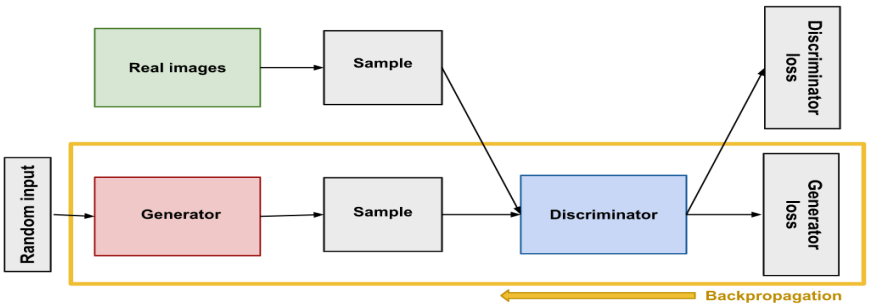
dNote: For all models 80 epochs recommended as minimum around 250 max

Pure Image Generator

GAN - Generative Adversarial Network

The name for the basic concept of having a 2 part neural network, a generator and a discriminator. The generator starts with a random input and produces an output in the same shape as the training set. The discriminator takes examples of both the training set and the output of the generator and tries to categorise them into real (training data) or fake (generated). If the discriminator can do this easily the weights of the generator are updated a lot but if it can't the discrimator's weights are updated a lot. In the end we take just the generator to use in a project.

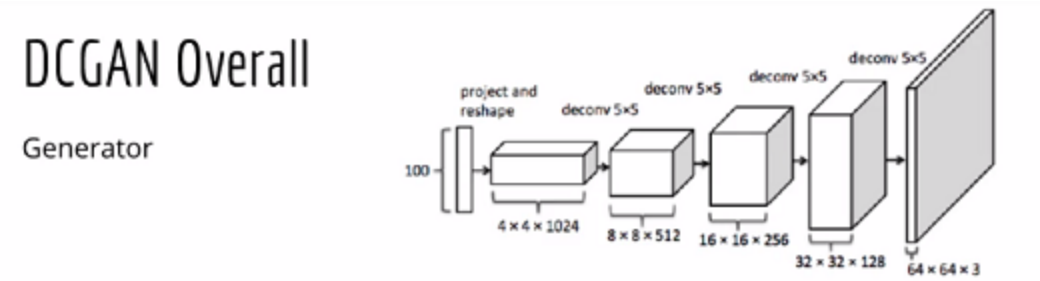


DCGAN - Deep Convolutional GAN

While a GAN is just the underlying concept, DCGAN is a type of architecture of a GAN. It uses Convolutional-Transpose layers to turn the random seed input into something like the training data. DCGANs use just a large bank of images of a certain type as a training set (no labels)

DCGANs have multiple blocks of repeating layers. These blocks start with a convolution-transpose layer and then pass into a batch norm 2d layer and then to a ReLU layer. Convolution-transpose layers change the shape or size of an input but change it slightly adding some features from the weights of the layer. Batch norm 2d layer makes it easier to find approximately the "correct" weights which speed up training. Finally the ReLU layer allows the output to not so heavily depend on input so if you get a really weird random input the output is still similar to the training data.

The number or size of these blocks can change between implementations but generally the first one up-samples the random input and every one after down-samples until you have reached the shape of the training data. In this context upsampling is massively increasing the size of the input since the random input is usually much smaller than the desired output size and we down-sample to allow us to impart the trained weights onto the data



Binary GAN

Binary GANs are essentially the same as DCGANs but they can only work with binary data. In the case of images this means black and white images only. This does however come with a boost in runtime performance and precision of detail with a reduction in training time

Prompt Image Generator

cGAN - Conditional GAN

Conditional GANs extend off of DCGANs. The training dataset is now labelled but unlike traditional classifiers each datapoint can have multiple attributes associated with it. Each combination of attributes encodes for a different tensor. The generator is now fed 2 things; The random input just as with DCGANs but also a list of randomly generated attributes encoded to a tensor. The random input goes through the first upsampling block but is then concatenated with this encoded tensor. As the result goes through the rest of the model the features that are imparted by the weights differ for each of these encoded tensors.The discriminator is also fed the encoded tensor to better help it discriminate the real and fake samples. Layers in a cGAN are modified from DCGAN but not majorly different as it's not a complete restructuring just modifying to include this attribute tensor. The main work is in loading and encoding the attributes before entering the model

cGANs allows us to generate outputs with specific attributes but does require larger training time to learn the weights for all the attributes.

Image to Image Generator

Image to image models are a subset of cGANs which are specially purposed to take in an image as the encoded attribute tensor. Each image in the training set instead of having a list of attributes now is paired with a before image. For example a bridge dataset might have each

picture of a bridge paired with a sketch of all the lines of said bridge.

The specific architecture changes but most if not all are built upon a U-Net architecture. The idea with this architecture is there are 2 parts: the contracting path (Encoder) and the expansive Path (decoder).

The contracting path is similar in function to a CNN where the image is passed through convolutional layers to capture all of its high level features while downsizing the image. The blocks here are made from a convolutional 2D layer to capture features, a batch norm to help speed up training and a LeakyReLU - LeakyReLU has the same function as ReLU but is better as it solves some issues with normal ReLU.

The expansive Path takes the downsized image and upscales it adding in the features from the learned weights (similar in function to a cGAN) The blocks here are made from a convolutional-transpose, batch norm and ReLU layers just like the DCGAN and cGAN. There are usually 8 or more blocks in each path.

This can yield some stunning results like constructing building faces from shapes but is also the most complex model and will take the longest to train.