**Introduction**

A Generative Adversarial Network (GAN) is a generative transformer model that takes an input (as noise or actual) and converts it into new or updated data. For example, a GAN may take a black-and-white image as an input, then output a colored version of the image. It is able to perform this with the help of a generator trained for this purpose. A GAN also has a discriminator that evaluates whether the output of the generator is fake or real. This discriminator is essential in training the generator. This is how the training works:

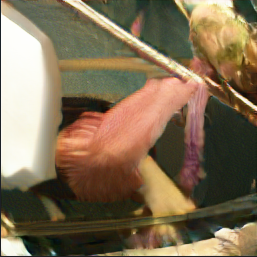
Let’s say we have a dataset of colored images. We convert them into black-and-white images, which is now our input dataset. We give these images to our generator. Our generator performs its operations and creates an output image. This output image is passed on to the discriminator that compares it with the original dataset of colored images and tries to determine which of the two is real and which is fake. The discriminator tries its best to guess correctly and improves its classification through training. Whereas the generator tries its best to fool the discriminator, improving its image generation.

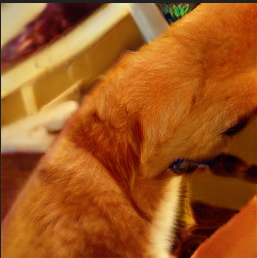
**Experiment Summary**

I used the BigGan model from the pytorch\_pretrained\_biggan library. To use this model, I first loaded the pretrained model. Then I created a random vector of size 1x128, which is a batch size of 1 with 128 tensors to match the input vector size for BigGan. Then, I created a class label for Golden Retriever with the same batch size and gave the model a truncation value of 0.4 to introduce a little diversity.

**Observations**

When I was generating the images, I noticed huge changes between the images even with the same class label, just by changing the values of the input vector. This can be majorly attributed to how a generator generates the images. A generator has its own weights and biases that are multiplied and added to the input vector values to create a generated image. Changing the values of the input vectors changes the final calculated values leading to drastically different images. The following are three of the images I generated:

A group of dogs playing in the grass

AI-generated content may be incorrect.

These are all generated to be part of the Golden Retriever class. Some images (2nd and 4th) can be clearly identified as dog, whereas some (1st and 3rd) only partially address the class.

**Reflection**

I learnt that the images generated by GANs are heavily dependent upon the input. Even if the GAN is attempting to draw the same type of image, many different variations may be present depending on the training data. None of the images generated, however, can be said to make complete sense, meaning being able to perfectly generate the requested image from random noise may be very difficult. This can be attributed to the fact that I had added truncation, but this issue could be improved by slightly altering the training technique. From what I can see, the discriminator of the GAN may need more training, so training the discriminator separately might be one method to improve the generation of the GAN. If truncation is the only issue, then training with truncation can help improve the GAN.

**Notebook Link:** <https://github.com/Kartik-Tyagi/CognizantExternship/blob/main/GenAI/GAN.ipynb>