**An overview of DCGAN architecture and its components**

# Intro

Data Scientists use Generative Adversarial Networks (GANs) for a wide range of tasks, with image generation being one of the most common. A particular type of GAN known as DCGAN (Deep Convolutional GAN) has been created specifically for this.

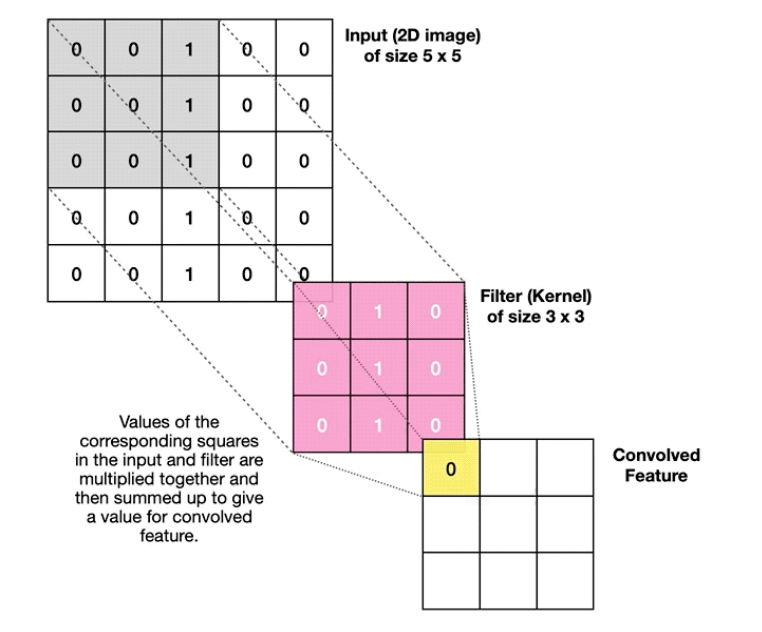
GAN only uses dense layers into its deep neural network architecture, since it does not use convolutional layers, we can create shapes available in python libraries but NOT necessarily recreate images in jpg or png format for example.

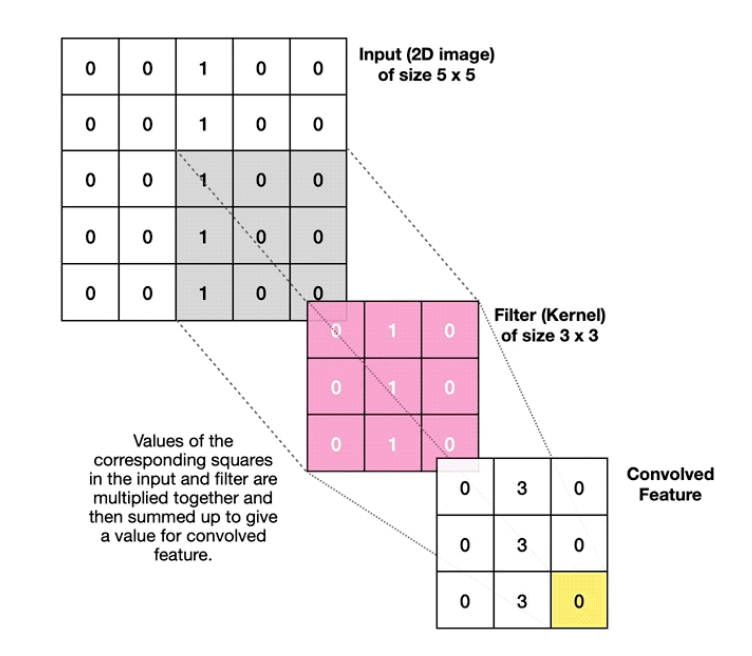
Let me start by highlighting that DCGAN utilizes **Convolutional** and **Transposed Convolutional** layers, which it strategically embeds into a **GAN architecture**.

## **Convolutional Layers**

Convolutions are a way to extract meaningful information from your data. They work particularly well with images allowing the network to learn the key features efficiently.

There are three parts to a convolution: Input (e.g., 2D image), a filter (a.k.a. kernel) and an output (a.k.a. convolved feature). The below image shows how we can apply a 3 x 3 filter on top of a 5 x 5 image to create a 3 x 3 convolved feature.





The above image uses a stride of (1,1)

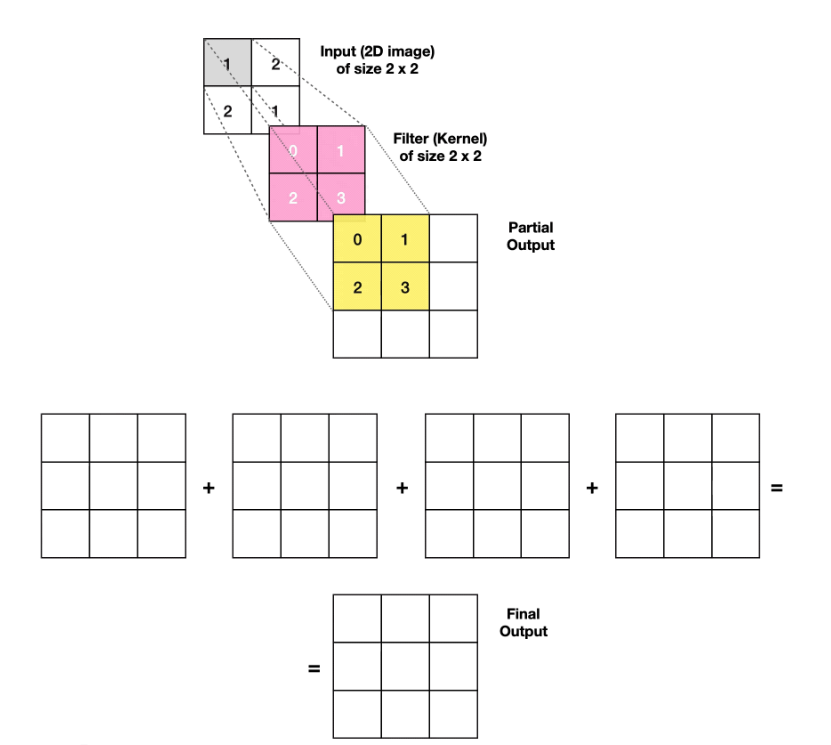
A neural network can contain multiple Convolutional layers and additional Pooling layers to reduce the size of convolved features.

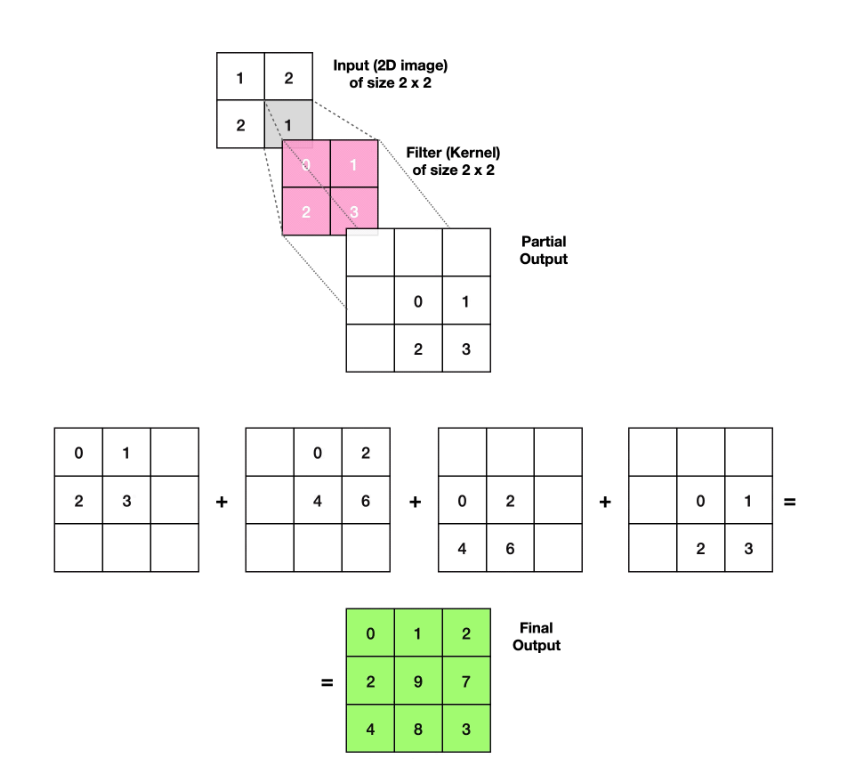
We will use Convolutional layers inside the **Discriminator model** and the final layer of the **Generator model** (more on this in the GAN section).

## **Transposed Convolutional Layers**

Transposed Convolutions also use filters to process the data, but their goal is the opposite of a regular Convolution. I.e., we use them to **upsample** the data to a larger output feature map, while regular Convolutions **downsample**.

The below image illustrates how the Transposed Convolution works. The example moves from a 2 x 2 input to a 3 x 3 output via a 2 x 2 filter using a stride of 1.





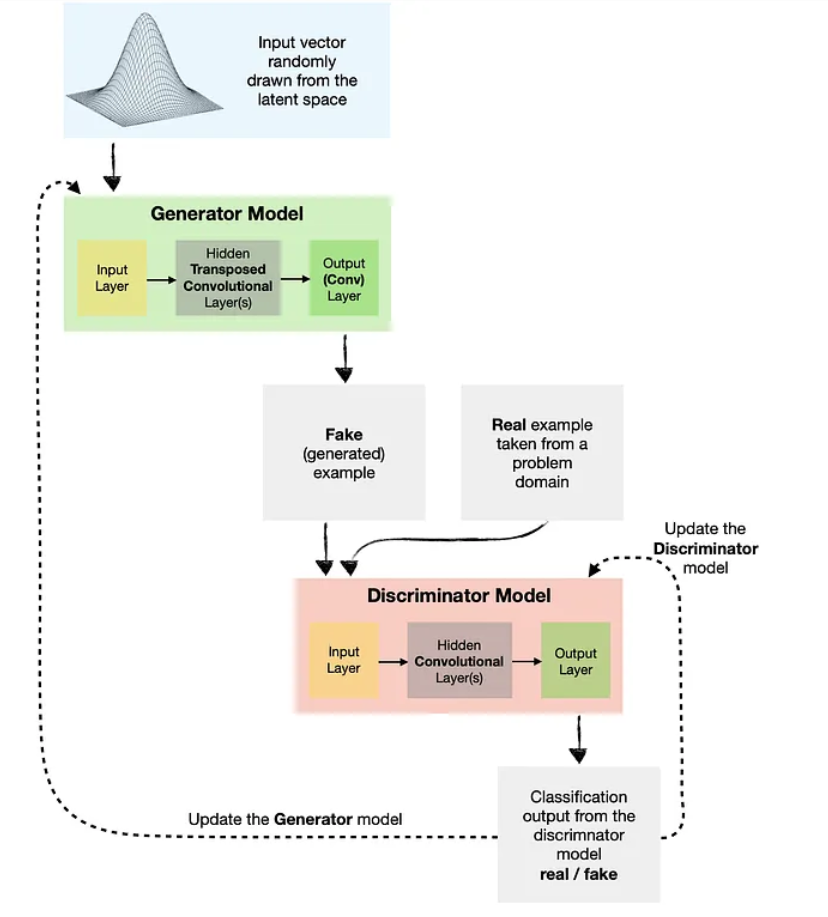
## DCGAN architecture

Now that we know what Convolutions and Transposed Convolutions are, let’s see how they fit into a GAN setup.

A Generative Adversarial Network (GAN) combines the **Generator** and the **Discriminator** models that **compete against each other**in a zero-sum game, making them adversaries.

The **Generator creates fake images** attempting to fool the Discriminator into believing they are real. Meanwhile, the **Discriminator learns the essential features** of the pictures in order to **differentiate** between real and fake examples.

The below diagram illustrates how the two models interact within the DCGAN architecture.



**As you can see, the Discriminator model is just a Convolutional classification model. In contrast, the Generator model is more complex as it learns to convert latent inputs into an actual image with the help of Transposed and regular Convolutions.**