN, LSGAN, InfoGAN etc for enhancing the robustness of the model, we make two tables. In the first table we will see how Mal-CTGAN generated adversarial samples can bypass the detector more than other GAN models as the attackers.

In the second table we put the results in a matrix form where each row mentions the adversarially trained model for different GANs and each column represents the adversarial examples from the generator of each GAN. Here the performance of a GAN relies on how the adversarial samples generated from it can train the model such that the TPR of the adversarial trained model is increased against the adversarial samples generated by rest of the GANs. We verify for two ML models: SVM and DNN for each of the adversarially trained detector with adversarial samples generated by five GANs: MalGAN, InfoGAN, WGAN, LSGAN-AT and Mal-CTGAN.

In this above table we see that CTGAN can generate adversarial malware samples which are being more undetectable than the samples generated by MalGAN, LSGAN, InfoGAN and WGAN. Now in this table 4 the first column is for malware detectors trained with adversarial samples generated by MalGAN, InfoGAN, WGAN, LSGAN and CTGAN with two classifiers as SVM and DNN as detector.



**Table 4.2:** Comparing the effectiveness of bypassing the detector by the adv. samples from each GANs without adversarial training

Classifier	Adversarial samples generated by the GANs				
	<i>MalGAN</i>	<i>InfoGAN</i>	<i>WGAN</i>	<i>LSGAN</i>	<i>Mal-CTGAN</i>
LR	0.63	2.58	2.82	0.0	0.0
SVM	10.93	12.36	9.50	4.81	1.48
DT	0.68	2.67	0.85	0.032	0.0
RF	11.62	3.11	5.06	3.28	0.45
MLP	10.79	14.42	3.58	1.82	0.0

**Table 4.3:** Comparing the robustness of the models after adversarial training

Adv. trained Model	Adversarial samples generated by the GANs				
	<i>MalGAN</i>	<i>InfoGAN</i>	<i>WGAN</i>	<i>LSGAN</i>	<i>CTGAN</i>
MalGAN (SVM)	21.49	27.62	4.34	16.23	0.06
InfoGAN (SVM)	5.88	7.95	2.34	2.38	0.00
WGAN (SVM)	12.32	21.90	7.86	1.23	0.18
LSGAN (SVM)	36.87	28.33	3.45	2.82	0.03
Mal-CTGAN (SVM)	42.63	26.24	39.06	17.85	5.72
MalGAN (DNN)	29.08	27.33	15.86	32.10	1.18
InfoGAN (DNN)	10.26	17.92	14.68	11.04	0.47
WGAN (DNN)	10.52	34.75	18.08	6.03	1.33
LSGAN (DNN)	43.81	42.18	17.44	22.63	10.56
Mal-CTGAN (DNN)	55.26	38.07	37.12	34.88	28.93

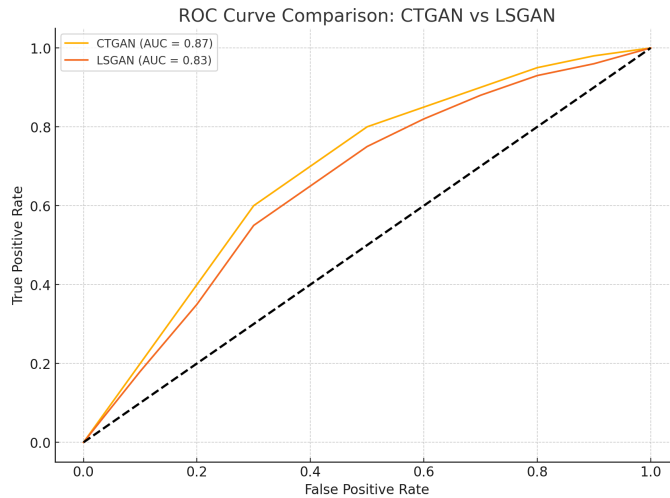
For example MalGAN (SVM) means adversarially trained detector based on SVM classifier trained on MalGAN generated adversarial samples. We evaluate the performance of these adversarially trained models on adversarial examples from GANs: MalGAN, InfoGAN, WGAN, LSGAN and CTGAN. Subsequent columns show detection accuracy for adversarial samples generated by various GAN attackers vs the DNN classifier as *Discriminator*. Like the same before we generate 500 adversarial samples and compare these models with their corresponding true positive rate.

We see that the CTGAN-trained model that is model trained by adversarial samples generated by CTGAN gives much better TPR than the other adversarial trained

models.

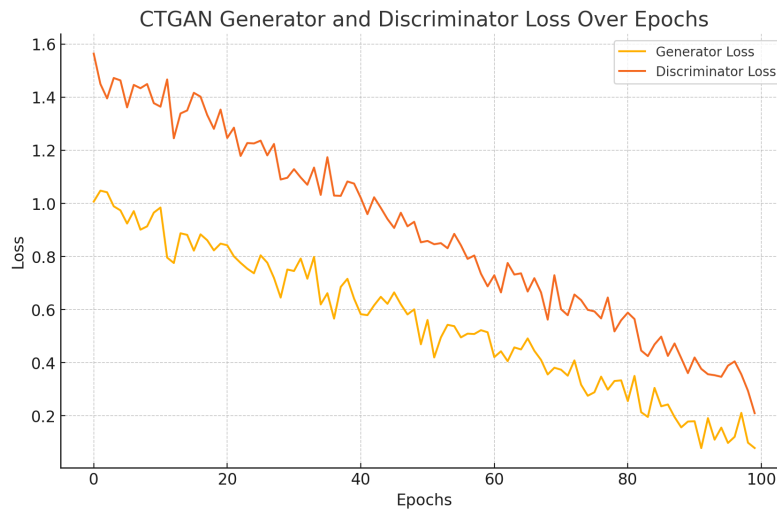
Now we are going to verify with some data analytical methods that how Mal-CTGAN can generate more similar looking synthetic tabular data or adversarial samples compare to LSGAN model.

Firstly, we compare with ROC-AUC metric generated for two models LSGAN-AT and CTGAN trained detector (Fig. 7). Here is the ROC curve comparison between two models retrained by adversarial samples generated by CTGAN and LSGAN. The ROC curves illustrate the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) for the two models. The Area Under the Curve (AUC) values are provided in the legend, showing CTGAN with an AUC of 0.87 and LSGAN with an AUC of 0.83. This suggests that CTGAN slightly outperforms LSGAN for the adversarial training.



**Figure 4.3:** ROC-AUC for CTGAN and LSGAN

Now we discuss on generator loss of CTGAN.



**Figure 4.4:** Configuring generator loss and discriminator loss

The generator loss starts relatively high and decreases gradually (Fig. 10). This shows that the generator is becoming better at producing realistic data that can fool the discriminator. A consistently decreasing generator loss suggests that the generator is learning and improving over time. The above graph suggests that the training process is proceeding correctly, with both models learning and improving. The leveling off of both losses indicates a stable training process, which is a positive sign of effective GAN training.

We are going to check the Performance of Adversarially trained models on the combined adversarial samples.

Earlier in table [3.3] we compare the adversarial trained models with CTGAN and other pre-discussed GAN and found that the model trained by CTGAN generated adversarial samples performs better than its counterparts. Here we want to make sure about our proposed method for adversarial training with respect to the other benchmark methods on the combining of generated adversarial examples by MalGAN, LSGAN and CTGAN in the setting of DNN based classifier vs the corresponding GAN attackers.

First we generate adversarial examples from the MalGAN, LSGAN and CTGAN and denote them as  $\mathcal{M}$ ,  $\mathcal{L}$  and  $\mathcal{C}$  respectively.

**Table 4.4**

Model	Combined synthetic data		
	$\mathcal{M} + \mathcal{L}$	$\mathcal{L} + \mathcal{C}$	$\mathcal{M} + \mathcal{C}$
Untrained model (SVM)	0.84	0.0	0.11
Untrained model (RF)	0.37	0.003	0.062
Untrained model (DNN)	0.52	0.081	0.087
MalGAN (SVM)	24.29	18.74	24.77
LSGAN-AT (SVM)	31.86	28.61	27.08
CTGAN (SVM)	50.36	48.55	52.54
MalGAN (RF)	28.03	18.21	22.94
LSGAN-AT (RF)	45.82	51.17	29.08
CTGAN (RF)	62.48	56.26	58.37
MalGAN (DNN)	42.15	40.75	32.58
LSGAN-AT (DNN)	30.08	36.22	28.57
CTGAN (DNN)	61.39	49.56	54.87

For example  $\mathcal{M} + \mathcal{L}$  denotes combined adversarial data from MalGAN and LSGAN both. We measure the accuracy in percentage of each model for classifiers SVM, RF and DNN. Though we combine the generated samples in all possible ways, our CTGAN trained (Adversarial trained) malware significantly performs well compared to other GAN based model, especially LSGAN-AT.

## Discussions

Interpreting the results obtained from our study sheds light on their implications for Android malware detection in real-world scenarios. The superior performance demonstrated by the model trained with adversarial training using CTGAN-generated samples underscores the effectiveness of this approach in bolstering the model's robustness against evolving malware threats. These findings suggest that incorporating

adversarial training strategies can significantly enhance the resilience and effectiveness of machine learning-based malware detection systems in real-world Android environments (10). By leveraging adversarial training, such systems can better adapt to the dynamic nature of malware and mitigate the risks posed by sophisticated adversarial attacks. Consequently, our results advocate for the adoption of adversarial training techniques as a key strategy for enhancing the security and reliability of Android malware detection systems in practical settings.

The significant improvement in model performance observed with adversarial training using CTGAN-generated samples, particularly in the context of tabular datasets containing Android application samples, can be attributed to several key factors. Firstly, CTGAN's compatibility with tabular data formats allows it to directly generate synthetic samples that closely mirror the intricate distribution of malware and benign instances within the dataset. This tailored approach ensures that the synthetic data generated by CTGAN aligns precisely with the characteristics and features present in the dataset, thereby enhancing the model's ability to generalize and detect subtle malware patterns effectively. Additionally, CTGAN's adversarial training strategy exposes the model to challenging adversarial examples, fostering resilience against evasion tactics employed by sophisticated malware variants, while also mitigating the impact of class imbalance inherent in the dataset. In contrast, other GAN variants such as MalGAN, LSGAN, infoGAN, and WGAN may struggle to accommodate the specific features and characteristics of tabular data, limiting their ability to generate realistic synthetic samples and effectively augment the training dataset. Thus, CTGAN's seamless integration with tabular data formats, coupled with its robust adversarial training strategy, positions it as a superior choice for malware detection tasks, offering enhanced flexibility, performance, and resilience compared to other GAN variants.

# Chapter 5

## Conclusion

The study highlights the effectiveness of adversarial training with CTGAN-generated samples in bolstering Android malware detection. Through comprehensive experimentation, it was demonstrated that CTGAN, coupled with adversarial training, significantly improves model performance by enhancing its resilience against sophisticated malware variants. Notably, CTGAN's ability to generate high-fidelity synthetic samples, closely resembling real-world data, ensures that the model is well-equipped to detect subtle malware patterns. The integration of adversarial training further fortifies the model's robustness, enabling it to withstand evasion tactics commonly employed by malicious software. This approach holds significant promise for strengthening cybersecurity defenses against evolving malware threats, offering a proactive strategy to mitigate risks in real-world Android environments. By leveraging CTGAN-generated samples and adversarial training techniques, organizations can enhance their ability to detect and combat emerging malware threats effectively, safeguarding against potential security breaches and protecting user data and privacy.

# References

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, p. 139–144, oct 2020. [Online]. Available: <https://doi.org/10.1145/3422622>
- [2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” 2014.
- [3] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak, “A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection,” *IEEE Access*, 2023.
- [4] W. Hu and Y. Tan, “Generating adversarial malware examples for black-box attacks based on gan,” in *International Conference on Data Mining and Big Data*. Springer, 2022, pp. 409–423.
- [5] J. Yuan, S. Zhou, L. Lin, F. Wang, and J. Cui, “Black-box adversarial attacks against deep learning based malware binaries detection with gan,” pp. 2536–2542, 2020.
- [6] J. Wang, X. Chang, J. Mišić, V. B. Mišić, Y. Wang, and J. Zhang, “Mal-lsgan: An effective adversarial malware example generation model,” in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

- 
- [7] J. Wang, X. Chang, Y. Wang, R. J. Rodríguez, and J. Zhang, “Lsgan-at: enhancing malware detector robustness against adversarial examples,” *Cybersecurity*, vol. 4, pp. 1–15, 2021.
- [8] E. D. Source, “Virusshare dataset,” 2021.
- [9] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [10] R. Yadav and R. S. Bhadoria, “Performance analysis for android runtime environment,” in *2015 Fifth International Conference on Communication Systems and Network Technologies*. IEEE, 2015, pp. 1076–1079.