# Deep Convolution Generative Adversarial Networks (DCGAN) for CIFAR10

Reetu Hooda
University of Alabama, Huntsville
*rh0059@uah.edu*

## I. Introduction

Generative Adversarial Networks (GAN) are an approach to generative modeling using deep learning methods, such as Convolutional neural network (CNN). The idea of GAN was inspired to make neural networks more human in nature by allowing it to create rather than just giving it visionary abilities. This project focuses on generating CIFAR-10 images using CNN based GAN called Deep Convolution GAN (DCGAN). DCGANs are more suitable for applications which requires generation of images/videos. The results shows that our network is successful in generating new images sharing similar patterns as that of the original images.

## II. The Dataset

CIFAR-10 training dataset is used for training that consists of 50,000 images of each size $32 \times 32$ of 10 classes with 5000 images per class. The test batch includes exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

## III. Network Architecture

GANs capture the training data's distribution so we can generate new data from that same distribution.GANs comprises of two separate neural networks called Generator and Discriminator. The generator generates fake images that looks similar to training images and discriminator finds whether the fed image is real or produced by the generator network. The generative model can be thought of as analogous to a team trying to duplicate an original art without getting detected while the discriminative model is analogous to the team trying to detect fake art. During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better detective and correctly classify the real and fake images. The equilibrium of this game is when the generator is generating perfect fakes 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.

### *A. Generator*

The Generator has 4 convolution transpose layers and 1 fully connected layer. The configuration of the network is shown in Fig 1. It is well known that the training time and total number of layers have linear relations in CNN architectures. In other words, increasing the number of layers increases the training time. Therefore, we are using GPU source for training the network. All the convolution layers use filters of size $3 \times 3$. Varying number of channels are used such as $256, 128, 64$ and $3$ to generate a 3-channel RGB image as the output of the generator. All the layers except last one employs LeakyReLU activation function. A Tanh activation function is used for the final layer to generate the image in range $[-1, 1]$. The input of the network is a random noise that is mapped to data space by the generator. During training, the generator learns to map the noise samples from the distribution closer to the distribution of the training images.

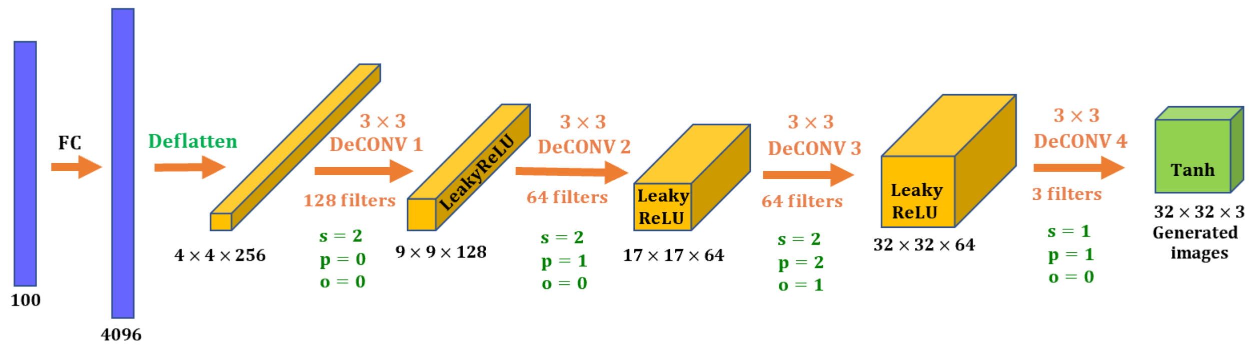

Fig. 1. Generator Architecture

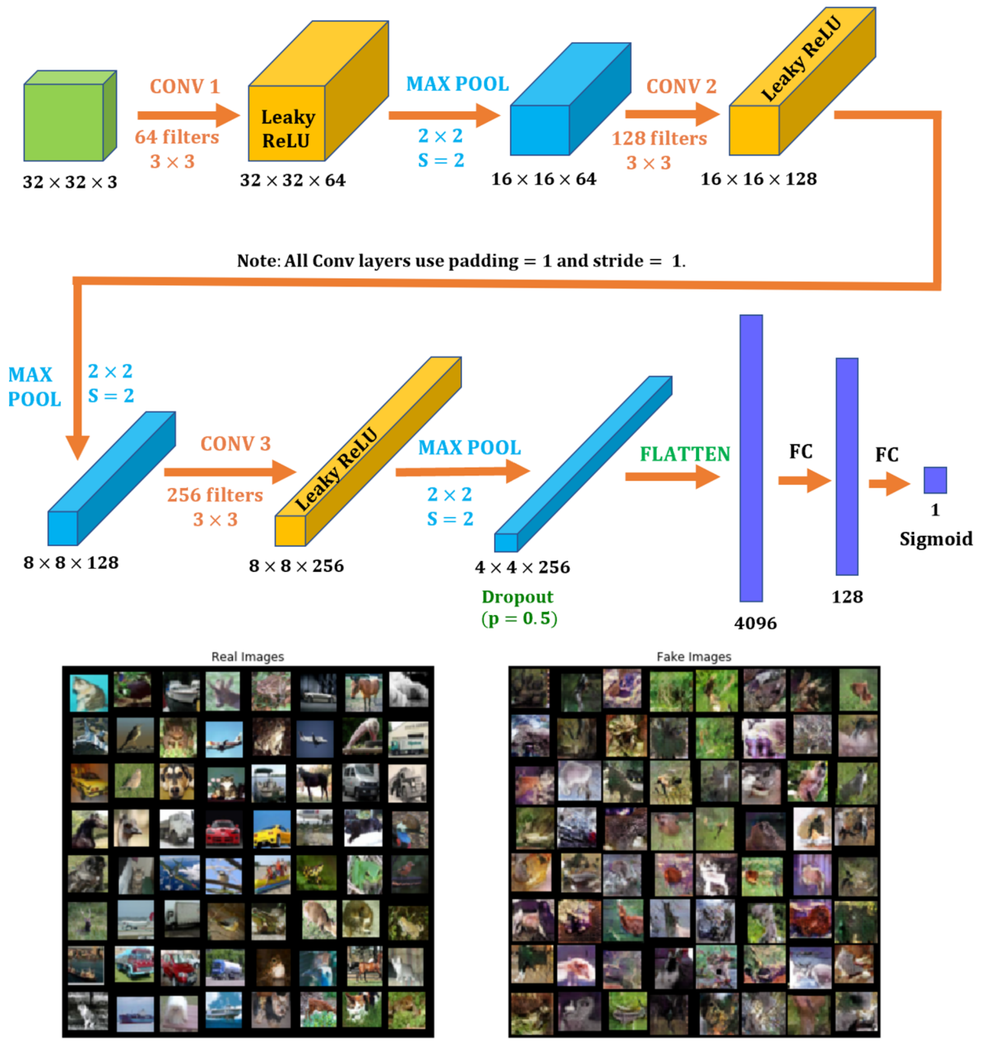

Fig. 2. Discriminator Architecture and Real Vs Fake Images

### B. *Discriminator*

The Discriminator which will be the opponent of Generator is fed with both the generated images as well as a real images at the same time, allowing it to learn to classify correctly as real or fake images. Therefore, it is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake).

After reaching a certain point called equilibrium point, the Discriminator will be unable to tell if the generate image is a real or a fake image, and that is when we can see images of a certain class(class that the discriminator is trained with.) being generated by out Generator that never actually existed before.

## IV. Results and Conclusion

The model was trained for 120 epochs. Fig 3 shows Generator and Discriminator Loss over each epoch.In this report we present a DCGAN architecture for generating CIFAR-10 images. Although the proposed model could not generate clear distinct images belonging to a certain class out of 10 classes, we can see that it was able to pick up the patterns that are similar to the original training images.

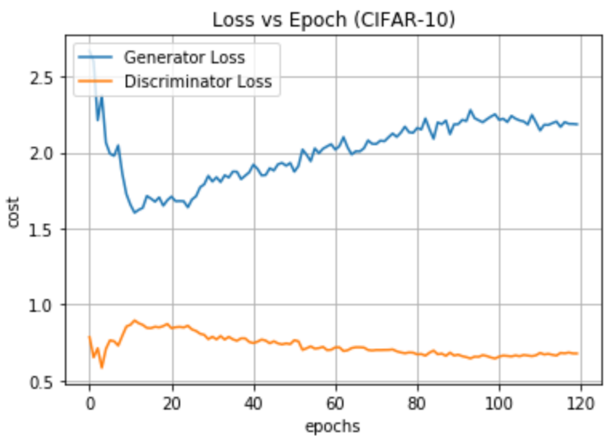

Fig. 3. Discriminator and Generator Loss Vs Epoch