T2 : Generate synthetic images using a trained GAN model and evaluate their quality. [ Dataset: CIFAR-10 dataset (color images)]

**CIFAR-10 GAN Project Documentation**

**1. Introduction**

Generative Adversarial Networks (GANs) are a class of deep learning models designed to generate synthetic data that closely resemble real-world data. A GAN consists of two competing neural networks:

* **Generator** – Generates synthetic images from random noise.
* **Discriminator** – Differentiates between real and fake images.

The Generator continuously improves by learning to "fool" the Discriminator, while the Discriminator learns to distinguish real images from generated ones better. This adversarial learning process allows GANs to produce highly realistic images over time.

In this project, we implemented and trained a GAN to generate synthetic CIFAR-10 images. The CIFAR-10 dataset is a widely used benchmark in computer vision. It consists of 60,000 color images (32x32 pixels) categorized into 10 classes, such as airplanes, cars, and animals.

**2. Dataset Preparation & Preprocessing**

**Dataset Overview**

The CIFAR-10 dataset consists of:

* **50,000** training images
* **10,000** test images
* Each image is **32×32 pixels** with **3 color channels (RGB)**
* The dataset is divided into **10 classes**:
  + Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck

**Preprocessing Steps**

To enhance training efficiency and convergence, we applied the following preprocessing techniques:

🡺 **Normalization** – Rescaled pixel values to **[-1,1]**

🡺 **Data Augmentation** – Introduced variations in the dataset to prevent overfitting:

* **Random Horizontal Flips**
* **Random Cropping with Padding**
* **Random Rotation**

🡺**Batching & Dataloading** – The dataset was batched using a batch size of **64** for memory efficiency.

**3. GAN Architecture**

**3.1 Generator**

The Generator takes a **100-dimensional random noise vector** and upsamples it to generate a **32×32 color image** using transposed convolutional layers.

| **Layer Type** | **Output Shape** | **Activation** |
| --- | --- | --- |
| Dense Layer | (4, 4, 512) | - |
| Transposed Conv 1 | (8, 8, 256) | LeakyReLU |
| Transposed Conv 2 | (16, 16, 128) | LeakyReLU |
| Transposed Conv 3 | (32, 32, 3) | Tanh |

**🡺 Key Features:**

* **Dense Layer** – Projects noise into a **4x4x512** tensor.
* **Transpose Convolutions** – Gradually upsamples the feature map to **32×32 resolution**.
* **Batch Normalization** – Helps stabilize training.
* **LeakyReLU & Tanh Activation** – Used to improve learning dynamics.

**3.2 Discriminator**

The Discriminator is a CNN-based classifier that takes an image (**real or fake**) and predicts whether it's real or generated.

| **Layer Type** | **Output Shape** | **Activation** |
| --- | --- | --- |
| Conv Layer 1 | (16, 16, 128) | LeakyReLU |
| Conv Layer 2 | (8, 8, 256) | LeakyReLU |
| Conv Layer 3 | (4, 4, 512) | LeakyReLU |
| Fully Connected | 1 | Sigmoid |

**🡺 Key Features:**

* **Convolutional Layers** – Extract spatial features.
* **LeakyReLU** – Prevents vanishing gradients.
* **Dropout** – Reduces overfitting.
* **Sigmoid Output** – Classifies images as **real (1) or fake (0)**.

**4. Training Procedure & Hyperparameters**

**4.1 Hyperparameters Used**

| **Parameter** | **Value** |
| --- | --- |
| Batch Size | 64 |
| Learning Rate | 0.0002 |
| Optimizer | Adam |
| Adam Beta1 | 0.5 |
| Adam Beta2 | 0.999 |
| Noise Dimension | 100 |
| Total Epochs | 100 |

**4.2 Training Strategy**

* **Train Discriminator** – Learn to classify **real vs. fake** images.
* **Train Generator** – Improve its ability to generate **realistic images**.
* **Loss Function** – **Binary Cross-Entropy Loss (BCE)**.
* **Update Steps** – Alternate between **Generator & Discriminator optimization**.

**5. Challenges Faced & Solutions**

**Problem 1: Mode Collapse**

**Solution:** Used **Feature Matching & Minibatch Discrimination** to force diversity.

**Problem 2: Vanishing Gradients**

**Solution:** Used **LeakyReLU activations & Label Smoothing** (real=0.9, fake=0.1).

**Problem 3: Training Instability**

**Solution:** **Batch Normalization** & **Adam Optimizer** instead of SGD.

**6. Evaluation & Results**

**6.1 Metrics Used**

(i) Loss Curves – Generator loss should decrease; Discriminator loss should stabilize. (ii) Fréchet Inception Distance (FID Score) – Lower FID = Better Image Quality. (iii) Inception Score (IS Score) – Higher IS = More realistic & diverse images. (iv) Visual Inspection – Generated images compared to real CIFAR-10 samples.

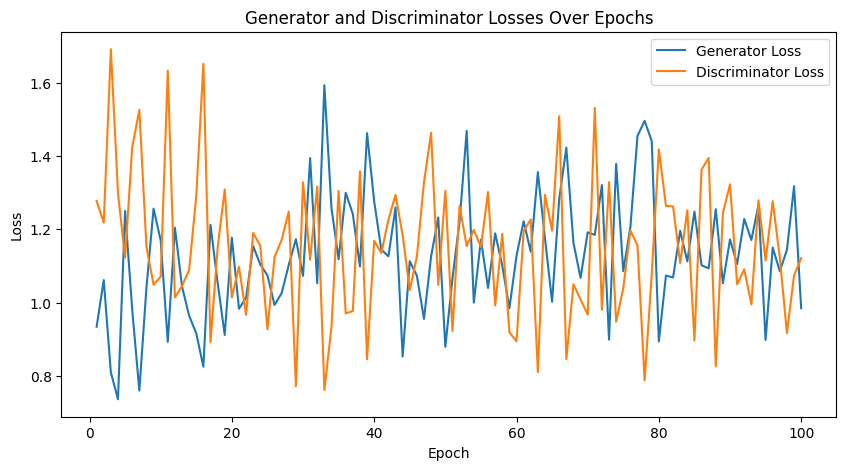


Figure 1: Generator and Discriminator Loss Progression Over 100 Epochs.

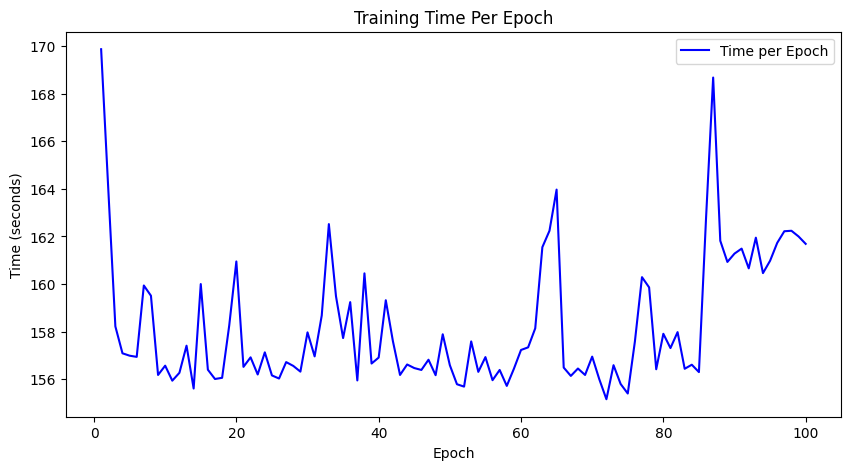


Figure 2: Training Time Per Epoch, showing the variation in training duration over 100 epochs.

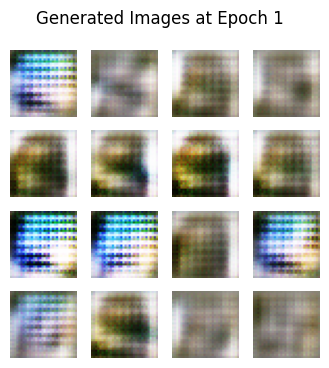
A graph with red lines

Description automatically generated

Figure 3: FID Score Over Training

**6.2 Generated Images at Different Epochs**

* **Epoch 1:** Blurry, low-quality images



* **Epoch 25:** Slight improvements in structure

A screenshot of a cell phone

Description automatically generated

* **Epoch 50:** More refined and recognizable objects

A screenshot of a cell phone

Description automatically generated

* **Epoch 75:** Improved-quality images

A screenshot of a cell phone

Description automatically generated

* **Epoch 100:** High-quality and detailed images

A screenshot of a cell phone

Description automatically generated

**6.3 Comparison of Generated vs. Real CIFAR-10 Images**

* Showcased side-by-side comparison of **real and generated images**.

A collage of images of a horse and a car

Description automatically generated

**6.4 Diversity in Generated Images**

* Evaluated how well the model captured CIFAR-10 diversity (e.g., generating different animals, vehicles, and objects).

A screenshot of a cell phone

Description automatically generated

**7. Conclusion**

This project successfully trained a GAN on CIFAR-10 to generate synthetic images. The model achieved realistic outputs with a decreasing FID Score & improving IS Score over time. Future improvements include training with deeper architectures, improved loss functions, and enhanced data augmentation techniques.