Generative Adversarial Networks

Submitted By:

Meher Abhijeet [194102308]
Department of EEE
IIT Guwahati

ABSTRACT
This document will give an introduction to DCGANs through an example. A generative adversarial network (GAN) can be used to generate new objects after showing it pictures of many real objects. GANs are a framework for teaching a DL model to capture the training data's distribution so we can generate new data from that same distribution.

INTRODUCTION
GANs are made of two distinct models, a generator and a discriminator. The job of the generator is to produce fake images that look like the training images. The job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator. During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to correctly classify the real and fake images. The equilibrium of this game is when the generator is generating images that look as if they came directly from the training data, and the discriminator is left to always guess at 50% confidence that the generator output is real or fake.

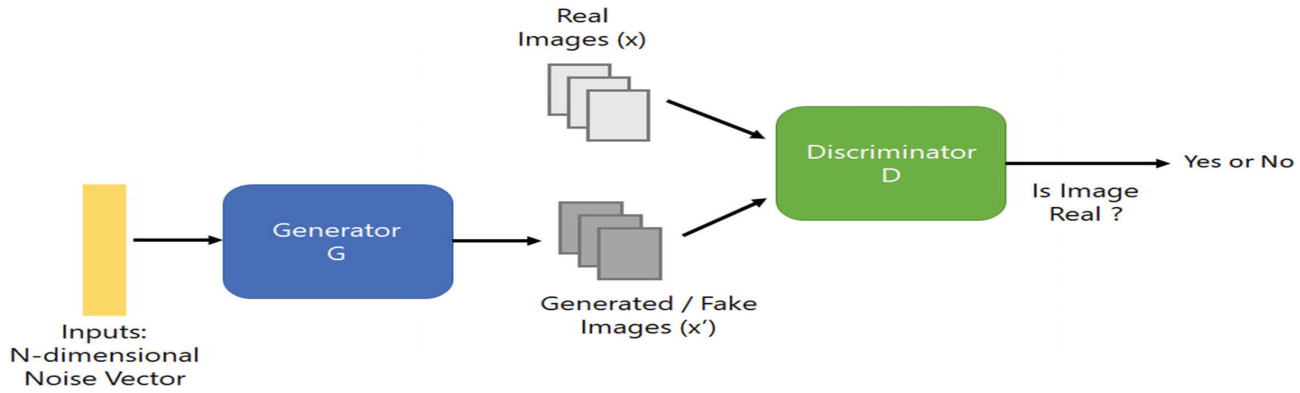

Figure 1. GAN Block Diagram

METHOD
Let x be data representing an image. D(x) is the discriminator network which outputs the probability that x came from training data rather than the generator. Here, the input to D(x) is an image from CIFAR-10 dataset . Preferably, D(x) should be high when x comes from training data and low when it comes from the generator.

For the generator , let z be a latent space vector sampled from a standard normal distribution. G(z) represents the generator function which maps the latent vector z to required data space. The goal of G is to learn the distribution that the training data comes from .

So, D(G(z)) is the probability that the output of the generator G is a real image. As described in Goodfellow's paper, D and G play a minimax game in which D tries to maximize the probability it correctly classifies reals and fakes (log

D(x)), and G tries to minimize the probability that D will predict its outputs are fake log(1-D(G(x))). From the paper, the GAN loss function is

$$min_G max_D(D,G) = E_{x\sim data}[logD(x)] + E_{z\sim latentspace}[log(1-D(G(z)))]$$

In theory, the solution to this is where estimated distribution = true disrtibution , and the discriminator guesses randomly if the inputs are real or fake.A DCGAN is a direct extension of the GAN described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

IMPLEMENTATION

Weight Initialization

From the DCGAN paper, the authors specify that all model weights shall be randomly initialized from a Normal distribution with $\mu = 0, \sigma = 0.02$.

Generator

It is designed using a series of strided two dimensional convolutional transpose layers, each paired with a 2d batch norm layer and a relu activation. The output of the generator is fed through a tanh function to return it to the input data range of [-1,1].

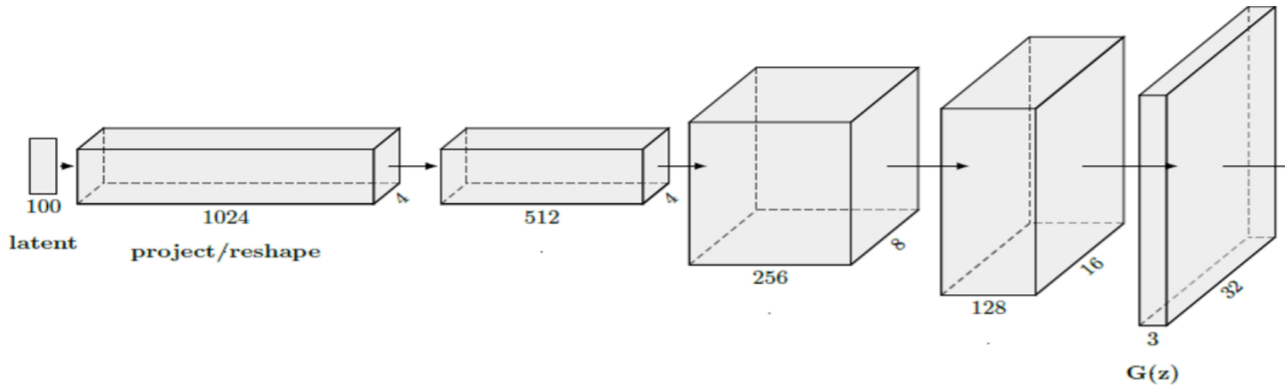

Figure 2. Generator Architecture

Discriminator

Here, D takes a 3x64x64 input image, processes it through a series of Conv2d, BatchNorm2d, and LeakyReLU layers, and outputs the final probability through a Sigmoid activation function.

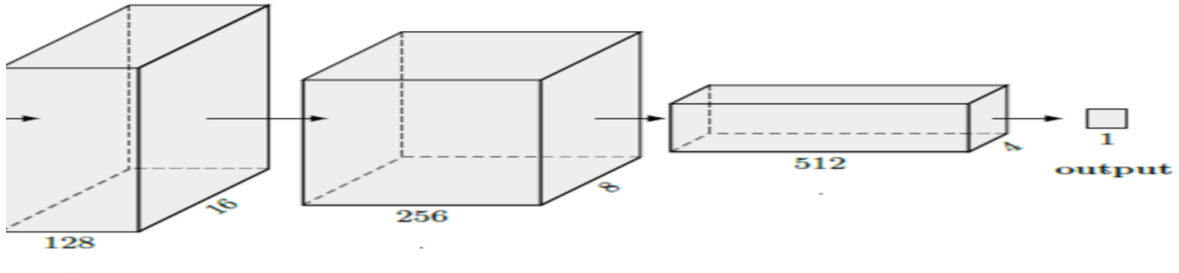

Figure 3. Discriminator Architecture

Loss Functions and Optimizers

As specified in the DCGAN paper, both are Adam optimizers with learning rate 0.0002 and Beta = 0.5. For keeping track of the generator's learning , a fixed batch of latent vectors that are drawn from a Gaussian distribution (i.e. fixed

noise) are used .

Training

First, a batch of real samples is constructed from the training set, forward passed through D , and the loss log(D(x)) is calculated, then the gradients are calculated in a backward pass. Secondly, a batch of fake samples are generated with the current generator, forward passed through D , and the loss log(1-D(G(z))) , its gradients are accumulated . Now , with the gradients accumulated from both the all-real and all-fake batches,a step of the Discriminator is called.
After computing G's loss using real labels and G's gradients in a backward pass, G's parameters are updated with an optimizer step.

RESULTS

A batch of real data next to a batch of fake data generated by G after 25 epochs are shown below .

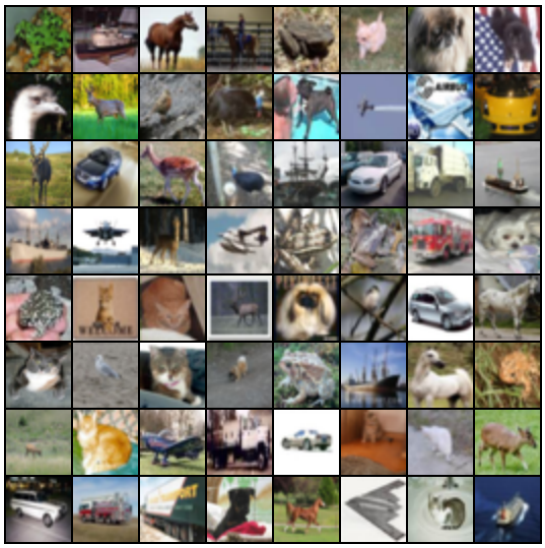

Figure 4. Real Images

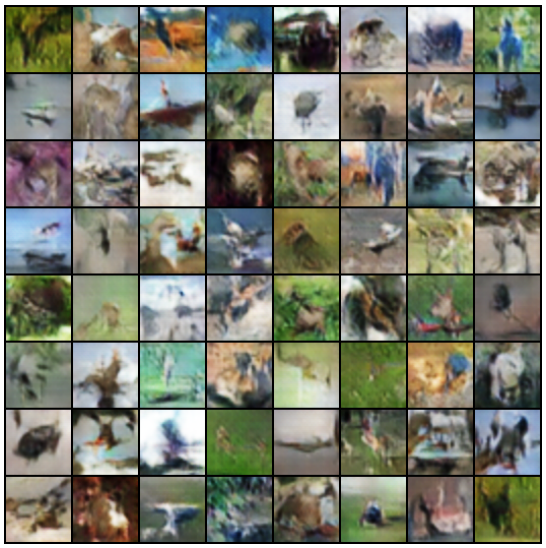

Figure 5. Fake Images

REFERENCES

1. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html.
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Advances in neural information processing systems, pp. 2672-2680, 2014.
3. HyungrokHam,TaeJoonJun, DaeyoungKim . UnbalancedGANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder
4. Alec Radford Luke Metz , Soumith Chintala .Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks .