Exploring GAN Variants for Balancing Imbalanced Datasets

Ahmad Al-Hmouz

0228229

Supervised by: Dr. Yousef Sanjalawe

01.Problem statement

Imbalanced datasets present a significant challenge in machine learning, as models tend to be biased toward the majority class. This project investigates the use of Generative Adversarial Networks (GANs) to synthetically augment underrepresented classes and enhance overall classification performance. Specifically, it compares the performance of a Vanilla GAN with more advanced GAN variants across multiple datasets, examining how each architecture attempts to match the minority class distribution using different generative strategies. The aim is to assess the extent to which these generative models can address class imbalance and contribute to more equitable and accurate predictions.

02.Dataset Description and imbalance analysis

This experiment utilizes both image and tabular datasets to better understand the applicability of different GAN architectures across varied data types and imbalance scenarios.

2.1 fashionMNIST

The FashionMNIST dataset, consisting of grayscale images representing 10 clothing categories, is used in a modified, imbalanced form. Specifically, the samples for Class 0 (T-shirt/top) and Class 4 (Coat) were reduced to approximately 2% of the total dataset, simulating a significant class imbalance. This adjustment helps evaluate how well generative models can synthesize realistic minority class examples in the context of image data

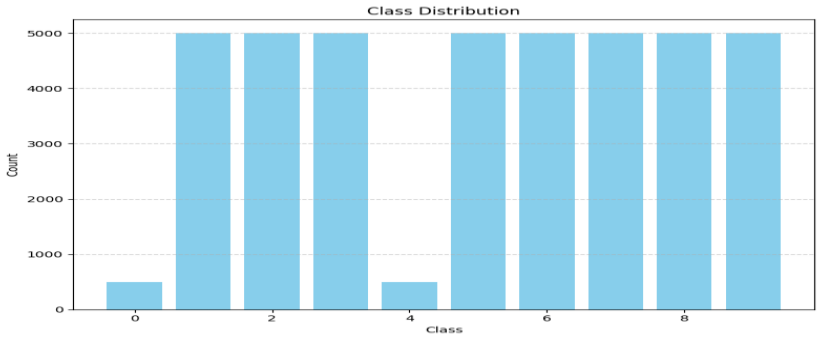

Figure 1: Class distribution in the imbalanced FashionMNIST dataset

2.2 fraud detection dataset

The second dataset is a highly imbalanced tabular dataset for credit card fraud detection, sourced from Kaggle. It contains two classes: legitimate and fraudulent transactions. The fraud class constitutes only 0.173% of the entire dataset, corresponding to an extreme imbalance ratio of approximately 578:1. This dataset is used to assess GAN performance in generating meaningful synthetic tabular data under severe imbalance conditions.

Figure 2: fraud detection dataset

These datasets exhibit significant class imbalance, which leads to biased performance in standard classifiers. Without balancing, classifiers tend to favor the majority class, reducing recall and F1-score for the minority.

03.GAN Architectures and Training

3.1. Vanilla GAN

The Vanilla GAN architecture implemented in this experiment varies slightly depending on the data type—image (FashionMNIST) or tabular (Fraud Detection)—but retains the core design principles of the original GAN framework. For the FashionMNIST dataset, the GAN consists of two fully connected neural networks: a Generator and a Discriminator. The Generator receives a 64-dimensional random noise vector and passes it through two linear layers (256 hidden units) with LeakyReLU activation (slope = 0.2), ultimately outputting a 4096-dimensional vector corresponding to a flattened 64×64 grayscale image. A Tanh activation normalizes the pixel values between -1 and 1. The Discriminator, in turn, processes either real or generated images through two fully connected layers (128 units) with LeakyReLU activation and outputs a real/fake probability via a Sigmoid function. Training used Binary Cross-Entropy Loss (BCELoss) with the Adam optimizer (learning rate = 3e-4), and employed label smoothing—assigning 0.9 to real samples and random values between 0 and 0.1 to fake samples—to stabilize training. Over 100 epochs (batch size = 32), generator progress was monitored using TensorBoard with periodic logging of real and synthetic image grids.

For the Fraud Detection dataset, a simpler, fully connected Vanilla GAN was employed to generate synthetic tabular samples representing fraudulent transactions. The Generator transforms a 32-dimensional noise vector into a feature vector matching the original data dimensionality (e.g., 29 features), using two linear layers with ReLU activations. The Discriminator, similar in design, consists of two linear layers with LeakyReLU followed by a Sigmoid output layer. The model was trained for 100 epochs with a batch size of 64 using the same BCELoss and Adam optimizer setup. Real and synthetic batches were alternately used to train the Discriminator, while the Generator was updated to "fool" the Discriminator into classifying fake data as real. Despite the tabular format, the GAN effectively learned to approximate the minority class distribution, providing synthetic fraud samples that can augment training datasets and improve classifier sensitivity.

3.2. DCGAN

As the advanced GAN variant, Deep Convolutional GAN (DCGAN) was implemented to leverage convolutional architectures for improved image generation. Unlike the fully connected Vanilla GAN, DCGAN utilizes convolutional and transposed convolutional layers, which are more effective for image data. The Discriminator is composed of a sequence of convolutional layers with increasing feature depth, each followed by LeakyReLU activations and batch normalization (except the first layer), concluding with a Sigmoid output. It takes a 64×64 grayscale image as input and reduces it through downsampling to a single scalar output that represents the probability of the input being real. The Generator, in contrast, takes a 100-dimensional noise vector and passes it through a series of transposed convolutional layers, progressively up sampling it to generate a 64×64 synthetic image. ReLU activations are used in all layers except the final output, which uses a Tanh activation to normalize pixel values to the range [-1, 1].

Both networks were initialized using a normal distribution as recommended in the original DCGAN paper. Training was conducted using the Adam optimizer with a learning rate of 0.0002 and beta values (0.5, 0.999). Label smoothing was applied, with real labels set to 0.9 and fake labels randomly sampled from a small range near 0. The model was trained for 200 epochs with a batch size of 64. Training progress was monitored using TensorBoard by logging real and fake image grids at intervals

3.3 WGAN

To further enhance the stability and quality of GAN training, the Wasserstein GAN (WGAN) was implemented. Unlike DCGAN, WGAN replaces the traditional discriminator with a "critic" that scores realness instead of classifying inputs as real or fake. This change allows WGAN to use the Wasserstein distance (Earth Mover's Distance) as a loss metric, addressing issues like vanishing gradients and mode collapse. Importantly, unlike standard GANs where the loss often lacks direct interpretability, the WGAN loss provides meaningful insight into training progress: a lower generator loss correlates with the critic being more "fooled" by the fake samples, and the difference between critic scores for real and fake data directly estimates their distributional distance.

The critic architecture remains convolutional, with instance normalization and LeakyReLU activations for stable training, while the generator is a deep transposed convolutional network similar to the DCGAN structure. A key difference is the use of the RMSprop optimizer, recommended by the original WGAN paper, along with weight clipping (±0.01) applied to the critic to enforce the Lipschitz constraint. Training followed the WGAN procedure of updating the critic multiple times (five iterations) per generator update to ensure accurate Wasserstein distance estimation. The model was trained on 64×64 grayscale images with a 128-dimensional noise vector for 300 epochs, using a batch size of 64. TensorBoard was used to visualize training dynamics by logging real and fake image grids.

3.4 WGAN with Gradient Penalty (WGAN-GP)

The implemented WGAN with Gradient Penalty (WGAN-GP) enhances the original WGAN by addressing issues of training instability and poor gradient behavior, particularly in imbalanced data settings. For the imbalance FashionMNIST images dataset, the Generator is a deep transposed convolutional network that transforms a 100-dimensional noise vector into a 64×64 grayscale image. ReLU activations are used in intermediate layers, and a final Tanh activation normalizes output pixels between -1 and 1. The Critic (replacing the traditional discriminator) outputs real-valued scores to estimate the Wasserstein distance and is composed of convolutional layers with instance normalization and LeakyReLU activations, which help maintain stable gradient flow and prevent mode collapse. The training procedure follows the standard WGAN-GP methodology—using the Adam optimizer with learning rate 0.0001 and betas (0.0, 0.9), and updating the critic five times per generator update to accurately approximate the Wasserstein distance. The gradient penalty term ensures that the critic satisfies the 1-Lipschitz constraint without weight clipping, promoting more stable and robust training dynamics.

For the tabular Fraud Detection dataset, the WGAN-GP architecture was adapted accordingly. The Generator is a fully connected neural network that maps a 32-dimensional noise vector to a high-dimensional fraud sample using multiple linear layers, each followed by Layer Normalization and ReLU activations. The Critic similarly uses a fully connected

architecture with three linear layers and LeakyReLU activations, omitting any final activation to allow raw Wasserstein score outputs. A gradient penalty term is computed by interpolating between real and generated samples, enforcing the Lipschitz condition as proposed in the original WGAN-GP paper. The training loop adheres to the same principle as the image-based model: the Critic is updated 5 times for every generator update using the Adam optimizer with adjusted beta values. This design allows the WGAN-GP to generate high-quality, synthetic fraud samples that better represent the minority class and improve classifier sensitivity in downstream tasks.

The key difference between WGAN and WGAN-GP lies in the enforcement of the Lipschitz constraint: while the original WGAN relies on weight clipping (±0.01), which can cause training instability, WGAN-GP uses a gradient penalty computed on interpolated samples between real and fake images. This penalty enforces the 1-Lipschitz condition more effectively, preventing issues like mode collapse and vanishing gradients. The model was trained for 400 epochs with a batch size of 64, and TensorBoard was used to monitor training progress by logging real and generated images, enabling qualitative assessment of the generator's improvement over time.

3.5 Conditional Tabular GAN (CTGAN)

To address the extreme class imbalance in the fraud detection dataset, we also leveraged CTGAN, a GAN-based synthetic data generator tailored for tabular data. Unlike standard GANs, which struggle with the discrete and multimodal nature of tabular features, CTGAN is designed specifically to model such complexities. Using the open source ctgan library, i trained the model on the minority class (fraud cases) over 300 epochs. The minority data was first converted to a pandas DataFrame to match CTGAN's input format. We did not implement CTGAN from scratch, but rather utilized the high-level API provided by the library for efficient training. Despite being prebuilt, CTGAN internally handles mode-specific normalization and sampling strategies, such as conditional vector sampling, which helps in capturing the underlying distribution of rare events like fraudulent transactions.

04. Classifier Setup and Evaluation

For the Fashion-MNIST image classification task, we employed a modified ResNet-18 architecture pretrained on ImageNet. To accommodate grayscale inputs, the first convolutional layer was adjusted to accept a single input channel. The final fully connected layer was restructured to predict 10 output classes. The model was trained using the Adam optimizer with a learning rate of 0.0001 and the cross entropy loss function. To mitigate overfitting, early stopping with a patience of 5 epochs was applied, and model checkpoints were saved based on validation performance. Throughout training, key metrics such as accuracy, F1-score, recall, and precision were logged using Tensorboard, along with the training and validation loss. This enabled detailed, real-time monitoring of the model's learning behavior. Post-training, we computed the confusion matrix to visualize misclassification trends and assess class-wise performance, particularly for underrepresented categories.

For the tabular fraud detection task, we utilized a Random Forest classifier with 100 estimators and a fixed random seed for reproducibility. After training on the augmented data (including synthetic minority samples), the model’s predictions on the test set were evaluated using a confusion matrix and standard classification metrics. These included precision, recall, and F1-score, which were reported using classification_report function . The confusion matrix was visualized using ConfusionMatrixDisplay function to provide a clear depiction of false positives and false negatives, especially crucial in the context of highly imbalanced fraud detection.

05. Results & Comparisons

augmentation	Dataset	Precision	Recall	F1-score
IMBALANCE	fashionMNIST	0.906	0.880	0.891
VANILLA	fashionMNIST	0.893	0.880	0.878
WGAN	fashionMNIST	0.905	0.900	0.900
WGAN_GP	fashionMNIST	0.892	0.89	0.890
DCGAN	fashionMNIST	0.905	0.895	0.895
IMBALANCE	Fraud detection	0.941	0.816	0.874
VANILLA	Fraud detection	0.919	0.816	0.865
WGAN_GP	Fraud detection	0.8269	0.878	0.8515
CTGAN	Fraud detection	0.8830	0.846	0.8646

augmentation	Dataset	Δ Precision	Δ Recall	Δ F1-score	Improv%
VANILLA	fashionMNIST	-0.0133	-0.0132	-0.0129	-1.44 %
WGAN	fashionMNIST	-0.0020	+0.0088	+0.0092	1.03 %
WGAN_GP	fashionMNIST	-0.0143	-0.0034	-0.0014	-0.16 %
DCGAN	fashionMNIST	-0.0013	+0.0023	+0.0034	+0.39 %
VANILLA	Fraud detection	-0.022	0	-.009	-1.02 %
WGAN_GP	Fraud detection	-0.114	+0.062	-0.022	-.025 %
CTGAN	Fraud detection	-.108	+0.031	-0.01	-1.144 %

05.FashionMNIST Observations:

For FashionMNIST, most augmentation showed modest changes in metrics. The WGAN augmentation resulted in a +0.88% increase in recall, which is significant given the small scale of change across the board, improving the F1-score by +0.92%. This indicates that WGAN was able to slightly enhance the model's ability to detect minority class instances without sacrificing much in precision (just −0.20%). DCGAN also improved recall by +0.23% with a negligible precision drop, and slightly raised F1. In contrast, WGAN-GP, although powerful, caused a decline in both precision and recall, showing that not all GAN-based augmentations uniformly help when data is already well-distributed.

.

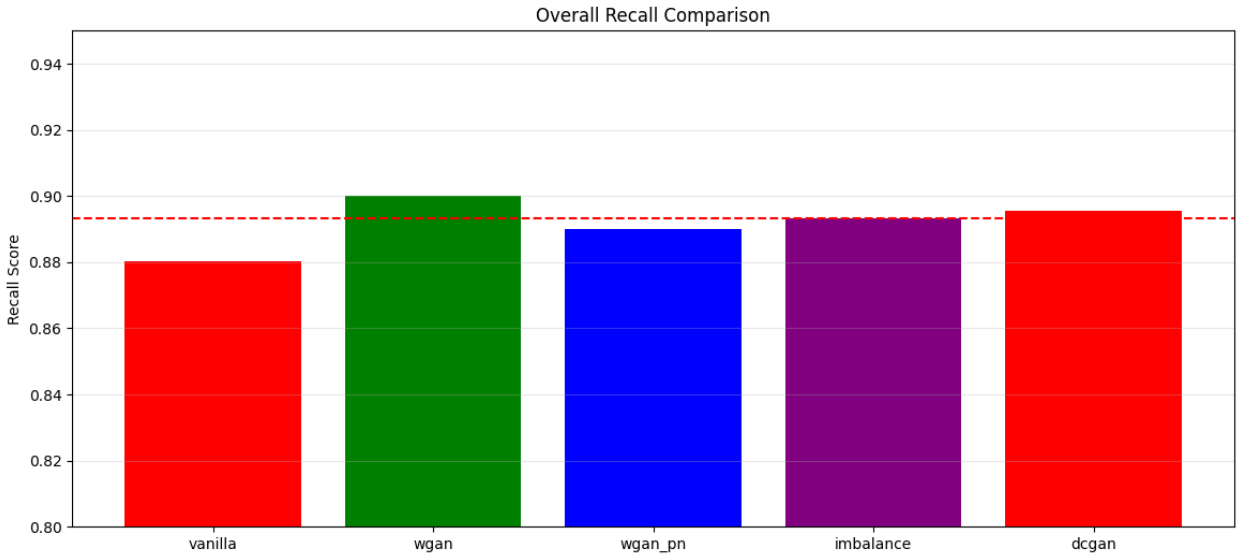

Figure 3: best recall in fashionMNIST

And the WGAN being the most recall improvement doesn't mean it's the most stable as this chart shows:

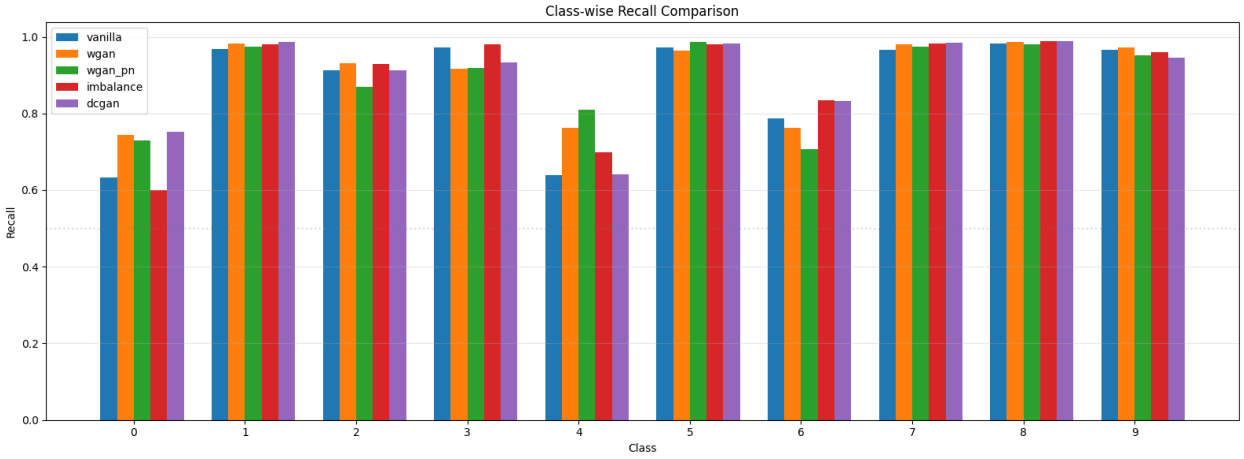

Figure 4:class-wise recall.

The experiments reveal a nuanced trade-off between stability and recall optimization across GAN architectures. While DCGAN demonstrated the most consistent performance overall maintaining robust recall across all classes without extreme fluctuations it did not always achieve the absolute highest recall values. In contrast, WGAN-GP emerged as the top performer for class 4, surpassing all other GAN variants in recall for this critical minority category. Both WGAN and DCGAN also showed marked improvements in detecting minority classes (0 and 4), underscoring their effectiveness in addressing class imbalance through synthetic data augmentation.

However, these gains in recall often came at a cost. The precision-recall trade-off became evident as models like WGAN-GP, while excelling in identifying true positives, occasionally introduced more false positives particularly in classes with ambiguous decision boundaries. This suggests that while GAN-augmented models enhance sensitivity to underrepresented samples, they may also reduce classifier specificity.

The choice of GAN should align with domain-specific needs precision-critical scenarios (e.g., low-risk fraud tolerance) may favor DCGAN, whereas recall-critical use cases (e.g., medical diagnosis) could leverage WGAN-GP.

06.fraud detection Observations:

The results reveal distinct performance trade-offs among generative models when applied to fraud detection. The imbalanced baseline achieved the highest precision (0.941) but suffered in recall (0.816), reflecting a conservative classifier that misses fraudulent cases. While VANILLA GAN and CTGAN improved recall slightly (0.816→0.846 for CTGAN), WGAN-GP achieved the highest recall (0.878) but at a significant precision cost (0.8269), suggesting it generates diverse fraud samples at the expense of more false positives. Notably, WGAN-GP's recall improvement +0.062 came with the largest precision drop −0.114 highlighting a trade-off between catching fraud and maintaining accuracy. CTGAN balanced this slightly better, with smaller deviations in both metrics. Overall, no model universally outperformed the imbalanced baseline in F1-score, underscoring the challenge of fraud detection's class imbalance. The marginal F1 declines (e.g., −0.022 for WGAN-GP) suggest that while augmentation expands minority-class representation,

The confusion matrices reveal critical differences in how each model handles fraud detection:

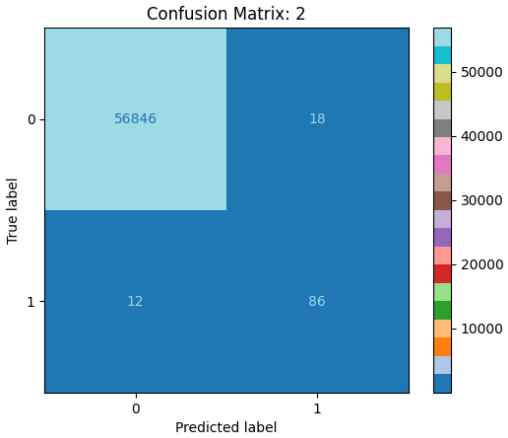

Figure 6:confusion Matrix for WGAN_GP

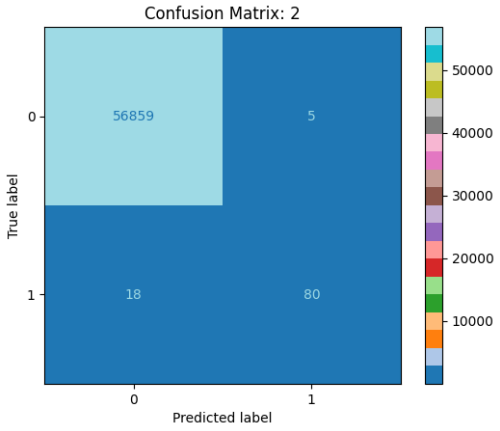

Figure 5: confusion Matrix for imbalance

- **imbalanced Baseline:**
  - **High precision but poor recall**: Correctly classifies most negatives (e.g., 56,846 true negatives) but misses 86 out of 104 fraud cases (≈82.7% false negatives). This

reflects a conservative model that prioritizes avoiding false alarms at the cost of missing fraud.

- **Severe class imbalance**: The model defaults to predicting "not fraud" (class 0) due to the dataset's skew, as seen in the extreme disparity between true negatives (56,846) and true positives (18).

- **WGAN-GP:**
  - **Improved fraud detection (recall)**: Reduces false negatives (e.g., 12 vs. 86 in the baseline), catching more fraud cases (higher true positives). However, this comes with more false positives (12 vs. 1 in the baseline), indicating a trade-off where synthetic samples may introduce noise.
  - **Balanced predictions**: The model's willingness to flag suspicious cases (higher true positives and false positives) suggests better adaptation to class imbalance, though at a slight cost to specificity.

WGAN-GP's augmentation helps address class imbalance by improving recall (fewer missed fraud cases), but practitioners must weigh this against the rise in false positives. The baseline's strict precision might suit low-risk tolerance scenarios, while WGAN-GP aligns with use cases where fraud catching is critical.

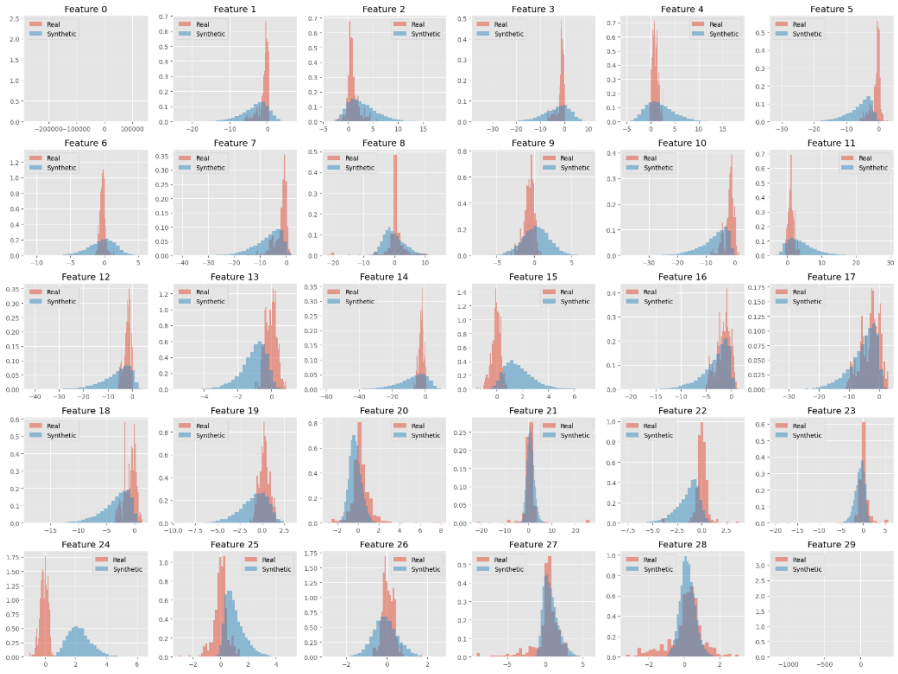

Figure 7:Vanilla GAN output distribution match.

The Vanilla GAN demonstrates significant limitations in capturing the underlying data patterns, with synthetic distributions (shown in blue) frequently exhibiting poor alignment with real distributions (shown in red) across multiple features. This is particularly evident in features such as Feature 0, Feature 6, Feature 12, and Feature 18, where the synthetic data shows either compressed ranges, shifted modes, or completely different distributional shapes compared to the real data. The synthetic distributions often appear more concentrated or skewed, indicating mode collapse issues typical of vanilla GAN training.

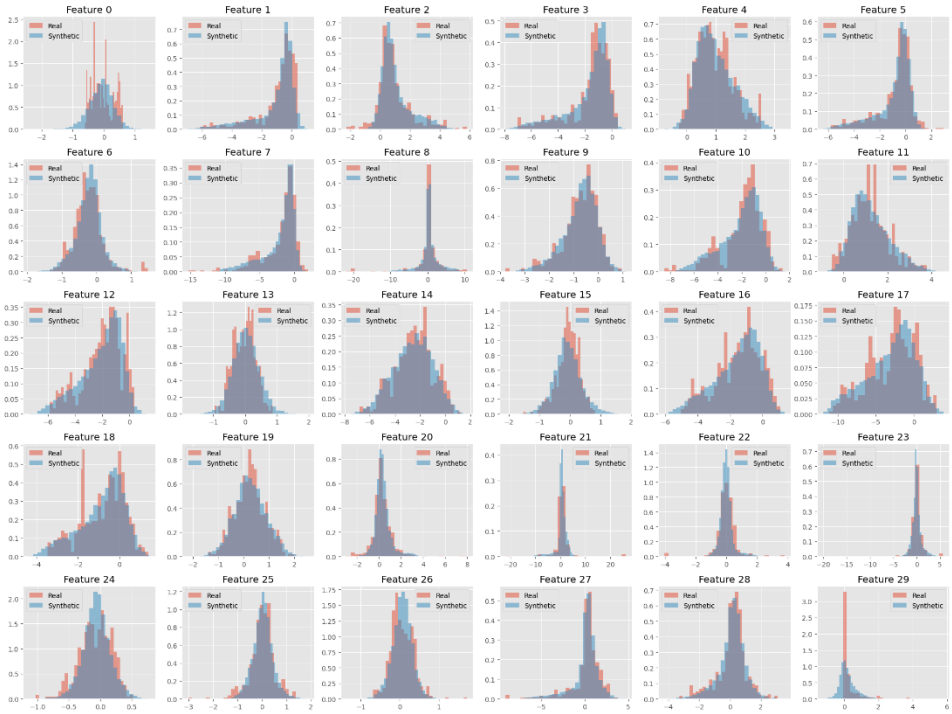

Figure 8:WGAN output distribution match.

In stark contrast, WGAN-GP shows dramatically improved performance in distribution matching across virtually all features. The synthetic data distributions generated by WGAN-GP exhibit remarkable fidelity to the real data distributions, with overlapping histograms that closely follow the same statistical patterns, modes, and spread characteristics. Features that showed poor alignment in the Vanilla GAN, such as Feature 1, Feature 3, Feature 9, and Feature 15, demonstrate near-perfect distribution matching in WGAN-GP, with synthetic and real data histograms overlapping extensively. This superior performance can be attributed to WGAN-GP's gradient penalty mechanism, which provides more stable training dynamics and better gradient flow, effectively addressing the mode collapse and training instability issues inherent in vanilla GANs. The consistent quality across all features suggests that WGAN-GP successfully captures the full complexity of the data manifold, making it significantly more reliable for generating high-quality synthetic data that preserves the statistical properties of the original dataset.

07.Conclusion

This comprehensive investigation into GAN variants for addressing class imbalance reveals nuanced performance characteristics that challenge the assumption that synthetic data augmentation universally improves classification outcomes. The study's findings across both image (FashionMNIST) and tabular (fraud detection) datasets demonstrate that the effectiveness of GAN-based augmentation is highly dependent on the specific architecture employed, the nature of the data, and the evaluation priorities of the target application.