Project Report: Anime Face Generation using GANs

📌 Objective

The goal of this project was to design and train a **Generative Adversarial Network (GAN)** capable of generating high-quality **anime face images**. Using the publicly available Anime Face Dataset from Kaggle, the model was trained to learn the underlying patterns of anime-style facial features and synthesize novel, realistic-looking faces.

🔨 Methodology

1. Data Preparation

- **Source**: Anime Face Dataset (~63,000 images)
- **Preprocessing Steps**:
    - Loaded and decoded `.jpg` files from dataset folder.
    - Resized all images to **64×64 pixels** for efficient training.
    - Normalized pixel values to range **[-1, 1]** (as expected by `tanh` output layer).
- **Batching**: Used TensorFlow's `tf.data.Dataset` API with shuffle, batch (size = 256), and prefetch for optimized data pipeline.

---

2. Model Architecture

Generator:

- **Input**: 100-dimensional random noise vector.
- **Layers**:
    - Dense → Reshape to 8×8×256
    - 3× `Conv2DTranspose` layers to upsample to 64×64
- **Activations**: `ReLU` (hidden layers), `Tanh` (output)

Discriminator:

- **Input**: 64×64×3 RGB image
- **Layers**:
    - 3× `Conv2D` layers with `LeakyReLU` and `Dropout`
    - Flatten → Dense
- **Output**: Single logit score (real/fake)

Losses:

- **Generator**: Binary Crossentropy (tries to fool discriminator)
- **Discriminator**: Binary Crossentropy (classify real/fake)

---

3. Training Strategy

- **Epochs**: 50
- **Optimizers**: Adam with learning rate = 0.0001 and β_1 = 0.5
- **Batch Size**: 256
- **Regular Checkpoints**: Model saved every 5 epochs
- **Image Generation**: Snapshot of 16 generated images saved every epoch

---

Qualitative Analysis:

- **Image Quality**:
  - Images are visually sharp with coherent anime facial structures.
- **Diversity**:
  - Generated faces show a variety of hairstyles, colors, and expressions.

Challenges Faced

1. **Training Instability**:
   - Early epochs saw frequent mode collapse and poor discriminator loss.
   - Solution: Used `label smoothing` and tuned learning rates.
2. **Hardware Constraints**:
   - Training on Google Colab with limited VRAM required tuning batch sizes.
3. **Evaluation Difficulties**:
   - No labeled classes made using classifier-based metrics tricky.
   - Solution: Used pre-trained InceptionNet and resized images for FID/IS.
4. **GAN Saturation**:
   - At times, the discriminator became too strong.
   - Introduced Dropout and moderate label noise to improve balance.