Nicholas Hamilton

Cognitive Generative AI Externship

May 25 2025

Generative Adversarial Networks Overview

Generative Adversarial Networks take two AI models and pits them against each other. In a common supervised machine learning workflow, we can see the following: We feed this model input, create a model, then compare the model's output with the training data, and from there determine how to update our model to get better outputs. GAN's are actually unsupervised! This form of AI supervises itself. We have two submodel components: The generator, and the discriminator. The generator has the primary task of creating fake samples, fake data, that then gets passed onto the discriminator. From there, the discriminator takes this data, and attempts to determine if it is a real or fake sample.

This process happens iteratively, over and over, until the generator can convince the discriminator that the data is real. A common use case of this kind of model is image generation. We can start by training the discriminator submodel. We teach this network to be able to correctly identify a certain object, while pausing the generator. The generator then takes a random input vector, and attempts to create a fake flower. This is then sent to the discriminator, where the decision is made. Both of these models receive the answer, that the flower is fake. Both models will change their behavior. The winner of these two models does not have to change for this round, while the other model will have to update.

Eventually, the discriminator can no longer tell when the data is fake, and that is the point in which we can determine that we have built a successful GAN. Both of these sub-models are usually Convolutional Neural Networks.

While image creation is one common use of this framework, there are many possibilities! If we can identify in a video common themes leading up to certain events, we may be able to then predict when those events occur! This leads to items like surveillance systems to become much more advanced. We can also see an example like, taking a low resolution image, use a GAN, and then create a higher image resolution from that! We can even use these iterations to create something like strong encryption algorithms, and continually test and make a stronger algorithm!

For this project, we start by creating 128 random numbers, and give that to the model. We then give the model a class, what we want to create. We fine tune how creative we want our model to be through the truncation hyperparameter, basically dictating how strict we want the model to be when following its training data. Then we convert the image to a photograph in the range that files expect. Adjusting the latent vector acts like rolling the dice on the kind of image we get! Radically changes aspects of it.

What is going on behind the scenes, is the noise vector is a sample of 128 numbers that is

a normal distribution. When we change the truncation, we are actually asking the computer for values that fall within that many standard deviations of the mean. If we say 0.4, then 0.4 standard deviations. Wecan actually calculate the exact probability of any truncation value. For truncation = 0.4, you're using noise that has about a 31% probability of occurring naturally. For truncation = 1.0, you're using noise with about a 68% probability. This mathematical relationship explains why lower truncation values produce more "safe" images - We are literally constraining the AI to use only the most probable inputs.

Our noise vector does not directly become an image. BigGAN morphs the 128 numbers into a massive amount of numbers that becomes our final image.