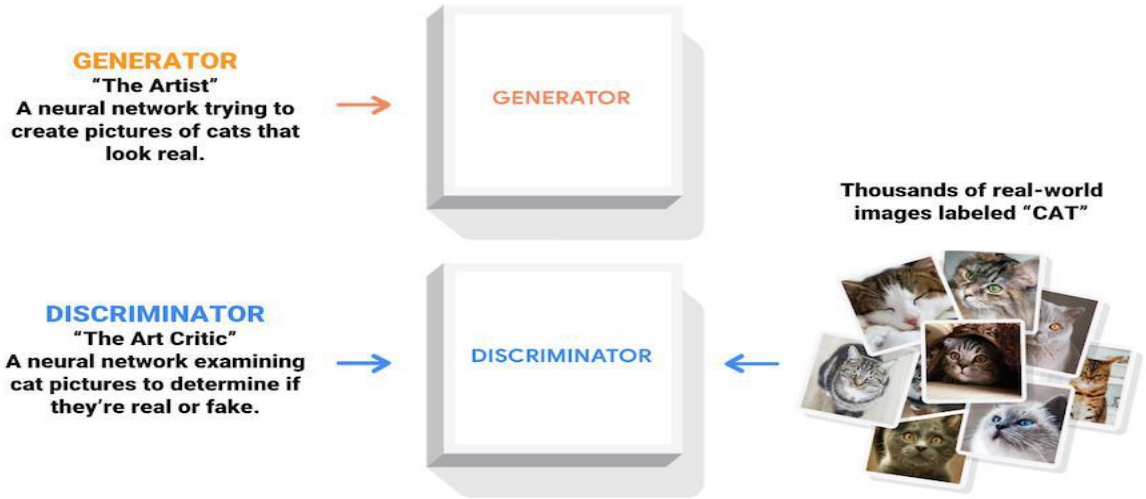
Semester: VIII Subject: Advanced AI Academic Year:2024-2025

Module 2

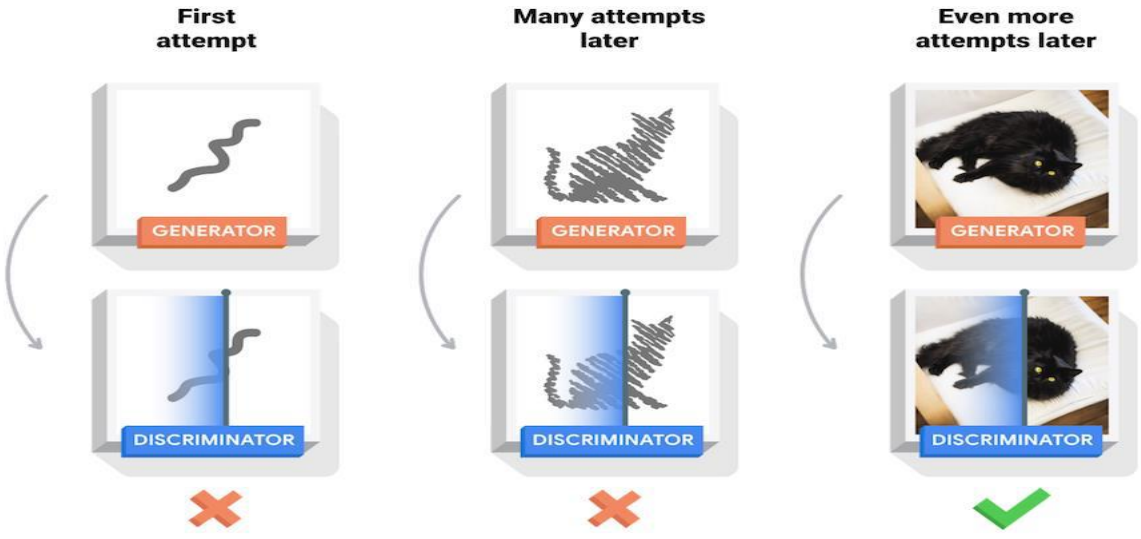
Deep Convolutional GAN(DCGAN)

What are GANs?

Generative Adversarial Networks (GANs) are one of the most interesting ideas in computer science today. Two models are trained simultaneously by an adversarial process. A generator ("the artist") learns to create images that look real, while a discriminator ("the art critic") learns to tell real images apart from fakes.

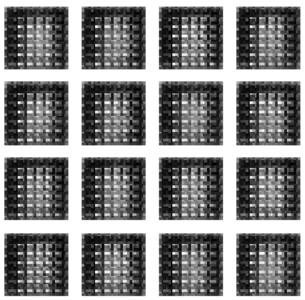


During training, the generator progressively becomes better at creating images that look real, while the discriminator becomes better at telling them apart. The process reaches equilibrium when the discriminator can no longer distinguish real images from fakes.



Semester: VIII Subject: Advanced AI Academic Year:2024-2025

On the MNIST dataset. The following animation shows a series of images produced by the generator as it was trained for 50 epochs. The images begin as random noise, and increasingly resemble hand written digits over time.



Deep Convolutional GAN (DCGAN) was proposed by a researcher from MIT and Facebook AI research. It is widely used in many convolution-based generation-based techniques. It focuses, to make training GANs stable. Hence, DCGAN comes up with some proposed architectural changes in the computer vision problems. In this article, we will be using DCGAN on the fashion MNIST dataset to generate images related to clothes.

Need for DCGANs:

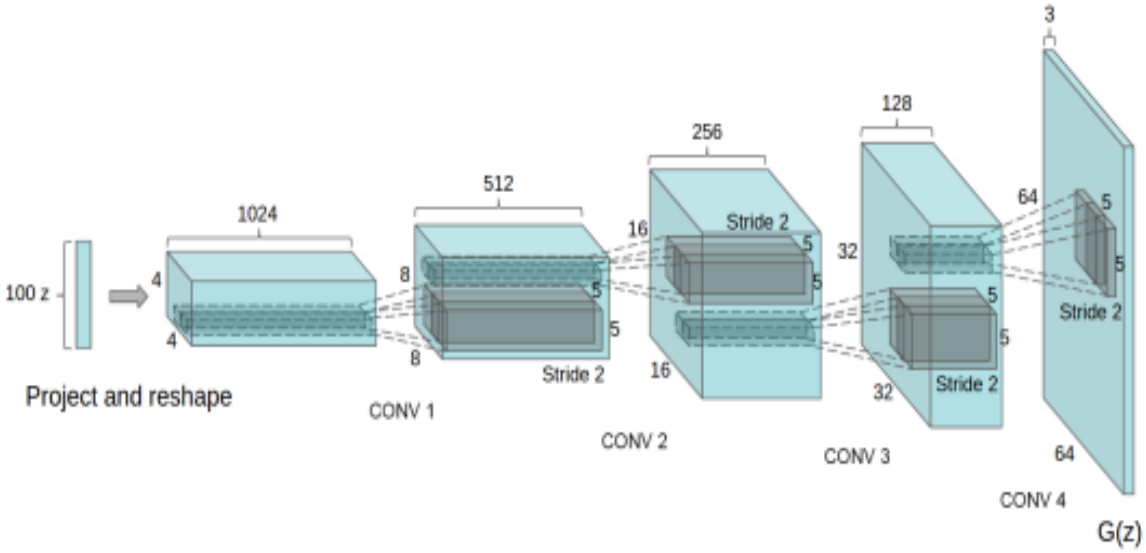
DCGANs are introduced to reduce the problem of mode collapse. Mode collapse occurs when the generator got biased towards a few outputs and can't able to produce outputs of every variation from the dataset. For example- take the case of mnist digits dataset (digits from 0 to 9) , we want the generator should generate all type of digits but sometimes our generator got biased towards two to three digits and produce them only. Because of that the discriminator also got optimized towards that particular digits only, and this state is known as mode collapse. But this problem can be overcome by using DCGANs.

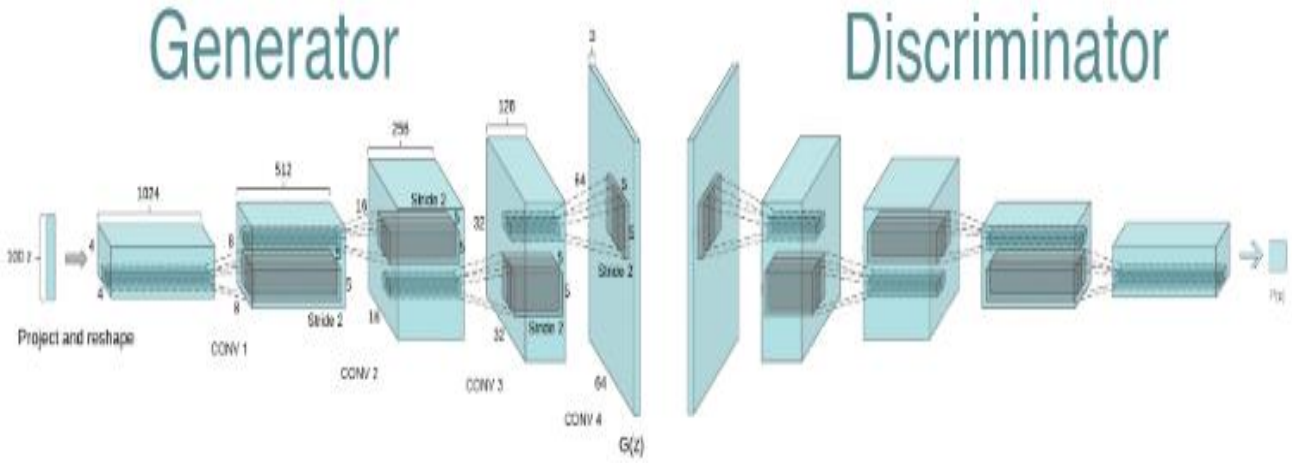
Architecture:

DCGAN, or Deep Convolutional GAN, is a generative adversarial network architecture. It uses a couple of guidelines, in particular:

- Replacing any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Using batchnorm in both the generator and the discriminator.
- Removing fully connected hidden layers for deeper architectures.
- Using ReLU activation in generator for all layers except for the output, which uses tanh.
- Using LeakyReLU activation in the discriminator for all layer.

Semester: VIII Subject: Advanced AI Academic Year:2024-2025





The generator of the DCGAN architecture takes 100 uniform generated values using normal distribution as an input. First, it changes the dimension to 4x4x1024 and performed a fractionally stridden convolution 4 times with a stride of 1/2 (this means every time when applied, it doubles the image dimension while reducing the number of output channels). The generated output has dimensions of (64, 64, 3). There are some architectural changes proposed in the generator such as the removal of all fully connected layers, and the use of Batch Normalization which helps in stabilizing training. In this paper, the authors use ReLU activation function in all layers of the generator, except for the output layers.

The role of the discriminator here is to determine that the image comes from either a real dataset or a generator. The discriminator can be simply designed similar to a convolution neural network that performs an image classification task. However, the authors of this paper suggested some changes in the discriminator architecture. Instead of fully connected layers, they used only strided-convolutions with LeakyReLU as an activation function, the input of the generator is a single image from the dataset or generated image and the output is a score that determines whether the image is real or generated.
Compared to the original GAN:

1. Replace pooling function with stridden convolutions
2. Get rid of fully connected layers on top of convolutional layers
3. Use LeakyReLu activation