Enhancing Classification on Imbalanced MNIST Data using Generative Adversarial Networks

Overview

This project investigates the effectiveness of Generative Adversarial Networks (GANs) in addressing the problem of class imbalance in image datasets. Using the MNIST dataset as a case study, we artificially reduced "Class 0" to create a severe imbalance. I implemented and trained three different GAN architectures—Vanilla GAN, Deep Convolutional GAN (DCGAN), and Conditional GAN (CGAN)—to generate synthetic samples for the minority class. These synthetic samples were used to augment the training set, and a Convolutional Neural Network (CNN) classifier was trained to evaluate the impact on classification performance.

Project Structure

classifiers_results/ – Trained classifiers, logs, and plots
data/ – Original dataset (MNIST)
gan_saved_models/ – Checkpoints of trained GANs
generated_images_samples/ – Sample images from GAN training
synthetic_data/ – Final synthetic datasets for augmentation
classifier.ipynb – Data prep, augmentation, and classifier training
gan_train.ipynb – Define and train GAN models
class_0_reduced.pt – Reduced subset of minority class for GAN training
README.md – Project overview and instructions

Requirements

- Python 3.x
- PyTorch (with CUDA support recommended)
- Torchvision
- Matplotlib
- Numpy
- Scikit-learn
- TQDM

Methodology

1. GAN Training (gan_train.ipynb)

This notebook handles the generative modeling part of the project.

- **Data**: Loads class_0_reduced.pt (approx. 300 samples) for Vanilla/DCGAN, and full MNIST for CGAN.
- **Models**:
    - **Vanilla GAN**: A simple fully connected generator and discriminator.
    - **DCGAN**: A convolutional architecture for better image stability and quality.
    - **CGAN**: A conditional GAN trained on the full dataset, capable of generating specific classes based on input labels.
- **Evaluation**: Uses (FID) to monitor image quality during training.
- **Output**: Saves the best models and generates 5700 synthetic images for Class 0 per model into synthetic_data.

2. Classification & Evaluation (classifier.ipynb)

This notebook assesses the utility of the generated data.

- **Data Preparation**: Loads MNIST and artificially reduces Class 0 to ~300 samples (Train set).
- **Scenarios**: Creates four distinct training sets:
    1. **Original**: Imbalanced dataset (Baseline).
    2. **Vanilla Augmented**: Original + 5700 synthetic images from Vanilla GAN.
    3. **DCGAN Augmented**: Original + 5700 synthetic images from DCGAN.
    4. **CGAN Augmented**: Original + 5700 synthetic images from CGAN.
- **Classifier**: A standard CNN architecture (3 Conv layers, 2 FC layers).
- **Metrics**: Evaluates Accuracy, Precision, Recall, F1-Score, and ROC-AUC on a held-out test set (10,000 samples). Special attention is paid to Class 0 performance.

How to Run

1. **Train GANs**: Open and run all cells in gan_train.ipynb. This will train the models and generate the synthetic data folders (synthetic_data/vanilla_gan, etc.).

2. **Train Classifiers**: Open and run all cells in classifier.ipynb. This will load the synthetic data, train the classifiers for each scenario, and output the comparative results and confusion matrices.