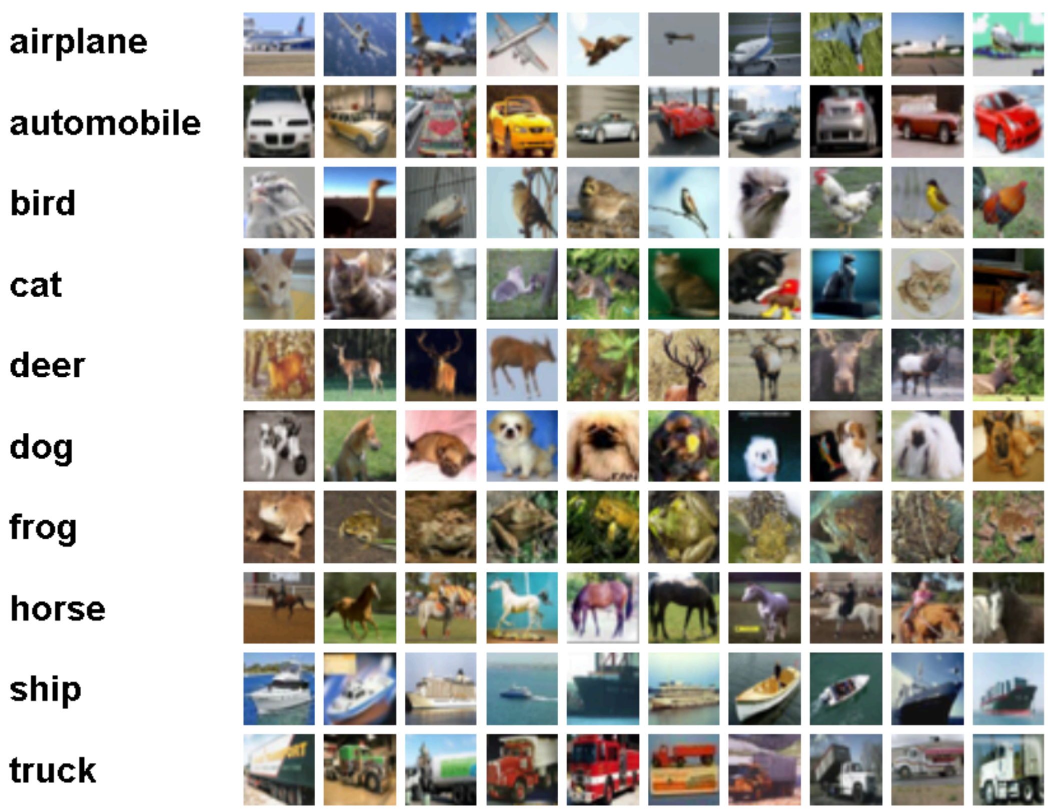
Project Overview: Using DCGAN on CIFAR-10 for Image Generation

The process to get the dataset:

To obtain the CIFAR-10 dataset using TensorFlow Datasets (TFDS), the process begins with installing the necessary libraries, including TensorFlow and TensorFlow Datasets. Once installed, the dataset is loaded using the `tfds.load()` function, which automatically downloads and prepares the dataset for use. The dataset is then split into training and testing sets, containing 50,000 and 10,000 images, respectively.



1. Overview of GAN Architecture and Choice of Hyperparameters

For this project, a **Deep Convolutional Generative Adversarial Network (DCGAN)** was implemented using the **CIFAR-10 dataset**. DCGANs improve traditional GANs by leveraging convolutional layers instead of fully connected ones, making them particularly suitable for image generation tasks.

Architectural Choices:

- **Generator:** Uses transposed convolutions (ConvTranspose2D) to upsample noise into realistic images.
- **Discriminator:** Uses convolutional layers with batch normalization and LeakyReLU activation to classify real vs. fake images.
- **Latent Space Dimension:** A **100-dimensional** random noise vector was used to generate images.
- **Activation Functions:** LeakyReLU was used in the discriminator, while the generator used ReLU and Tanh.

Hyperparameter Selection:

- **Learning Rate:** Set at **0.0001** using the **Adam optimizer** for both networks.
- **Batch Size: 256**, balancing training efficiency and stability.
- **Epochs:** Initially set at **100**, but increased to **200** due to blurry outputs.
- **Beta1 for Adam Optimizer: 0.7**, to prevent oscillations in learning.
- **Loss Function: Binary Cross-Entropy (BCE)** loss.

---

2. Observations on Training Stability and Solutions

During training, several challenges arose that impacted the model's ability to generate high-quality images:

Real image from the dataset:

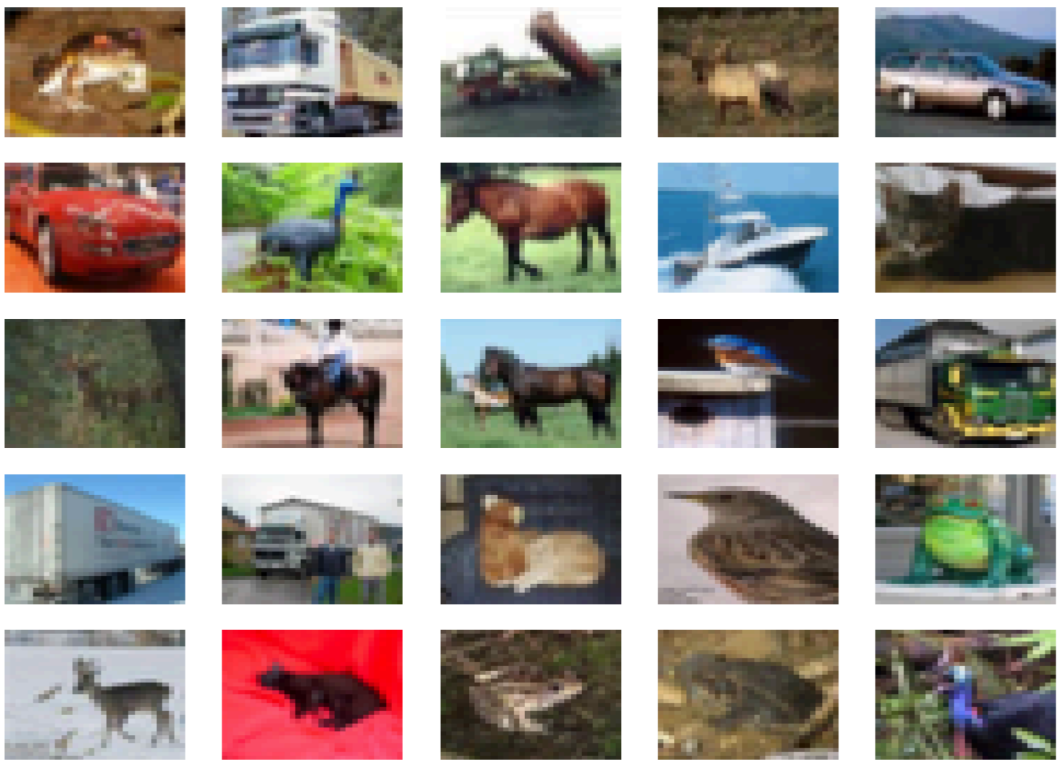


Fig1: The CIFAR-10 dataset

Image generated for 200 Epochs:

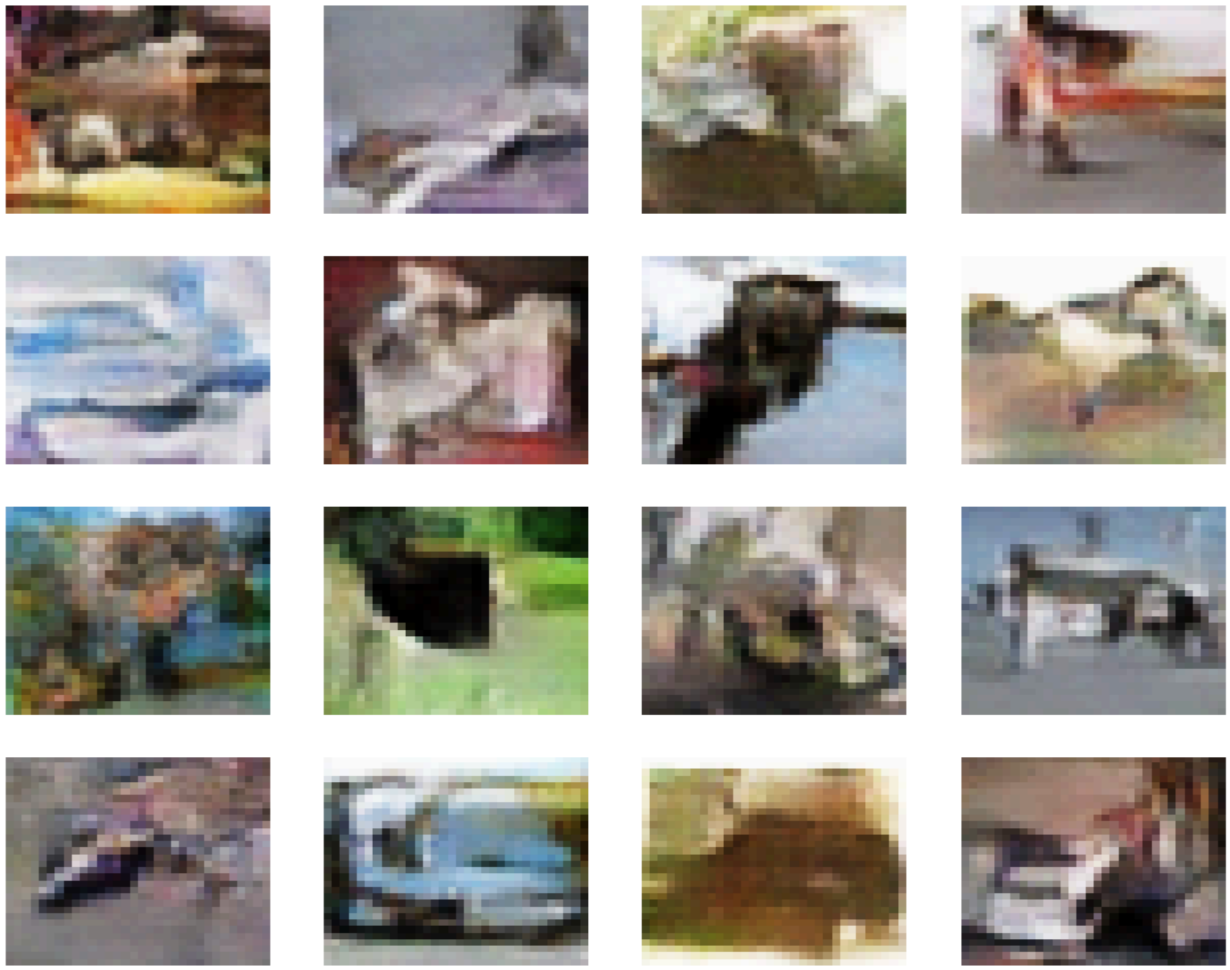


Fig2: Image generated after training for 200 Epochs

Challenges Encountered:

1. **Blurry Generated Images:**
    - CIFAR-10 images have low resolution (32x32), making it hard for the generator to learn fine details.
    - Solution: Increased training epochs from **100 to 200**, allowing more time for the model to converge.
2. **Mode Collapse:**
    - The generator sometimes produced highly similar images, failing to capture diversity.

    - Solution: Introduced **batch normalization** in both networks and adjusted the **learning rate** to prevent overfitting to a few patterns.
3. **Unstable Discriminator Loss:**
    - The discriminator quickly outperformed the generator, leading to vanishing gradients.
    - Solution: Used **label smoothing** (assigning real labels as 0.9 instead of 1.0) to slow down discriminator dominance.

---

3. Visual Results: Comparing Generated and Real Images

- At **epoch 50**, generated images appeared noisy and lacked clear object structures.
- By **epoch 100**, recognizable shapes began to form, but images were still blurry.
- By **epoch 200**, outputs became significantly sharper but still not up to the dataset used which requires more training Epochs so that it will closely resemble CIFAR-10 classes.

A side-by-side comparison of real vs. generated images showed:

- Real images: blurry, well-defined objects.
- Generated images: Improved over time but still lacked fine details.

Graph illustration of loss functions:

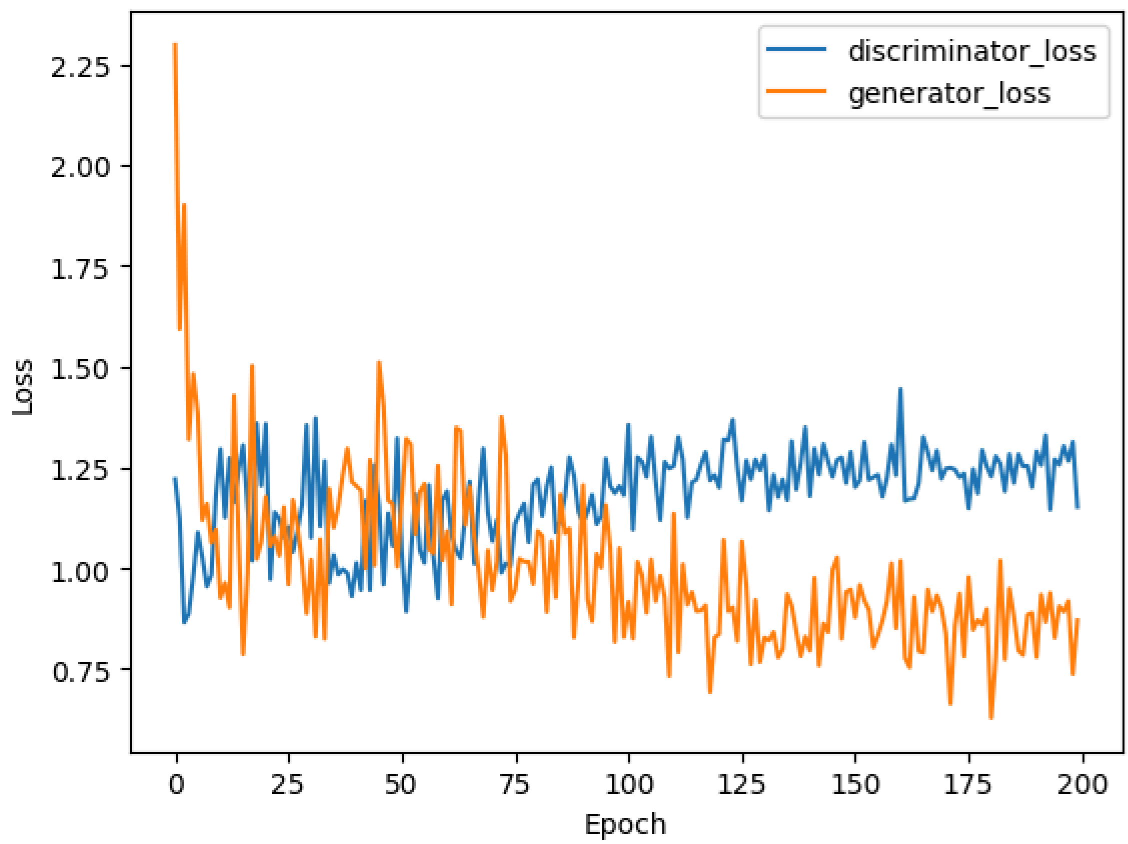


Fig3: Graph shows the discriminator and generator struggling with the

---

4. Practical Challenges in Using GANs

1. Computational Requirements:

- Training GANs on a **CPU** was prohibitively slow.
- A **GPU (e.g., NVIDIA RTX 3090)** drastically reduced training time from **days to hours**.

2. Hyperparameter Sensitivity:

- GANs are highly sensitive to **learning rates, batch sizes, and optimizer choices**.
- Fine-tuning was essential to avoid mode collapse and gradient instability.

3. Evaluation Difficulties:

- Unlike classification tasks, GANs lack a clear accuracy metric.
- Used **FID (Fréchet Inception Distance)** to assess image realism.

Conclusion

Implementing DCGAN on CIFAR-10 required extensive **hyperparameter tuning and computational resources**. Despite initial challenges with blurry images and unstable training, solutions like **longer training epochs, batch normalization, and label smoothing** significantly improved results. Future improvements could explore **Progressive Growing GANs (PGGAN) or StyleGAN** for enhanced image quality.