Assignment: Exploring GANs with Pretrained Models

Introduction

What is a GAN and how does it work?

A GAN is a Generative Adversarial Network and it is a type of machine learning model that generates new, realistic data based on existing data. It consists of two neural networks working against each other—a generator and a discriminator. The generator produces fake data starting from random noise to try to mimic real data, while the discriminator receives both real data and fake data (from the dataset) and fake data (from the generator). This adversarial process continues until the generator creates data that is indistinguishable from the real data. GANs are commonly used to generate images, videos, or audio and are famous for applications like creating deepfake content and enhancing images.

Experiment Summary

What model did you use (e.g., BigGAN)?

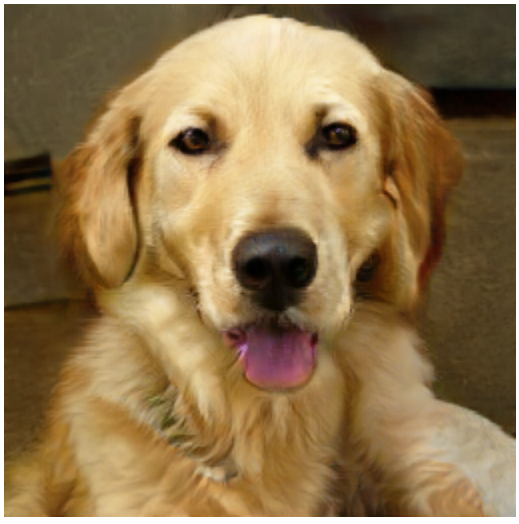
For this project, I used BigGAN-deep-256, which is a pre-trained BigGAN model which was loaded from HuggingFace's `pytorch-pretrained-biggan` library.

How did you generate images from noise?

I generated images from noise by first sampling a 128-dimensional latent vector (noise) from a truncated normal distribution using `truncated_noise_sample()`. I also used a class vector (a one-hot encoded vector indicating a specific category) and a truncation value to control image diversity and realism.

Observations

Describe 3–5 generated images and any noticeable patterns.



This image was generated with a truncation value of 0.4 and a class vector of 207.

This image was generated with a truncation value of 0.9 and a class vector of 207.

This image was generated with a truncation value of 0.9 and a class vector of 980.

How did changes to the latent vector affect the images?

Changes to the latent vector affect the images through subtle smooth variations such as different head poses and background shifts.

Reflection

What did you learn about how GANs generate images?

GANs can generate surprisingly realistic and diverse images from random vectors. Pretrained models like BigGAN make high-quality synthesis accessible.

Were there any limitations or challenges? What could be improved?

Some outputs were not as realistic, highlighting GAN limitations when the latent vector does not align well with learned features. One improvement is to use class-conditional inputs to guide the GAN toward specific image types.