

# MalDataGen: A Modular Framework for Synthetic Tabular Data Generation in Malware Detection

Kayuã Oleques Paim<sup>1</sup> , Angelo Gaspar Diniz Nogueira<sup>2</sup>   
 Diego Kreutz<sup>2</sup> , Weverton Cordeiro<sup>1</sup> , Rodrigo Brandão Mansilha<sup>2</sup> 

<sup>1</sup> Universidade Federal do Rio Grande do Sul (UFRGS)

<sup>2</sup>Universidade Federal do Pampa (UNIPAMPA)

`kopaim@inf.ufrgs.br, angelonogueira.aluno@unipampa.edu.br,`  
`diegokreutz@unipampa.edu.br, weverton.cordeiro@inf.ufrgs.br`  
`mansilha@unipampa.edu.br`

**Abstract.** High-quality data scarcity hinders malware detection, limiting ML performance. We introduce *MalDataGen*, an open-source modular framework for generating high-fidelity synthetic tabular data using modular deep learning models (e.g., WGAN-GP, VQ-VAE). Evaluated via dual validation (TR-TS/TS-TR), seven classifiers, and utility metrics, *MalDataGen* outperforms benchmarks like SDV while preserving data utility. Its flexible design enables seamless integration into detection pipelines, offering a practical solution for cybersecurity applications.

## 1. Introduction

Modern machine learning algorithms, particularly deep learning architectures, depend on large-scale datasets with reliable annotations to achieve optimal performance. However, current methods for dataset collection and labeling require substantial resources and time investments [Zha et al. 2025]. The field faces persistent challenges with data availability, as approximately 80% of AI project failures stem from insufficient data quantity or quality [AI & Data Today 2023].

These challenges are especially observable in domains with limited or imperfect data sources. Synthetic data generation offers a potential solution [Figueira and Vaz 2022, Lee 2025, Hao et al. 2024], creating artificial samples that maintain key characteristics of real-world data. In cybersecurity, this approach has been applied to improve malware detection systems [Nogueira et al. 2024b, Nogueira et al. 2024a, Fernandes et al. 2025, Casola et al. 2023, Pepes et al. 2023], identify anomalous network traffic [Kumar and Sinha 2023], and generate polymorphic malware variants [Dunmore et al. 2023].

We observe the development of several libraries and frameworks that help streamline and standardize the process for synthetic data generation. We provide a comparative overview of libraries for synthetic tabular data generation in Table 1. The table highlights the models and algorithms they use for generating synthetic data.

<sup>1</sup><https://gretel.ai/>

<sup>2</sup><https://ydata.ai/>

<sup>3</sup><https://docs.sdv.dev/sdv>

<sup>4</sup><https://github.com/SBSEG25/MalDataGen>

**Table 1. Deep Learning-Based Libraries for Synthetic Tabular Data Generation.**

Library	Algorithms	Models
<b>Gretel Synthetics</b> <sup>1</sup>	GAN	DGAN, DPGAN, ACTGAN
<b>YData</b> <sup>2</sup>	GAN	GAN, cGAN, WGAN, WGAN-GP, DRAGAN, CramerGAN, CWGAN-GP, CTGAN
<b>SDV</b> <sup>3</sup>	Statistical	Copula
	GAN	CTGAN
	AE	TVAE
<b>MalDataGen</b> <sup>4</sup> <b>(This Work)</b>	GAN	GAN, WGAN-GP, cGAN
	AE	AE, VAE, VQ-VAE
	Diffusion	LDM

Existing libraries face two main constraints: limited flexibility for custom modifications and a narrow range of pre-implemented algorithms. We address these issues with a new Python-based modular and extensible framework, named *MalDataGen*<sup>4</sup>, with broader algorithms support. While Table 1 shows most current tools focus on GAN-based approaches [Mirza and Osindero 2014], and SDV includes Autoencoders [Kingma et al. 2013], our solution expands on these foundations. We incorporate additional implementations of established methods and present what we believe are the first tabular data applications of VQ-VAEs [Van Den Oord et al. 2017] and Latent Diffusion Models [Rombach et al. 2022]. Our solution is also designed to be composable, enabling scientists and practitioners to assemble other models from our basic components.

Our evaluation shows that *MalDataGen* outperforms SDV, which currently offers the widest range of algorithms. We assess all generative models from both libraries using seven classifiers and follow the methodology from [Esteban et al. 2017], implementing two validation approaches: TR-TS (Train on Real - Test on Synthetic) and TS-TR (Train on Synthetic - Test on Real).

The paper is organized as follows. Section 2 describes our framework. Section 3 presents and discusses the results, with concluding remarks in Section 4.

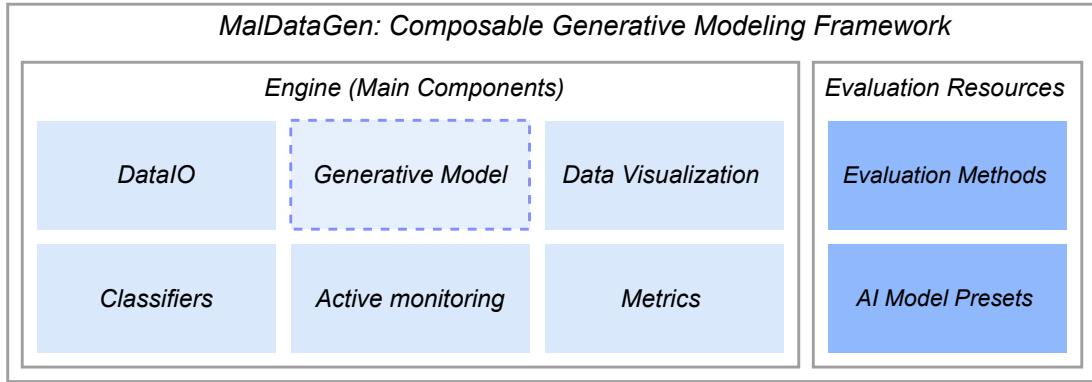
## 2. *MalDataGen*: Composable Generative Modeling Framework

Figure 1 shows the architecture of *MalDataGen*. The design includes two core components: (1) the *Engine* with components for developing and managing deep learning-based generative models, and (2) the *Evaluation Resources* for validating synthetic data quality.

### 2.1. Engine

The *Engine* contains modules for development, training, and orchestration of deep learning-based generative models. We organize the library into six key modules: *DataIO*, *Data Visualization*, *Classifiers*, *Metrics*, *Active Monitoring*, and *Generative Models*. Additionally, the Engine includes auxiliary modules, such as those for handling arguments and exceptions. This structure helps manage complexity while allowing customization and reuse.

We built the *DataIO module* to handle data acquisition, transformation, and storage. It works with structured formats like *.csv* and *.xls*, supporting dataset normaliza-



**Figure 1. Overview of the MalDataGen composable framework.**

tion, schema-aware editing, and format-preserving serialization. The pluggable I/O layer adapts to new file formats and domain-specific representations.

The *Data Visualization module* offers tools for analyzing both real and synthetic datasets. Its capabilities include visualization of statistical dependencies through correlation heatmaps, cluster analysis using techniques like K-Means with UMAP, and graphical representation of performance metrics such as confusion matrices and bar plots. These components support customization of visual styles and export options.

Our *Classifiers module* contains 15 supervised learning algorithms, including SVM, Random Forest, and MLP. The abstract interface design enables efficient comparison across different modeling approaches.

In the *Metrics module*, we implement measures for predictive performance (F1-score, AUC, MCC) and distribution similarity (Jensen-Shannon Divergence, Wasserstein Distance, MMD). These help evaluate how well synthetic data matches real data distributions.

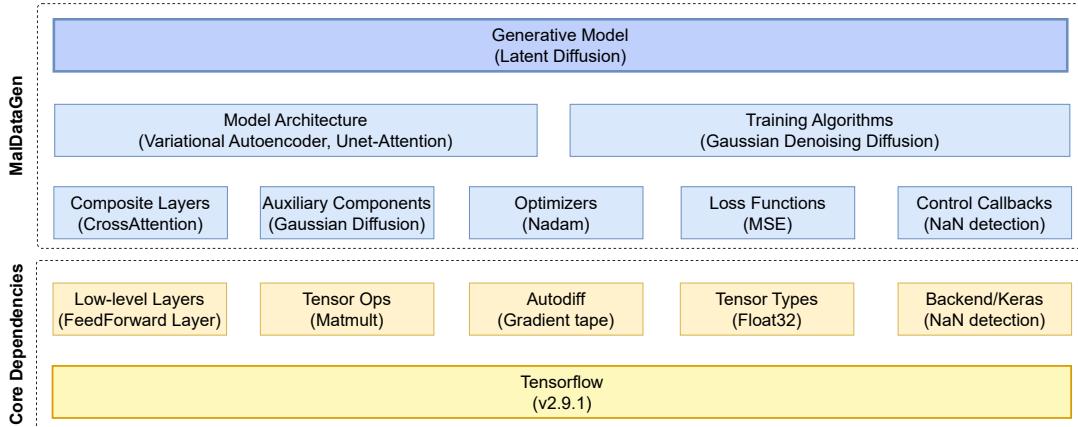
The *Active Monitoring module* oversees the entire data pipeline. It detects faults like NaN values and divergence, tracks resources including memory usage, and maintains time-stamped logs. The system includes convergence-aware callbacks for runtime adjustments.

For the *Generative Models* module, we provide a suite of fully configurable synthetic-data generators that allow architecture customization, hyperparameter tuning, and the selection of diverse generation strategies. Out-of-the-box, the module supports CTGAN, variational autoencoders (VAEs), and Gaussian copula models, each capable of producing samples that faithfully preserve the statistical properties of the original dataset.

Crucially, we have extended each base implementation to better suit Android-malware generation. In our GAN and VAE variants, we introduce an embedding layer for malware class labels (benign vs. malicious) and a projection layer to streamline feature preprocessing. For the latent diffusion model (LDM), we depart from the original TabDDPM design [Kotelnikov et al. 2023] by integrating a VAE subnetwork: this ensures a continuous, lower-dimensional latent space. Finally, during the reverse-diffusion (denoising) stage, we employ a bespoke U-Net composed of multi-channel feedforward layers and residual temporal-encoding blocks, and train it end-to-end with a combined

KL-divergence plus MSE loss.

The *Generative Models module* follows a two-layer architecture shown in Figure 2. We organize the upper layer with functional building blocks, while the lower layer handles core computational dependencies including TensorFlow and essential Python libraries.



**Figure 2. Overview of the *Generative Models* architecture (with illustrative examples in parentheses).**

For the functional building blocks, we implement reusable components that support flexible pipeline construction. The *optimizers* includes gradient-based algorithms like Adam, Nadam, and SGD, which we can configure for different learning scenarios. We provide various *loss functions* such as MSE, Cross-Entropy, and KL Divergence to handle different training objectives. The *composite layers* incorporates abstractions like Time Embedding and Sampling Layers that work with architectures such as VQ-VAEs. Additional *auxiliary components* includes Gaussian diffusion processes and attention mechanisms. For training supervision, we implement *control callbacks* that handles early stopping and resource monitoring.

The training architecture separates learning dynamics from model structure. We implement multiple *training algorithms* including GANs, VAEs, and Diffusion Models that work with different *model architectures* like UNet and Encoder-Decoder pairs. This separation allows combining architectures with different training objectives without structural modifications. For example, a UNet backbone can work with either VAE objectives or DDPM training.

At the foundation, we build on TensorFlow's computational framework<sup>1</sup> for core operations. The *low-level layers* provide basic neural network components including convolutional and dense layers. We implement essential *tensor operations* like matrix multiplication and other algebraic routines. The automatic differentiation system enables gradient computation through reverse-mode differentiation. For tensor management, we include shape and data type specifications through *tensor typing*. The system integrates with Keras<sup>2</sup> for execution management and high-level API access.

<sup>1</sup><https://www.tensorflow.org>

<sup>2</sup><https://keras.io/api/>

This architecture combines TensorFlow’s computational capabilities with modular components for generative modeling. The layered approach maintains performance while supporting flexible configuration and extension of the core functionality. Each component interacts through well-defined interfaces, allowing researchers to modify or extend specific aspects without affecting other system parts.

We augment our discussion with a comprehensive set of visuals. First, five in-depth figures unpack the internal workings of each generative model, clearly illustrating their key architectural distinctions. Complementing these, eight Mermaid diagrams chart the entirety of the MalDataGen framework – depicting everything from the high-level system layout and object-oriented class relationships to the complete data-processing pipeline, evaluation routines, training workflows, and metrics architecture. Together, these graphical resources present a unified, detailed perspective on how MalDataGen’s modules interlock to produce and rigorously assess high-quality synthetic data for cybersecurity applications. For the full collection of diagrams and detailed explanations, please visit our GitHub repository<sup>3</sup>.

## 2.2. Evaluation Resources

Our *evaluation resources* provides systematic protocols for assessing synthetic data quality and usefulness. These resources consist of two core components: the *Evaluation Methods* and *AI Model Presets* modules.

The *Evaluation Methods* component implements validation approaches including  $k$ -fold cross-validation, where we partition datasets into  $k$  subsets for iterative training and testing. We also employ domain transfer evaluation through two complementary approaches: *Train-on-Real/Test-on-Synthetic* (TR-TS) and *Train-on-Synthetic/Test-on-Real* (TS-TR). These methods allow us to assess both data transferability and model robustness using paired classifiers from our classification tools.

For the *AI Model Presets*, we maintain version-controlled configurations containing optimized parameters such as learning rates for GAN components and training epoch counts. The module includes architectural templates for supported network types and initialization parameters for models from our generative components. These presets help ensure consistent evaluation across different experimental setups.

The resources incorporate standardized metrics to compare synthetic and original datasets across multiple dimensions. All components integrate with other library modules through defined interfaces, supporting reproducible assessment of synthetic data quality. We designed the system to balance comprehensive evaluation with practical usability, allowing researchers to focus on their specific validation needs.

## 3. Results and Discussion

We present and analyze our key findings using the SVM (Support Vector Machine) classifier to demonstrate patterns observed across classifiers. Complete results for all classifiers and hyperparameter configurations are available in our public repository.

For our experiments, we employed the Androcrawl dataset from the Malware-DataHunter Project’s public repository<sup>4</sup>. This dataset comprises 20,340 samples—10,170

---

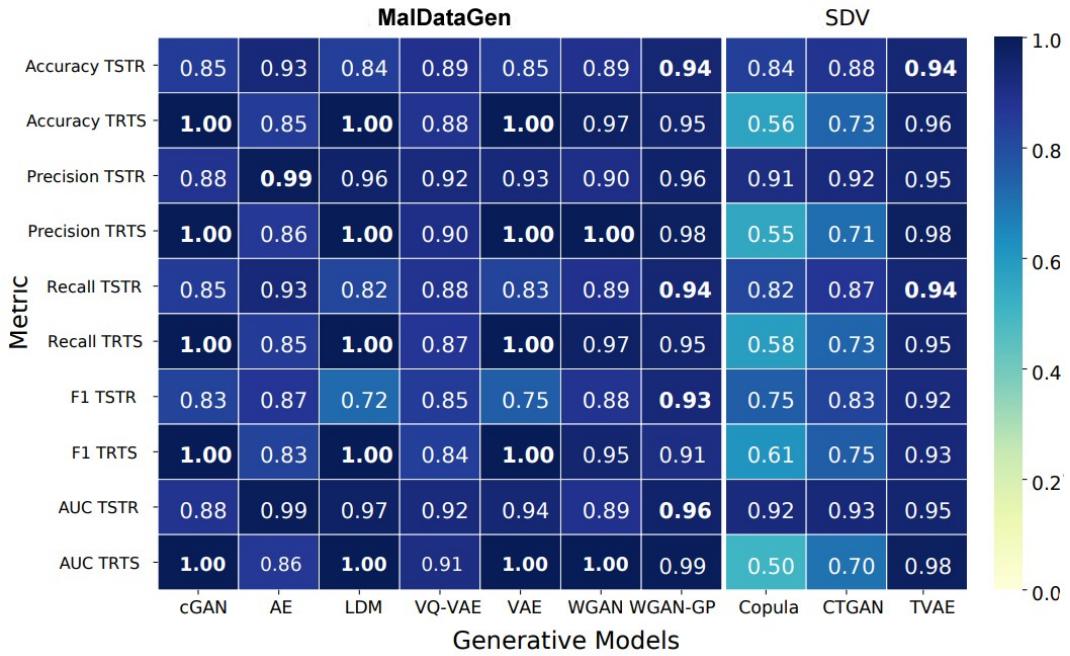
<sup>3</sup><https://github.com/SBSEG25/MalDataGen>

<sup>4</sup><https://github.com/Malware-Hunter/datasets.git>

malware and 10,170 benign—each described by 136 features. Prior to training, we applied chi-square feature selection to retain the top 200 features, then down-sampled each class to 10,000 instances (20,000 total) where needed to ensure balance.

We leave a comprehensive examination of synthetic data’s influence on class balance—and a rigorous investigation into potential data leakage stemming from our feature-selection process—for subsequent studies.

Figure 3 presents a heatmap of average 5-fold cross-validation scores for SVM classifiers trained on synthetic samples generated by both our MalDataGen framework and the SDV library. Darker cells indicate stronger performance (closer to 1.0) and lighter cells indicate weaker performance (closer to 0.0). We report utility metrics—accuracy, precision, recall, F1-score, and AUC—with paired rows contrasting the TS-TR and TR-TS evaluation scenarios.



**Figure 3. Utility assessment: Binary classification metrics for SVM classifier performance using data generated by different models.**

Our results reveal notable performance variations across generative models. The WGAN-GP and WGAN implementations show strong performance across all metrics, often achieving near-perfect scores in both evaluation scenarios. CGAN and AE follow closely, with CGAN particularly excelling in TR-TS metrics. VAE and LDM maintain high performance but show slightly lower TS-TR F1-scores due to reduced recall. Our VQ-VAE implementation, while the lowest-performing among our models, maintains all metrics above 0.84.

SDV’s models demonstrate weaker TR-TS performance. The Copula model shows limited efficacy, with critical TR-TS metrics approaching random classification

levels. CTGAN shows improved TR-TS capability over Copula but remains behind our models. SDV’s TVAE performs exceptionally, matching our top-performing WGAN-GP across both evaluation paradigms.

These results suggest that with proper hyperparameter optimization, our models can match or surpass existing open-source implementations like SDV. The performance differences highlight the importance of model selection and tuning for synthetic data generation tasks.

We present the mean distance metric results from our 5-fold evaluation in Table 2. The table shows distance measurements between synthetic and real data, where lower values indicate higher fidelity. Each column represents a different generative model from our framework and the SDV library.

**Table 2. Fidelity assessment: Distance metrics comparing real data to synthetic outputs across generative models (lower values indicate better alignment).**

Metrics	Models								SDV		
	Ours										
	cGAN	AE	LDM	VQ-VAE	VAE	WGAN	WGAN-GP	Copula	CTGAN	TVAE	
Euclidean Distance ↓	3.59	3.97	3.46	4.62	3.46	<b>3.45</b>	3.89	4.43	4.51	4.29	
Hellinger Distance ↓	166.08	181.03	160.57	210.20	160.88	<b>159.50</b>	179.00	200.90	204.74	195.28	
Manhattan Distance ↓	405.7	483.39	379.17	650.19	380.63	<b>374.16</b>	471.21	593.54	616.59	560.79	
Hamming Distance ↓	2.98	3.55	2.79	4.78	2.80	<b>2.75</b>	3.46	4.36	4.53	4.12	
Jaccard Distance ↓	0.65	0.66	<b>0.60</b>	0.74	<b>0.60</b>	0.61	0.66	0.70	0.71	0.69	

The results show a consistent pattern with our utility metric findings. Our WGAN implementation achieves the lowest distance measures across most metrics, indicating the closest similarity to real data. The WGAN-GP, cGAN, and AE models follow with slightly higher but still competitive distance values. Among our implementations, VQ VAE shows the largest distance from real data distributions.

For SDV’s models, we observe generally higher distance metrics, with Copula and CTGAN showing the greatest dissimilarity to real data. This aligns with their weaker performance in utility evaluations. SDV’s TVAE stands as an exception, matching our WGAN-GP in both fidelity and utility metrics, demonstrating comparable effectiveness in generating authentic synthetic data.

The fidelity assessment reinforces the utility metric results, showing that models producing data closer to the real distribution also perform better in downstream tasks. The consistent performance across both evaluation dimensions suggests that careful model selection and optimization can yield synthetic data that preserves both statistical properties and practical usefulness.

## 4. Final Considerations

**Demonstration.** We will demonstrate how MalDataGen operates through a practical example executed on one of our devices. This demonstration will highlight its configuration parameters, execution pipeline and an analysis of the produced such as heatmaps, confusion matrices, and training curves. Examples of these outputs are presented in Appendix A.

**Conclusion.** We presented *MalDataGen*, a *Composable Generative Modeling Framework* designed for the generation of synthetic tabular data, with a specific focus on cybersecurity applications. Our implementation showed comparable or superior results to *SDV* in the evaluated scenarios. It also introduced a modular architecture that supports extensibility, a set of pre-configured generative models, and a methodology based on two validation strategies (TR-TS and TS-TR), which helped assess the quality of the generated data. We used visualization methods to support exploratory analysis and to examine the qualitative characteristics of the synthetic data.

**Future directions.** We plan to improve the library by adding new generative models, classifiers, and metrics. We also expect to integrate it with tools for data analysis and generation to increase interoperability. Our evaluation resources will be expanded to include other libraries, such as *YData* and *GretelSynthetics*, along with additional datasets and use cases.

**Acknowledgements.** This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and Grant #88887.954253/2024-00 (INCT ICoNIoT). It was also financed in part by Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), grants nr. 22/2551-0000841-0, 24/2551-0001368-7, and 24/2551-0000726-1. It was also financed in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - Grant #314506/2023-3 and #405940/2022-0 (INCT ICoNIoT). It was also financed in part by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - Grant #2020/05183-0, and #2023/00816-2.

## References

AI & Data Today (2023). Top 10 reasons why ai projects fail. <https://t.ly/wMBj5>.

Casola, K., Paim, K., Mansilha, R., and Kreutz, D. (2023). Droidaugmentor: uma ferramenta de treinamento e avaliação de gans para geração de dados sintéticos. In *Anais Estendidos do XXIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 57–64, Porto Alegre, RS, Brasil. SBC.

Dunmore, A., Jang-Jaccard, J., Sabrina, F., and Kwak, J. (2023). A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection. *IEEE Access*, 11:76071–76094.

Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*.

Fernandes, K. C., Nogueira, A. G. D., da Silva, A. L. G., Paim, K. O., Kreutz, D., (UNIPAMPA), R. M., and de Souza Bragança, H. L. (2025). *Uma Introdução sobre Redes Adversárias Generativas (GANs) e suas Aplicações na Cibersegurança*, chapter 2. EDIURCAMP. Acessado em 2025-05-04.

Figueira, A. and Vaz, B. (2022). Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15):2733.

Hao, S., Han, W., Jiang, T., Li, Y., Wu, H., Zhong, C., Zhou, Z., and Tang, H. (2024). Synthetic data in ai: Challenges, applications, and ethical implications. *arXiv preprint arXiv:2401.01629*.

Kingma, D. P., Welling, M., et al. (2013). Auto\$-encoding variational bayes.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. (2023). Tabddpm: Modelling tabular data with diffusion models. In *ICML*. PMLR.

Kumar, V. and Sinha, D. (2023). Synthetic attack data generation model applying generative adversarial network for intrusion detection. *Computers & Security*.

Lee, P. (2025). Synthetic data and the future of ai. *Cornell L. Rev.*, 110:1.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Nogueira, A., Paim, K., Bragança, H., Mansilha, R., and Kreutz, D. (2024a). Geração de dados sintéticos tabulares para detecção de malware android: um estudo de caso. In *Anais do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 808–814, Porto Alegre, RS, Brasil. SBC.

Nogueira, A., Paim, K., Bragança, H., Mansilha, R., and Kreutz, D. (2024b). Malsyngen: redes neurais artificiais na geração de dados tabulares sintéticos para detecção de malware. In *Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 129–136, Porto Alegre, RS, Brasil. SBC.

Peppes, N., Alexakis, T., Daskalakis, E., Demestichas, K., and Adamopoulou, E. (2023). Malware image generation and detection method using dcgans and transfer learning. *IEEE Access*, 11:105872–105884.

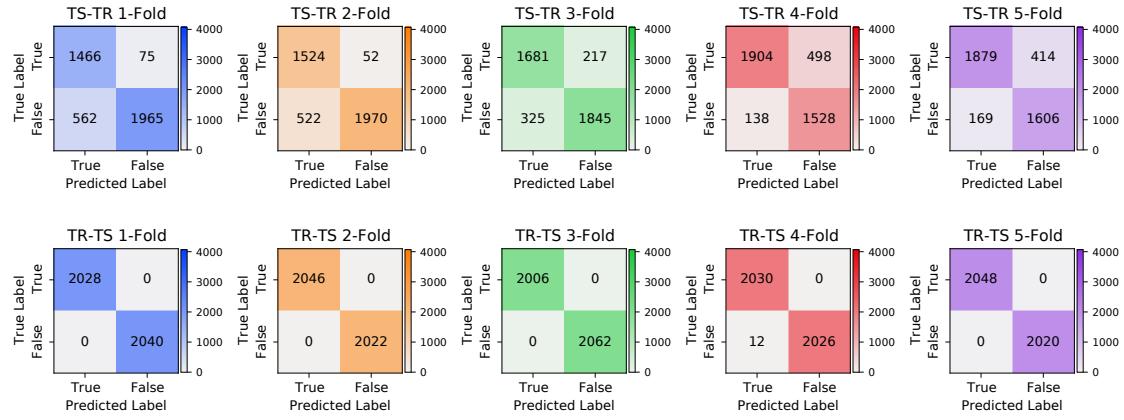
Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models . In *IEEE/CVF CVPR*.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

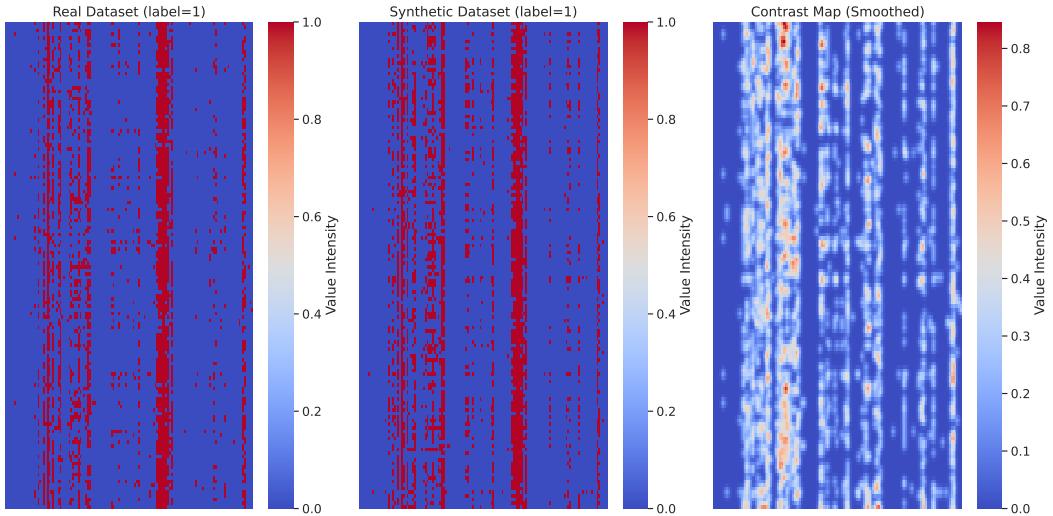
Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., Jiang, Z., Zhong, S., and Hu, X. (2025). Data-centric artificial intelligence: A survey. *ACM Computing Surveys*, 57(5).

## Appendix A: Examples of outputs

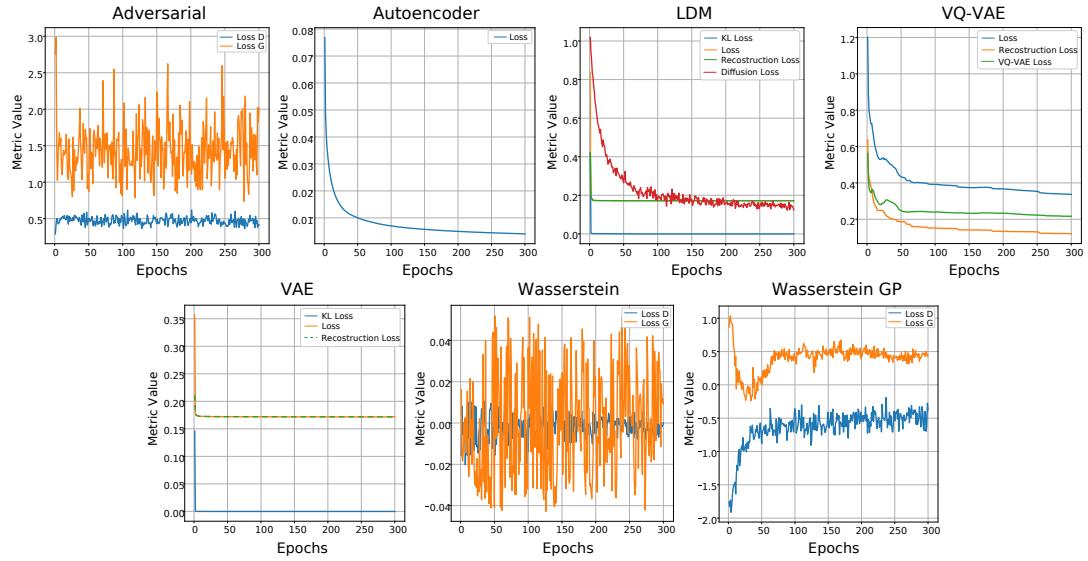
Figures 4, 5 and 6 encapsulate our evaluation pipeline’s key results. The aggregated confusion matrices in Figure 4 summarize binary classification performance (TP, FN, TN, FP) under both TR-TS and TS-TR protocols. Figure 5 compares real versus synthetic feature means, with an overlaid plot highlighting divergences to assess generation fidelity. Lastly, Figure 6 traces each model’s training stability: WGAN/WGAN-GP generator and discriminator losses, VQ-VAE’s total, reconstruction and quantization losses, and a single reconstruction curve for the autoencoder. Collectively, these visuals demonstrate the synthetic data’s quality and the robustness of our training workflows.



**Figure 4. Evaluating of Adversarial model via SVM Confusion Matrices.**



**Figure 5. Comparative heat map.**



**Figure 6. Training loss curves of the models.**