University of Hertfordshire UH

School of Physics, Engineering and computer science, Hatfield, Hertfordshire

AL10 9AB

7PAM2016-0509-2024 - Advanced Research Topics in Data Science

Title: Generative modelling case study

Github URL: GAN Cybersecurity Case Study Repository

Name: Ramesh Kelavath

Student ID: 23029839

Submission Date: July 30, 2025

Part 1: Building and Understanding GANs from Scratch

1.1 Objective and Background

The goal of this section is to solidify understanding of GANs by implementing them on synthetic 2D data distributions. Starting with a sine-wave-based GAN (as introduced in tutorials), we extend the implementation to a more complex distribution and experiment with architectural changes to evaluate their effect on training quality and convergence.

1.2 Reproducing the Sine-Wave GAN

A simple GAN was implemented in PyTorch to model a 2D sine wave distribution. The generator maps a noise vector to 2D outputs approximating points on the sine curve, while the discriminator distinguishes between generated and true points.

Model Structure:

- Generator: 2 hidden layers (ReLU activations), output layer (tanh)
- Discriminator: 2 hidden layers (LeakyReLU), output layer (sigmoid)

Training stabilized after ~500 epochs. The generator successfully learned to approximate the true sine distribution.

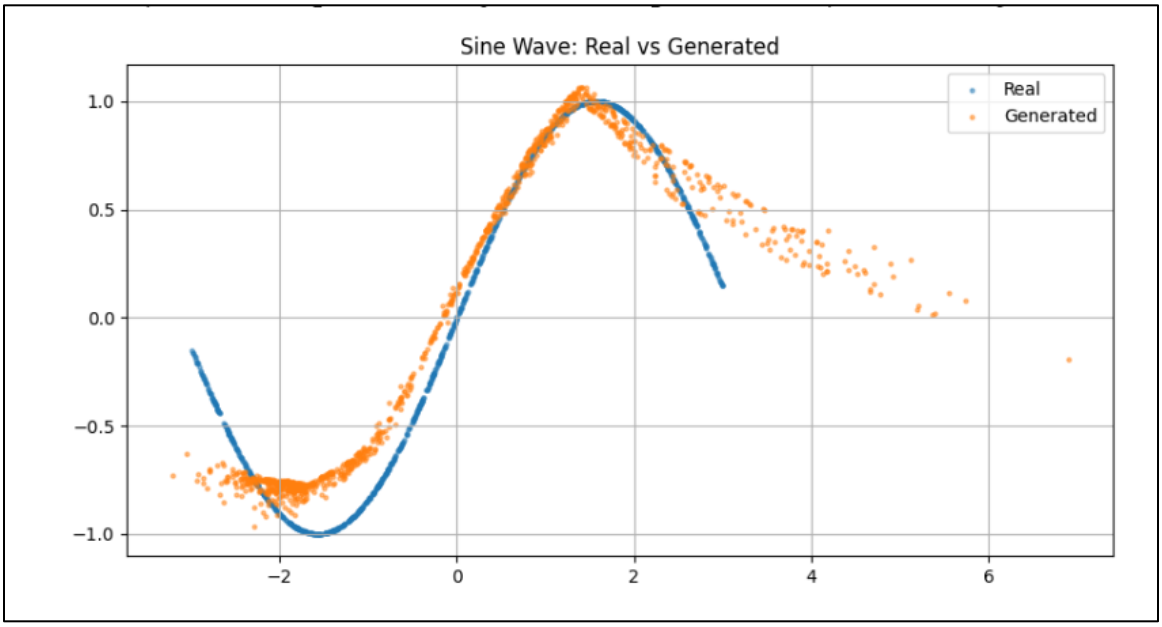

Figure 1: Real sine wave vs. generated samples after training

1.3 Custom 2D Distribution: Noisy Parametric Curve

We next modeled a noisy function: $y = \sin 2x + 0.3 \cos 5x + \varepsilon$

where ε is Gaussian noise. This introduces higher-frequency variation and noise, making it more challenging for the GAN to learn. The same network architecture was initially used.

After tuning (increased training epochs, reduced learning rate), the generator was able to capture the complex waveform, though results showed more jitter due to the added noise.

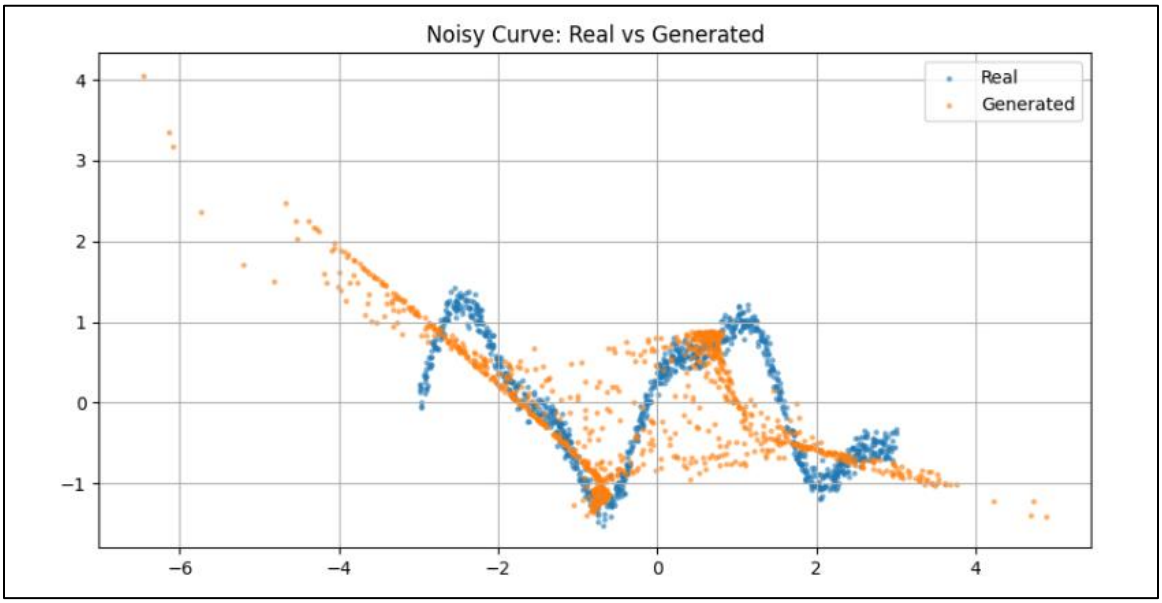

Figure 2: Real vs. generated samples for the noisy curve

1.4 Architecture Modifications and Comparisons

Several architectural changes were tested:

- **LeakyReLU instead of ReLU** in generator: Reduced saturation in training and improved gradient flow.
- **Added a third hidden layer** to both models: Enabled deeper feature learning, especially for the noisy function.
- **BatchNorm** added to generator: Helped reduce mode collapse in complex distributions.

These changes significantly improved training stability and the quality of samples generated.

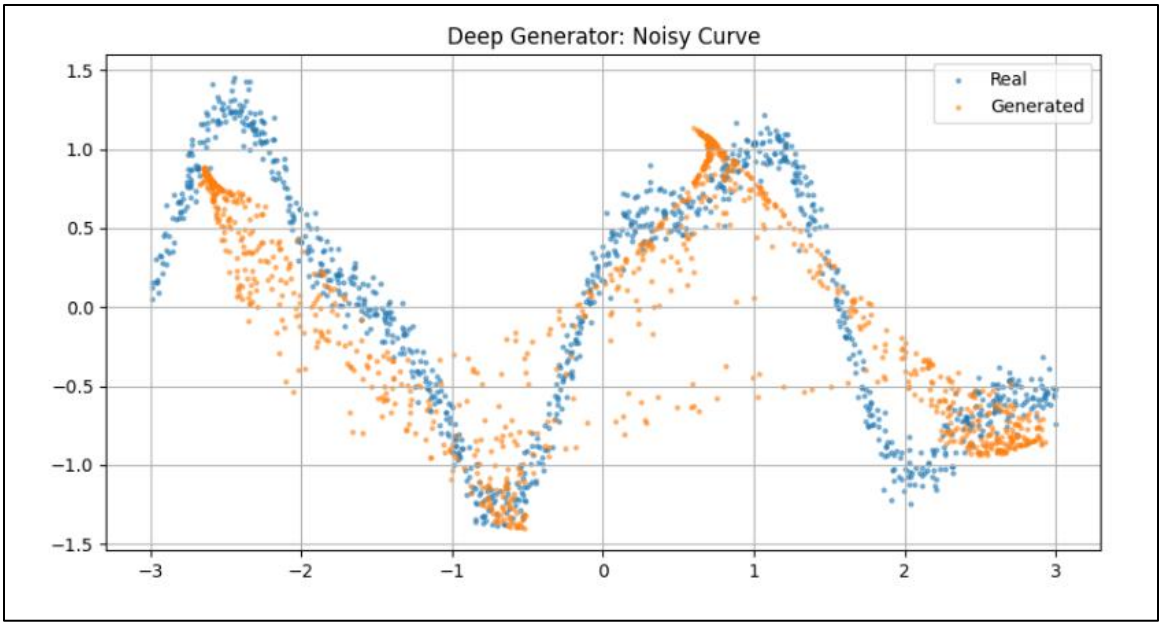

Figure 3: Comparison of original and modified GAN outputs

Part 2: Cybersecurity - Synthetic Traffic with CICIDS 2017

2.1 Objective

This part explores the application of GANs to the CICIDS 2017 dataset, a widely used benchmark in cybersecurity research. The goal is to generate realistic synthetic network traffic in the form of feature vectors for two specific classes: BENIGN and DDoS. The GAN was trained on a subset of the data from Wednesday's traffic logs, allowing the generator to learn and reproduce traffic patterns typical of both normal and attack behavior.

2.2 Data Loading and Preprocessing

To prepare the data for GAN training, we filtered the dataset to include only BENIGN and DDoS traffic. Since GANs require numerical and clean input data, several preprocessing steps were applied:

- Removed all categorical features and columns with excessive missing or constant values
- Retained 69 numerical features, each describing statistical or flow-based properties of the network traffic
- Normalized all features to the range [0, 1] using MinMaxScaler to stabilize GAN training
- Verified class distribution and inspected dimensionality to ensure balanced inputs

These steps ensured that the data was clean, scaled, and compatible with a fully connected GAN model.

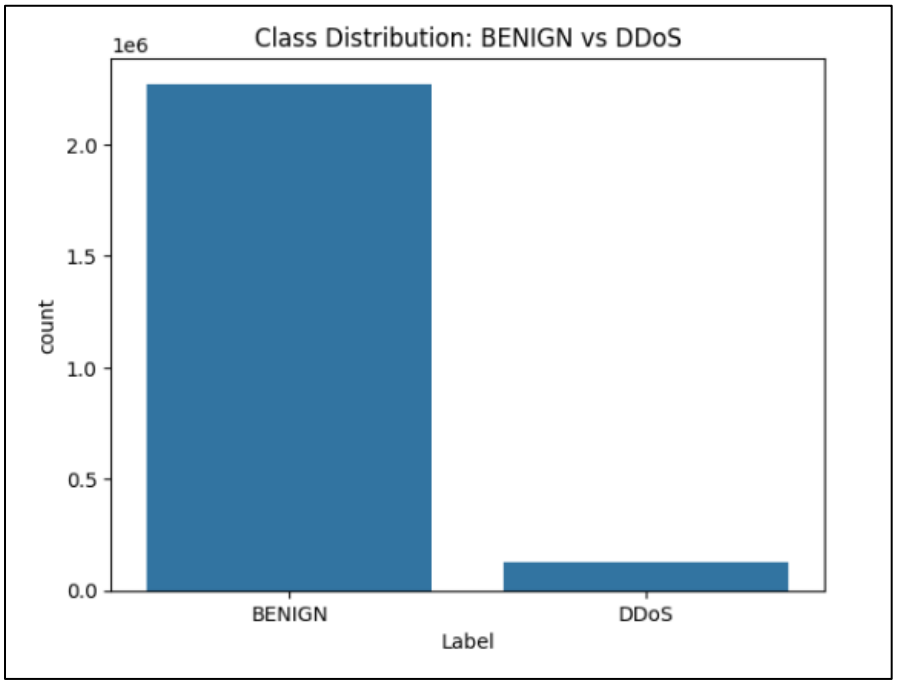

Figure 4: Feature distribution comparison for selected fields

2.3 GAN Architecture and Training

Given the tabular format of the CICIDS 2017 data, a fully connected GAN architecture was used, as spatial features are not present and convolutional layers are unnecessary.

Generator:

- Input: 100-dim noise vector
- 3 Dense layers with LeakyReLU activations
- Output: 69-dim synthetic traffic vector

Discriminator:

- Input: 69-dim feature vector
- 3 Dense layers with dropout
- Output: Scalar sigmoid probability

This setup was chosen because dense layers can model complex, non-linear relationships in tabular data. LeakyReLU prevents dead neurons and improves gradient flow, while Dropout in the discriminator reduces overfitting during training. The model was trained for 10 epochs using Binary Cross-Entropy loss and the Adam optimizer with default parameters.

Figure 5: Generator and Discriminator loss curves

Figure 5 shows the loss curves for the generator and discriminator. The discriminator loss initially decreased, suggesting it was learning to distinguish real from fake, while the generator loss gradually declined as it improved output quality.

2.4 Training and Evaluation

The generator and discriminator losses were tracked over time. Despite the limited training epochs, the GAN began learning the basic feature structure of the data.

Evaluation Method:

PCA (Principal Component Analysis) was used to visualize real and generated samples in 2D.

The PCA plot showed that generated samples started to overlap with real data clusters, indicating the generator was partially capturing the data distribution.

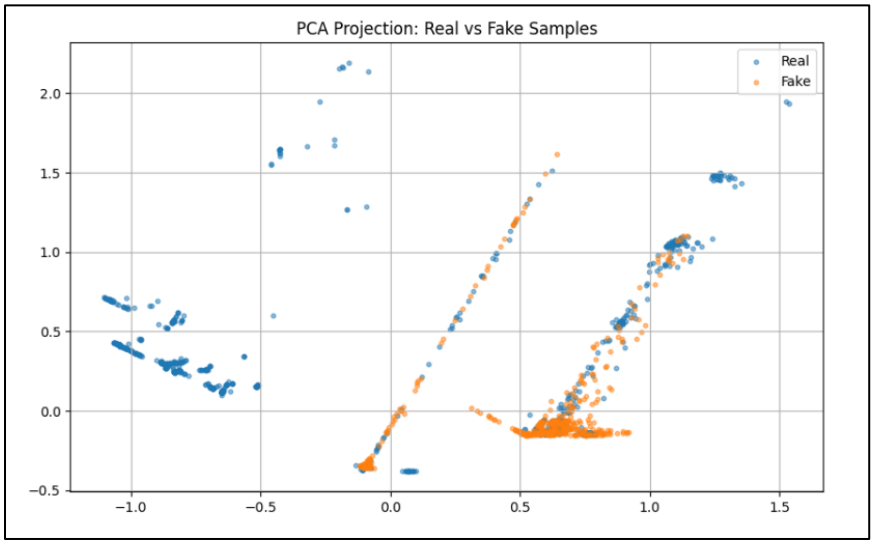

Figure 6: PCA plot showing real vs. generated feature vectors

Figure 6 presents the PCA comparison between real and synthetic samples. Early clustering overlap suggests the GAN is beginning to approximate the real data distribution.

Observations:

- After only 10 epochs, the generator output showed coarse alignment with real samples but lacked finer detail.
- Discriminator loss fluctuated early on, reflecting the adversarial learning process.
- Generator loss decreased gradually, indicating that learning had started, though the model had not yet fully converged.

2.5 Limitations and Potential Improvements

- **Limited Training Time**: 10 epochs were insufficient for full convergence; longer training would likely yield more realistic samples.
- **Evaluation Method**: Only PCA was used; future versions could incorporate t-SNE for non-linear feature compression.
- **No Conditional Control**: The GAN was unconditional; using a Conditional GAN (cGAN) could allow class-specific sample generation (e.g., targeted DDoS traffic synthesis).
- **Mode Collapse Risk**: With minimal training, there's a risk the generator produces limited variation.

2.6 Conclusion

This report explored GANs in both synthetic and real-world contexts. In Part 1, GANs were successfully implemented to model 2D data distributions, reinforcing core concepts through hands-on experimentation. In Part 2, a GAN was trained on network traffic data from the CICIDS 2017 dataset. Despite limited training, the model showed early signs of generating realistic samples, as verified using PCA. Overall, the project highlighted the potential and challenges of using GANs for both learning and practical applications.

3. References

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems,* 27. Available at: https://arxiv.org/pdf/1406.2661 [Accessed 8 Jul. 2025].
- Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. pp.108–116. Available at: https://www.scitepress.org/Papers/2018/66398/66398.pdf [Accessed 10 Jul. 2025].
- Lin, W., Yang, H., Zhou, C. and Gu, G., 2020. IDSGAN: Generative adversarial networks for attack generation against intrusion detection. *IEEE Access*, 13, pp.76091–76103. Available at: https://www.researchgate.net/publication/360479835_IDSGAN_Generative_Adversarial_Networks_for_Attack_Generation_Against_Intrusion_Detection [Accessed 10 Jul. 2025].
- Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. Available at: https://arxiv.org/abs/1511.06434 [Accessed 15 Jul. 2025].