Synthetic Dataset Generation using GANs (Generative Adversarial Networks)

DA312 Advanced Machine Learning Lab Project
Soumya Savarn (220150031)

***Abstract*—In this project, I built a tool to generate synthetic data using GANs. My app can work with both tabular (CSV) and image (JPG/PNG) data. I used CTGAN and vanilla GANs for tables, and a trained Pix2Pix GAN for comic face images. I also checked how adding synthetic data helps machine learning models, using the Iris dataset as an example.**

I. Introduction

Sometimes, we do not have enough real data for machine learning, or we want to protect privacy. I wanted to solve this by making new (synthetic) data using GANs. I made a Streamlit app where I can upload a CSV or an image, and the app will figure out what kind of data it is and let me choose how to generate new data.

A. How My App Works

- If I upload a CSV, I can pick between a simple GAN or CTGAN, set how many epochs to train, and choose how many samples to generate. The model trains right away on my data.
- If I upload an image, my app uses a trained Pix2Pix GAN model (trained for 120 epochs on comic faces).

B. Running the App

First, I install the requirements with `pip install -r requirements.txt`. Then, I run the app with `python -m streamlit run app.py`. The `testing_data` folder has some example CSVs and images. For image generation, it works best to use these test images.

II. Methodology and Experiments

A. CTGAN and Vanilla GAN for Tabular Data

I implemented two approaches for tabular data synthesis:

CTGAN Architecture:

Generator: 4-layer MLP with batch normalization, LeakyReLU, residual connections, and Tanh output.

$$G(z) : R^{64} \to R^{256} \to R^{256} \to R^{d_{out}}$$

Discriminator: 3-layer MLP with spectral normalization, LeakyReLU, and Sigmoid output.

$$D(x) : R^{d_{in}} \to R^{256} \to R^{256} \to [0, 1]$$

Vanilla GAN Architecture

Generator: 3-layer MLP with ReLU.

$$G(z) : R^{d_z} \to R^{128} \to R^{256} \to R^{d_{out}}$$

Discriminator: 3-layer MLP with LeakyReLU.

$$D(x) : R^{d_{in}} \to R^{256} \to R^{128} \to [0, 1]$$

I use Adam optimizer, BCE loss, and MinMax scaling.

B. Pix2Pix GAN for image data on Comic Dataset

Generator: U-Net encoder-decoder with skip connections. Encoder uses 4×4 convolutions, batch normalization, and LeakyReLU. Decoder uses transposed convolutions, batch normalization, ReLU, dropout, and skip connections. Output uses Tanh.

Discriminator: PatchGAN, classifies 70×70 patches as real or fake, uses convolutions and Sigmoid. I train with adversarial loss and L1 loss, using Adam optimizer. I trained for 120 epochs on Kaggle on GPU P100 for 9.5 hours (approximately).

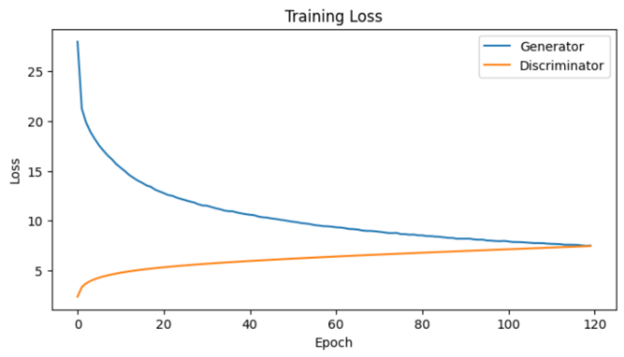

Fig. 1: training loss for Pix2Pix obtained on Kaggle after a careful training about 9.5 hours

Model	Avg L2 Score	Avg JS Distance	Avg FID Score
Face2Comic	56.8705	0.1684	17.4550

TABLE I: Average L2, JS, and FID scores for different face-to-comic conversion models.

C. Iris dataset Augmentation and Performance Boost on Classical ML algorithms

I tested synthetic data quality using:

- **Jensen-Shannon Divergence** (lower is better):
  - Petal width: 0.0068
  - Petal length: 0.0087
  - Species: 0.0145
  - Sepal width: 0.0463
  - Sepal length: 0.0848
- **Fréchet Inception Distance (FID)**: 0.53 (lower is better)

Training a Random Forest classifier showed:

- **Accuracy improved** from 90.00% (real data) to 93.33% (real + synthetic)
- **F1-score improvements**:

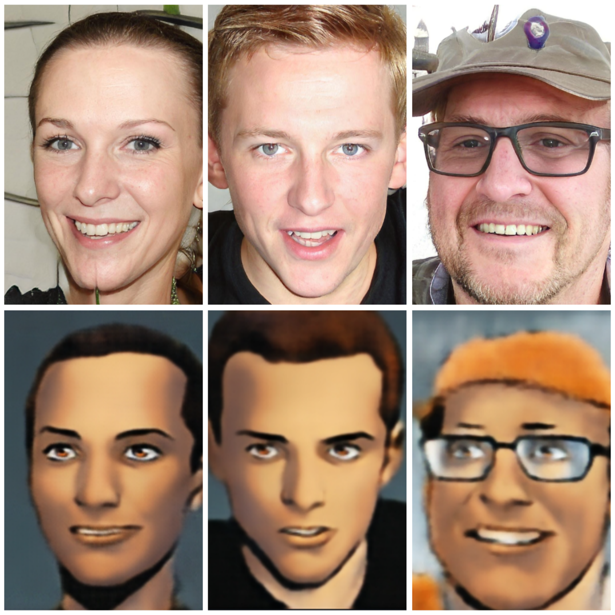

Fig. 2: test set generation by the trained Pix2Pix model

  - Class 0 (setosa): 1.00 → 1.00
  - Class 1 (versicolor): 0.86 → 0.90 (+4.7%)
  - Class 2 (virginica): 0.84 → 0.90 (+7.1%)
- **Overfitting reduction**: Smaller gap between training/validation scores

Key findings:

- Petal measurements (most important features) had best JS scores
- Higher sepal length divergence matches its higher natural variability
- Low FID score confirms good overall distribution matching

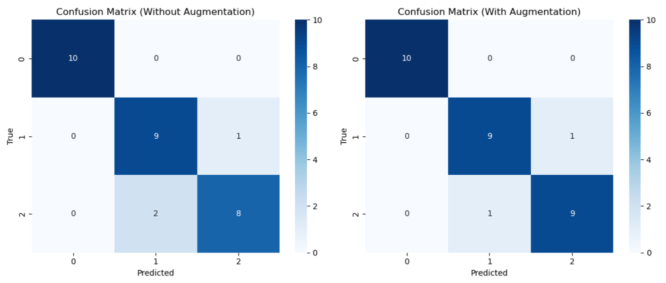

Fig. 3: confusion matrix: Real vs Synthetic Iris Data

III. Conclusion

- GANs are powerful for making new data, but training them (especially for images) takes time and care.
- For tables, training GANs on the fly works well for small datasets.
- Adding synthetic data can help machine learning models do better.

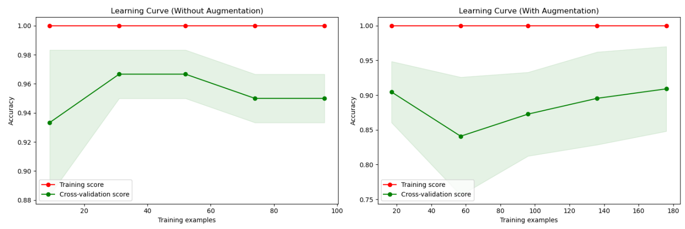

Fig. 4: learning curve comparision: Real vs Synthetic Iris Data

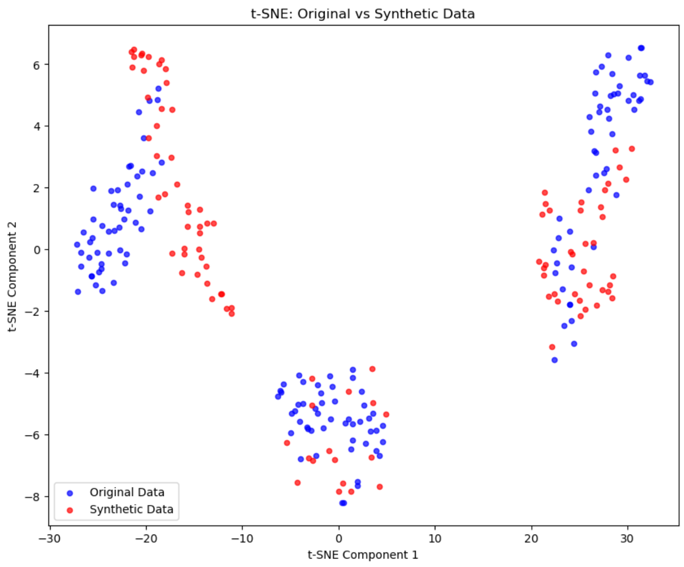

Fig. 5: t-SNE Real vs Synthetic Iris Data

- GANs can overfit, so results are best when new data is similar to the training data.

IV. Future Work

In the future, I want to:

- Try more advanced GANs and diffusion models.
- Add more options to the app, like privacy controls.
- Explore how to make GANs work better with very little data.

V. References

1) Goodfellow, I. et al. (2014). Generative adversarial nets.
2) Xu, L. et al. (2019). Modeling Tabular data using Conditional GAN.
3) Radford, A. et al. (2015). Unsupervised representation learning with DCGAN.