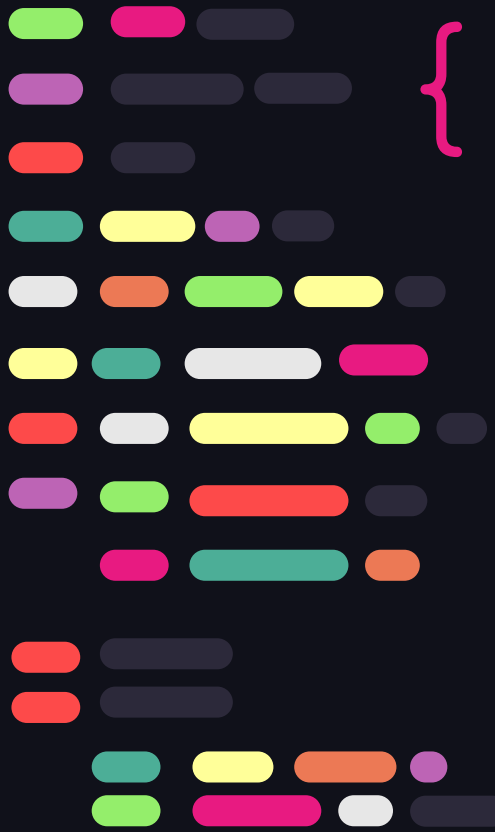


Generative Adversarial Network

How does it work? And how to build your own?

01 { ..

Intro to GAN

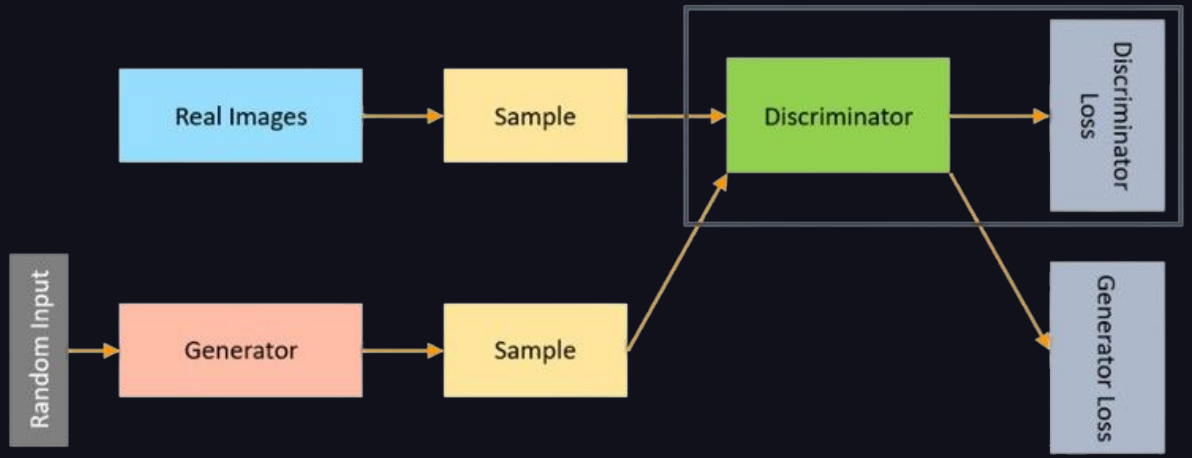
} ..

What is a GAN?

- A type of unsupervised machine learning algorithm
- Introduced by Ian Goodfellow and Co. in a 2014 paper
- Powerful and popular method for generating synthetic data
- Advantages include:
  - Realistic Data Generation
  - Diversity in Generated Samples
  - No Encoding required, just generates data from noise
- Disadvantages include:
  - Very difficult to train
  - Mode collapse - Same type of samples generated over and over
- There are many applications such as:
  - Image Generation
  - Style Transfer
  - Super Resolution
  - Data Augmentation
- Many variations exist such as StyleGAN, WGAN, CGAN and more

Parts of a GAN

A GAN consists of a generator and discriminator. The generator's job is to generate realistic looking images from noise, to fool the discriminator. Meanwhile, the discriminator's job is to determine real images and generated images. Both work together to help us generate images. First the generator generates images. Then the discriminator is fed the real and fake images, which it then categorizes. Then the loss values are calculated, and backpropagation is performed to train the models



02 { ..

Understanding GAN Loss

Goal: Lower loss for respective models

} ..

Loss Functions

Generator Loss Function:

$$L(G) = -\frac{1}{m}\sum_{i=1}^{m}\log(D(G(z_i)))$$

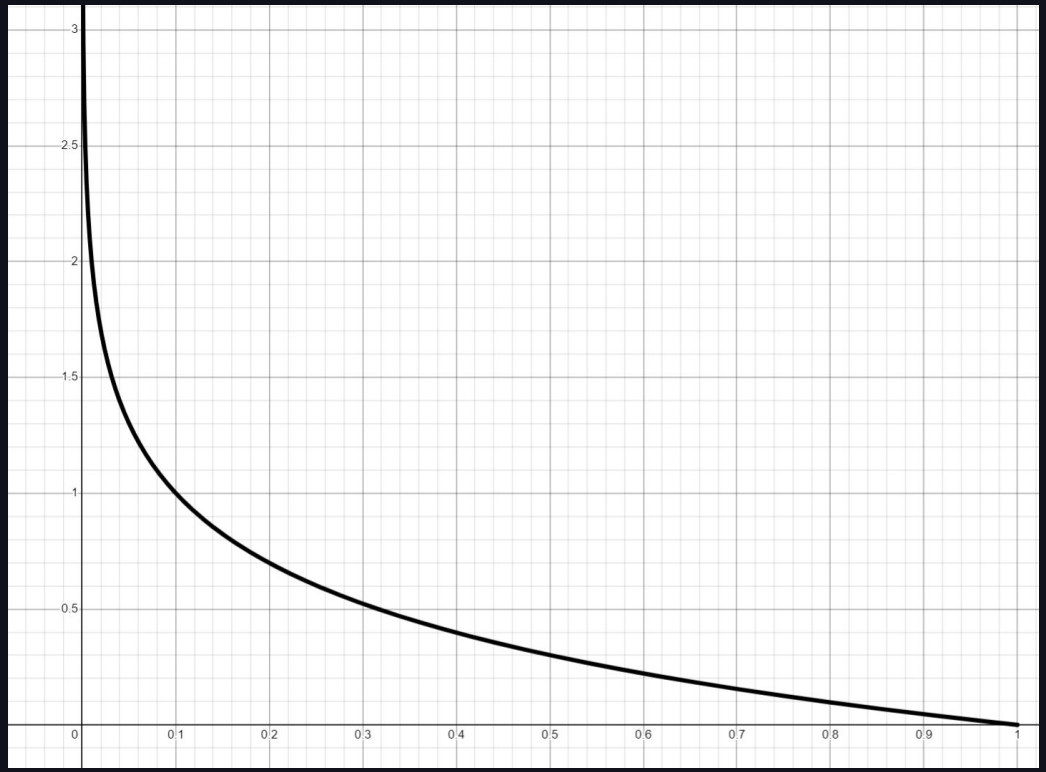
First, we go through all the samples in the batch, and we calculate the log of the prediction of the discriminator, based on the output of the generator from the input vector. Then we normalize it

Discriminator Loss Function:

$$L(D) = -\frac{1}{2m}\sum_{i=1}^{m}\log(D(x_i)) + log(1 - D(G(z_i)))$$

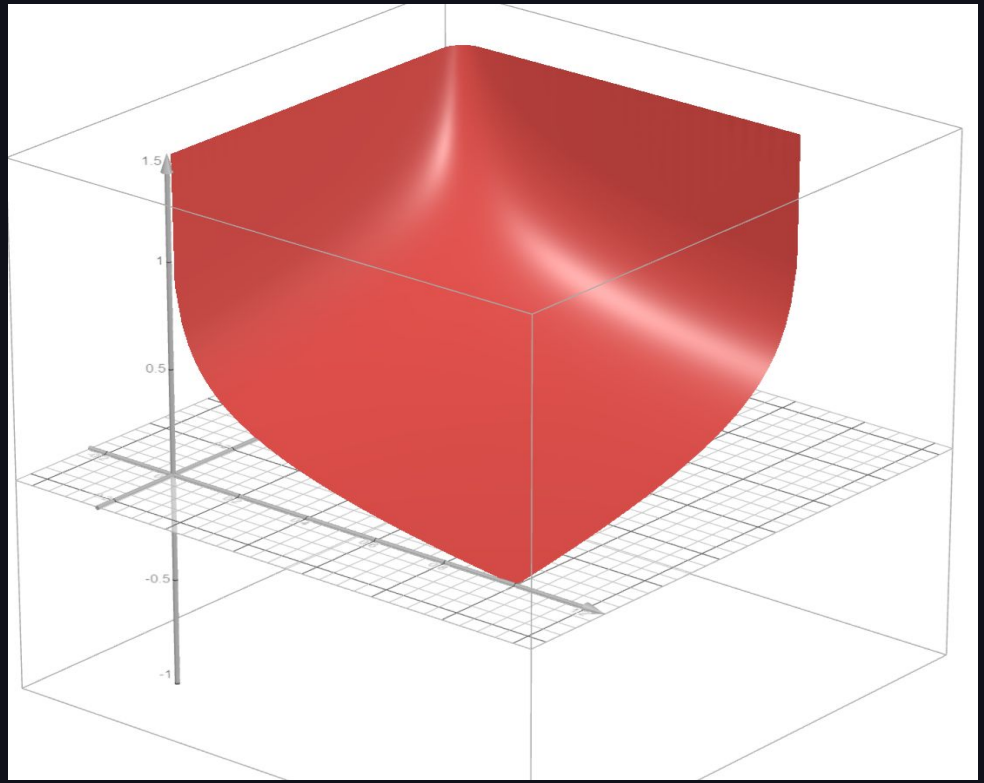
We calculate the real loss, which is the discriminator's predictions on the real images, and take the log of it's prediction. We do something similar for the fake loss, but subtract one.

Understanding Generator Loss



Here we can see, as the discriminator prediction goes closer to 1 (real image), the generator loss closer to 0, as the generator is trying to trick the discriminator, into thinking the generated image is real, since the discriminator predicts 1, it's doing a good job at it. Likewise, as the prediction gets closer to 0, the generator loss starts rapidly increasing

Understanding Discriminator Loss



As the discriminator's prediction for the real image gets closer to 1, and for the fake closer to 0, our loss is at its lowest. Meanwhile, if the prediction for the real image is 0, and for the fake it's 1, the loss is highest. This is just like the binary cross entropy loss function, as the discriminator needs to assign the correct label