Enhancing Classification on Imbalanced MNIST Data using Generative Adversarial Networks

1. Problem Statement

In machine learning, class imbalance occurs when the number of examples for some classes (majority) far exceeds that of others (minority). This imbalance biases classifiers toward the majority classes, leading to poor generalization and low recall for the minority class, which is often the class of most interest (e.g., fraud detection, rare disease diagnosis).

Standard techniques like data augmentation basically duplicate existing minority samples with some refinements, which might not be enough and can lead to overfitting. This project solves this problem using Generative Adversarial Networks (GANs) as a data augmentation technique. By learning the underlying distribution of the minority class, GANs can generate diverse, novel synthetic samples to balance the dataset, potentially improving the classifier's ability to generalize.

2. Description of Dataset & Imbalance Analysis

I utilized the **MNIST dataset** of handwritten digits (0-9). To simulate a real-world imbalanced scenario, I artificially reduced the number of examples for **Class 0**.

- **Original Dataset**: ~6,000 samples per class.
- **Imbalanced Training Set**:
    - **Class 0 (Minority)**: Reduced to **300 samples**.
    - **Classes 1-9 (Majority)**: ~6,000 each.
    - **Total Training Samples**: 54,377.
- **Imbalance Ratio**: Approximately 1:20 (Minority:Majority).

This severe imbalance poses a significant challenge for standard training, as the model can achieve high accuracy simply by ignoring Class 0.

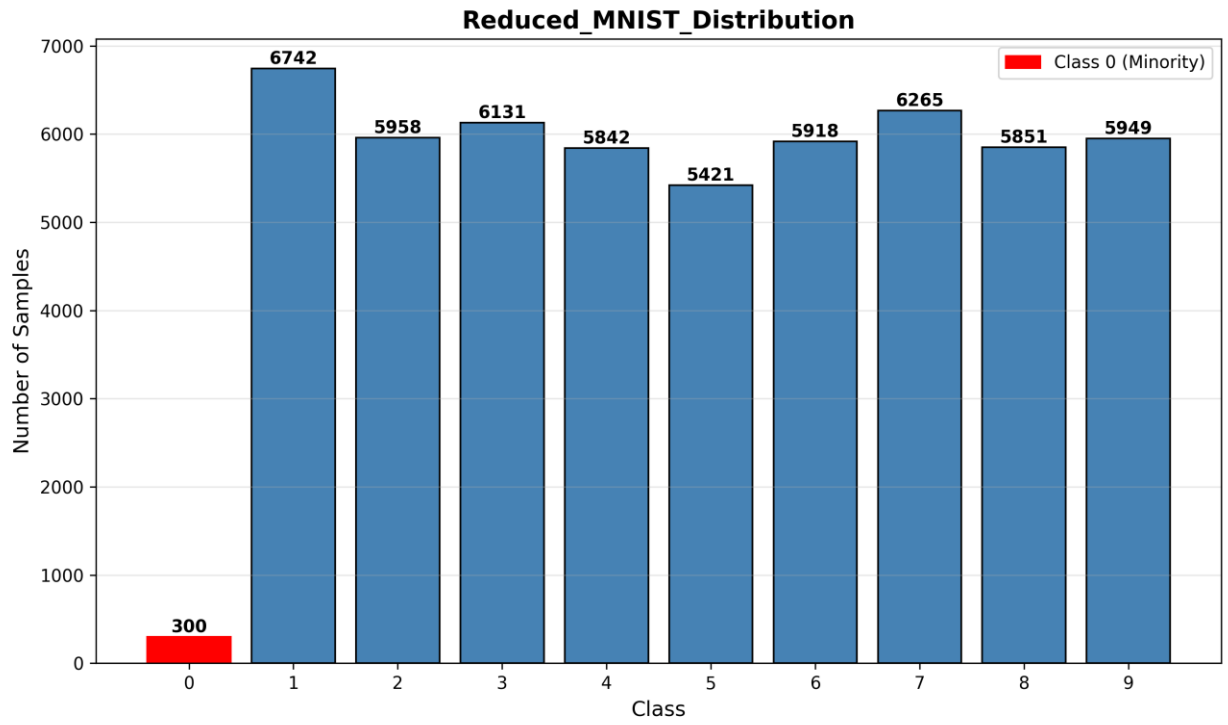


Figure 1: Reduced MNIST Distribution

3. Details of GAN Architectures & Training

All models were trained on **28×28 grayscale reduced class 0 images** normalized to $[-1,1]$, and was trained on NVIDIA RTX 4060 GPU.

A. Vanilla GAN

- **Generator:** A fully connected network that maps a 100-dimensional latent vector to a 28×28 image. It uses four Linear layers (256 → 512 → 1024 → 784) with **LeakyReLU** activations and **Batch Normalization** in intermediate layers, followed by **Tanh** output.
- **Discriminator:** A fully connected classifier that flattens the 28×28 image and processes it through four Linear layers (784 → 1024 → 512 → 256 → 1) with **LeakyReLU** and **Dropout**, ending with **Sigmoid**.
- **Training:** Trained only on the reduced Class 0 dataset.
- **Configuration:** 120 epochs.
- **Best Performance:** Epoch 72.

B. Deep Convolutional GAN (DCGAN)

- **Generator:** Projects the latent vector using a Linear layer into a 7×7×128 feature map, then upsamples using **two ConvTranspose2D layers** (7→14→28) with **BatchNorm** and **ReLU**, and **Tanh** at the output.
- **Discriminator:** A fully convolutional network with **spectral normalization** (Prevents the discriminator from becoming too strong), using strided Conv2D layers (28→14→7) and **LeakyReLU**, producing a single realism score.
- **Training:** Trained only on the reduced Class 0 dataset.
- **Configuration:** 240 epochs.
- **Best Performance:** Epoch 236.

C. Conditional GAN (CGAN)

- **Generator:** Extends DCGAN by conditioning on class labels. A learned **label embedding** is concatenated with the latent vector before projection to a 7×7×128 feature map, followed by the same convolutional upsampling pipeline as DCGAN.
- **Discriminator:** Conditions on labels by embedding the class label into a 28×28 map and concatenating it with the input image as an additional channel before convolutional classification.
- **Training:** Trained on the full MNIST dataset (all classes), enabling shared feature learning across classes to improve minority class generation.
- **Configuration:** 100 epochs.
- **Best Performance:** Epoch 40.

Synthetic Data Generation: After training, I generated **5,700 synthetic images** of Class 0 using each model to bring the total count of Class 0 samples up to 6,000, effectively balancing the dataset.

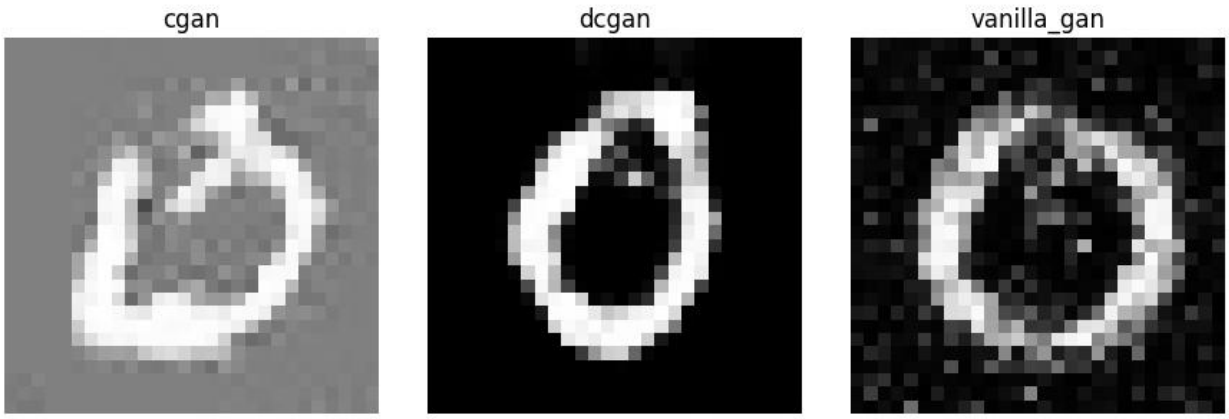


Figure 2: Samples of the Generated Images

4. Classifier Setup and Evaluation

To evaluate the quality of the generated data, I trained a Convolutional Neural Network (CNN) classifier under four different data scenarios using NVIDIA L4 GPU.

Classifier Architecture:

- 3 Convolutional Layers (32, 64, 128 filters) with ReLU and Max Pooling.
- 2 Fully Connected Layers with Dropout (0.5) to prevent overfitting.
- Output: 10 classes (Softmax).

Training Scenarios:

1. **Original**: Trained on the imbalanced dataset (300 samples of Class 0).
2. **Vanilla Augmented**: Original + 5,700 Vanilla GAN images (Class 0 balanced).
3. **DCGAN Augmented**: Original + 5,700 DCGAN images (Class 0 balanced).
4. **CGAN Augmented**: Original + 5,700 CGAN images (Class 0 balanced).

All classifiers were trained for 20 epochs with a batch size of 256 and evaluated on a separate, balanced Test Set of 10,000 images (the original MNIST testset).

5. Results & Comparisons

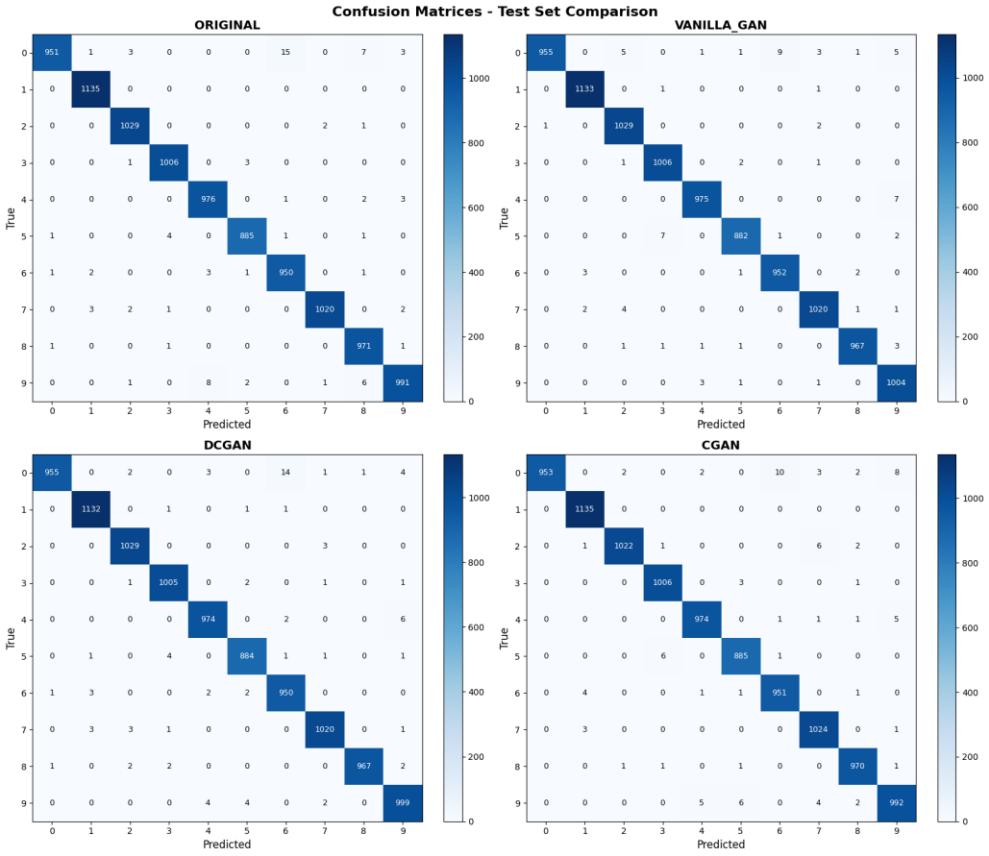
The performance was measured using Accuracy, F1-Score, Precision, and Recall.

Overall Test Set Performance:

Scenario	Accuracy	F1 Score	AUC-ROC
Original (Baseline)	98.59%	0.9859	0.9997
Vanilla GAN	**98.90%**	**0.9890**	**0.9998**
DCGAN	98.77%	0.9877	0.9998
CGAN	98.71%	0.9871	0.9998

Minority Class (Class 0) Specific Performance:

Scenario	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)
Original (Baseline)	0.9969	0.9704	0.9835
Vanilla GAN	0.9990	**0.9745**	**0.9866**
DCGAN	0.9979	**0.9745**	0.9861
CGAN	**1.0000**	0.9724	0.9860



Analysis

- **Baseline Performance**: The original classifier already performed well (97% recall on Class 0), likely because MNIST is a relatively simple dataset and 300 samples were sufficient to learn the digit "0". However, it still had the lowest performance among all groups.
- **Impact of Augmentation**: All GAN-augmented datasets improved the Recall and F1-score for Class 0. This confirms that adding synthetic data helped the model generalize better on the minority class.
- **Best Model**: Surprisingly, the **Vanilla GAN** augmentation yielded the highest overall accuracy (98.90%) and the best Class 0 F1-score (0.9866). While DCGAN and CGAN are theoretically more advanced, for this specific low-resolution task (28x28 grayscale), the Vanilla GAN produced sufficiently diverse samples to help the classifier.

6. Observations and Conclusions

This project demonstrated that Generative Adversarial Networks are a viable solution for handling class imbalance.

Key Observations:

1. **Synthetic Data Utility**: Replacing the "missing" real data with GAN-generated data consistently improved classifier robustness
2. **Model Complexity vs. Performance**: More complex GANs (CGAN/DCGAN) did not significantly outperform the simpler Vanilla GAN for this specific dataset. This suggests that for simple, low-dimensional data, a well-tuned simple GAN can be highly effective.
3. **CGAN Stability**: The CGAN benefited from training on the full dataset, allowing it to converge faster (best epoch 40) compared to models trained only on the small minority subset.

Conclusion: Data augmentation using GANs successfully mitigated the bias caused by class imbalance. By generating 5,700 synthetic samples for Class 0, I improved the classifier's precision and recall for that class, achieving a final test accuracy of **98.90%**. This approach shows great promise for real-world applications where collecting data for rare classes is expensive or difficult.