**GENERAL ADVERSARIAL NETWORKS(GANs)**

In the given problem statement GANs can be used for a variety of reasons:

· Object-Specific Color Generation(specific colors for specific objects)

Eg. For example, a GAN trained on sky images could generate realistic sky colors

· If the original black and white images are low resolution, GANs can be used to upscale them while colorizing

· In areas where object recognition is uncertain, GANs could generate multiple plausible colorizations

· After an initial colorization pass (perhaps using a simpler method), a GAN could be used to refine and enhance the results

**PIX2PIX GAN:**

This is designed to learn a mapping from an input image to an output image, making it highly versatile for various image processing tasks.

In this GAN we learn the mapping from input image to output image and also learn a loss function. If we take a naive approach and ask the CNN to minimize the Euclidean distance between predicted and ground truth pixels, it will tend to produce blurry results. This is because Euclidean distance is minimized by averaging all plausible outputs, which causes blurring

So the architecture for this type of GAN consists of a U-NET which is a combination of an encoder and a decoder.

U-NET is characterized by its U-shaped architecture, consisting of a contracting path (encoder) and an expansive path (decoder).

· Encoder is responsible for extracting features from the input images

· Decoder is responsible for up sampling intermediate features and producing the final output

· The green arrows represent skip connections, a key feature of U-NET.

· These connections directly link corresponding layers in the contracting and expansive paths.

· They allow the network to combine low-level, fine-grained spatial information with high-level semantic information.

Essentially a U-NET is a CNN with a Encoder-Decoder type architecture

**ENCODER:**

· Consists of repeated 3x3 Conv Layers + ReLU Layers

· 2X2 max pooling layers to down-sample

After each set of convolutions, max pooling reduces the spatial dimensions by half.



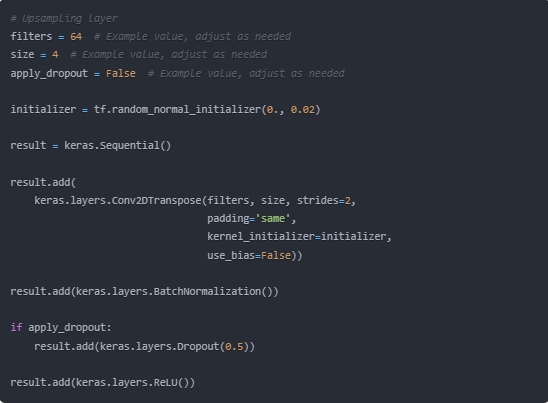
This helps in: a) Increasing the receptive field (area of input influencing each output). b) Reducing computation by decreasing feature map size.

During down-sampling the spatial dimensions get reduced and to compensate for this the channels get doubled after each downsampling operation. Typically, you might see channel progression like 64 -> 128 -> 256 -> 512.

**DECODER:**

· Reverse of the decoder and has the same architecture as the encoder

· Instead of down-sampling the decoder up-samples and therefore the channels get halved thereby nullifying the change made in the encoder finally giving the same number of channels as the input image



**CONNECTIONS BETWEEN ENCODER AND DECODER:**

**1.** **Connecting Paths**

· Direct connections between corresponding layers of the encoder and the decoder of the U-Net

· Features from the encoder layers are concatenated with corresponding decoder layers

· Enables more precise localization in the final output

· There are usually 4 or 5 skip connections in a U-Net and earlier connections carry more spatial info(position/layout of features in an image) while the later layers carry more semantic info(info about objects/scenes/actions in the image)

**2.** **Bottle-Neck**

· The bottleneck is the lowest part of the 'U' shape in U-NET, where the spatial dimensions are smallest and the number of channels is highest.

· Basically the transition between the encoder and the decoder and has the smallest spatial dimensions and highest number of feature channels

· This part forces the network to learn the most important features for the task at hand

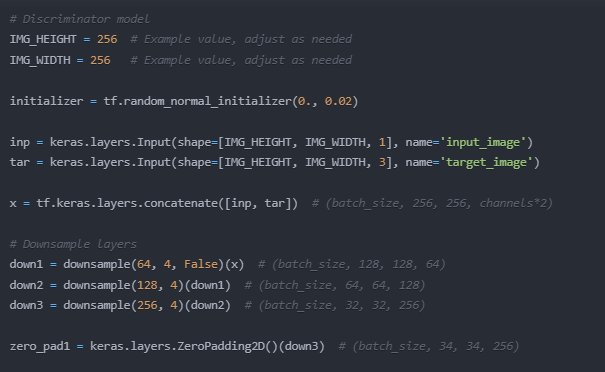
The combination of the bottleneck and connecting paths allows U-NET to effectively capture both global context (through the bottleneck) and fine spatial details (through skip connections), making it particularly effective for tasks like image segmentation and image-to-image translation in Pix2Pix GAN.

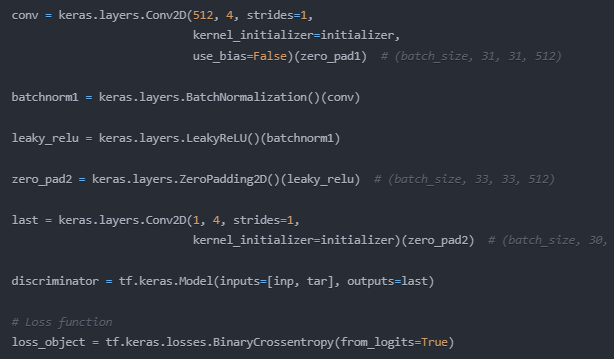
Any GAN is basically a battle between the Generator and the Discriminator

In Pix2Pix , the Discriminator is actually a PatchGan. The discriminator tries to classify is each NxN patch in an image is real or fake and penalizes structure at the scale of patches.

This is advantageous because a smaller PatchGAN has fewer parameters, runs faster and can be applied to arbitrarily large images.

Sample code for the Discriminator is shown:





**IMPLEMENTATION AND WORKING OF PIX2PIX:**

Example problem: Creating fake images that are indistinguishable from real images

As discussed earlier the architecture of the Generator is a U-NET with upsampling layers followed by downsampling layers that include skip connections as well

Discriminator is basically just a couple of CNN layers

(CNNs)

Image —-> Nx1x30x30 (Assuming 256x256 input image size)

(30x30 is the size of the PatchGAN)

(More about PatchGANs is in the PatchGAN file)