Custom GAN Implementation for CIFAR-10

Manahil Sarwar
AI-K Section,
FAST NUCES

***Abstract*—This report presents the implementation of a Generative Adversarial Network (GAN) for generating images of cats and dogs from the CIFAR-10 dataset. The network architecture includes a Generator and a Discriminator, trained to generate realistic-looking images. The results demonstrate progressive improvements in the quality of generated images over training epochs.**

***Index Terms*—Generative Adversarial Networks (GAN), CIFAR-10, Deep Learning, PyTorch, Image Generation**

I. Introduction

Generative Adversarial Networks (GANs) have become a popular technique in computer vision for image generation tasks. This assignment focuses on implementing a GAN to generate images of cats and dogs from the CIFAR-10 dataset. The GAN consists of two primary components: the Generator, which creates images, and the Discriminator, which classifies images as real or fake. Both models are trained adversarially, with the goal of improving the Generator's ability to produce realistic images over time.

II. Methodology

A. Dataset

We use the CIFAR-10 dataset, which contains 60,000 32x32 color images across 10 classes. For this assignment, we filtered the dataset to include only two classes: cats and dogs. The dataset was split into training and test sets.

B. Data Preprocessing

The images were normalized to have values between -1 and 1 using the transformation:

- Convert images to tensors
- Normalize with a mean of 0.5 and a standard deviation of 0.5

This normalization improves training stability for deep neural networks. Below is an example of an image from the dataset:

Fig. 1. Example image from CIFAR-10: Cat

C. Model Architecture

The GAN architecture includes two models:

- **Generator:** A deep neural network that uses transposed convolutions to generate images from random noise.
- **Discriminator:** A convolutional network using spectral normalization and minibatch discrimination to classify images as real or fake.

The Generator uses a latent vector of size 128 and progressively upsamples to create images of size 32x32x3. The Discriminator applies several convolution layers to distinguish real from fake images.

D. Training Procedure

The GAN was trained for 25 epochs using the Adam optimizer. The loss functions used were Binary Cross Entropy (BCE) for both Generator and Discriminator. During each iteration:

- The Discriminator was updated using both real and generated (fake) images.
- The Generator was updated to fool the Discriminator into classifying fake images as real.

III. Results

A. Training Loss

The loss for both Generator and Discriminator was recorded during training. The loss values stabilized after several epochs, indicating the network reached a balanced state between generation and discrimination.

B. Generated Images

After training, the Generator was able to produce realistic images of cats and dogs. The following figure shows a selection of generated images after 25 epochs:

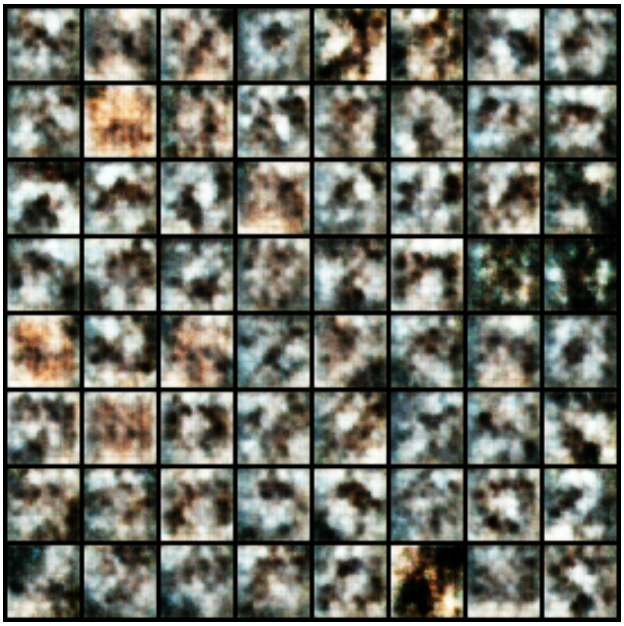

Fig. 2. Generated Images after 25 Epochs

IV. Discussion

Over the course of training, the quality of the generated images improved significantly. Initially, the Generator produced noisy and unrecognizable images. However, as training progressed, the images became clearer, with distinguishable features of cats and dogs.

Challenges encountered include:

- Mode collapse during early epochs, where the Generator produced similar-looking images.
- Training instability, which was mitigated by adding noise to the real images during Discriminator updates.

V. Conclusion

In conclusion, the implemented GAN successfully generated realistic images of cats and dogs from the CIFAR-10 dataset. The Generator improved its performance over time by learning from the feedback provided by the Discriminator. The adversarial training framework proves to be a powerful tool for image generation tasks.

VI. References

References

[1] I. Goodfellow et al., "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[2] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Technical Report, 2009.

[3] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.