**Steps to train a GAN**

**Step 1**: Define the problem. Do you want to generate fake images or fake text. Here you should completely define the problem and collect data for it.

**Step 2**: Define architecture of GAN. Define how your GAN should look like. Should both your generator and discriminator be multi layer perceptrons, or convolutional neural networks? This step will depend on what problem you are trying to solve.

**Step 3**: Train Discriminator on real data for n epochs. Get the data you want to generate fake on and train the discriminator to correctly predict them as real. Here value n can be any natural number between 1 and infinity.

**Step 4**: Generate fake inputs for generator and train discriminator on fake data. Get generated data and let the discriminator correctly predict them as fake.

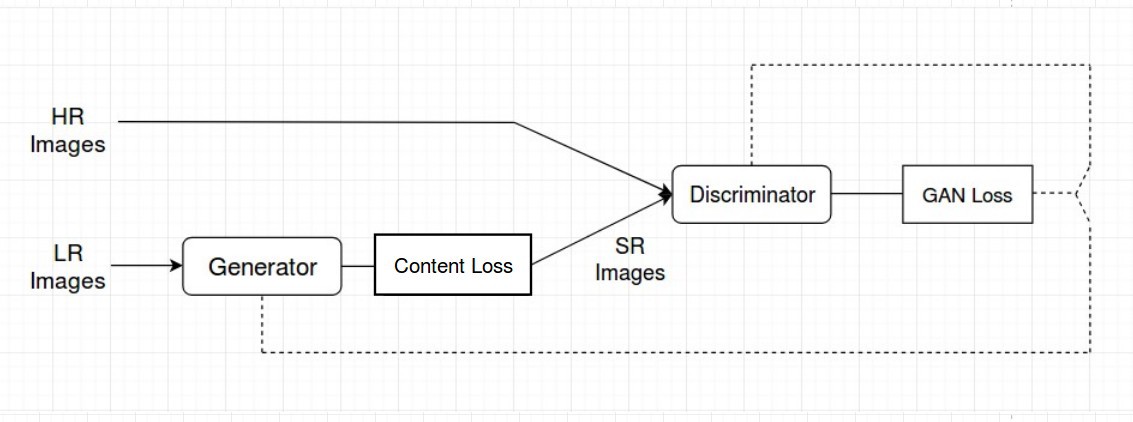
**Step 5**: Train generator with the output of discriminator. Now when the discriminator is trained, you can get its predictions and use it as an objective for training the generator. Train the generator to fool the discriminator.

**Step 6**: Repeat step 3 to step 5 for a few epochs.

**Step 7**: Check if the fake data manually if it seems legit. If it seems appropriate, stop training, else go to step 3. This is a bit of a manual task, as hand evaluating the data is the best way to check the fakeness. When this step is over, you can evaluate whether the GAN is performing well enough.

**Architecture of SRGAN**

SRGAN is similar to GAN, the Super Resolution GAN also contains two parts Generator and Discriminator where generator produces some data based on the probability distribution and discriminator tries to guess weather data coming from input dataset or generator. **Generator** than tries to optimize the generated data so that it can fool the discriminator. The generator architecture contains residual network instead of deep convolution networks because residual networks are easy to train and allows them to be substantially deeper in order to generate better results. This is because the residual network used a type of connections called skip connections. The task of the discriminator is to discriminate between real HR images and generated SR images. The **discriminator** architecture used in this paper is similar to DC- GAN architecture with LeakyReLU as activation. The network contains eight convolutional layers with of 3×3 filter kernels, increasing by a factor of 2 from 64 to 512 kernels. Strided convolutions are used to reduce the image resolution each time the number of features is doubled. The resulting 512 feature maps are followed by two dense layers and a leakyReLU applied between and a final sigmoid activation function to obtain a probability for sample classification.



The SRGAN uses perpectual **loss function** (LSR) which is the weighted sum of two loss components : content loss and adversarial loss. This loss is very important for the performance of the generator architecture:

Content Loss: We use two types of content loss in this paper : pixelwise MSE loss for the SRResnet architecture, which is most common MSE loss for image Super Resolution. However MSE loss does not able to deal with high frequency content in the image that resulted in producing overly smooth images. Therefore the authors of the paper decided to use loss of different VGG layers. This VGG loss is based on the ReLU activation layers of the pre-trained 19 layer VGG network.

Adversarial Loss: The Adversarial loss is the loss function that forces the generator to image more similar to high resolution image by using a discriminator that is trained to differentiate between high resolution and super resolution images.