RegGAN Summary

Ahmad Shayaan
as5948

{ahmad.shayaan@columbia.edu}@columbia.edu
Columbia University
December 4, 2020

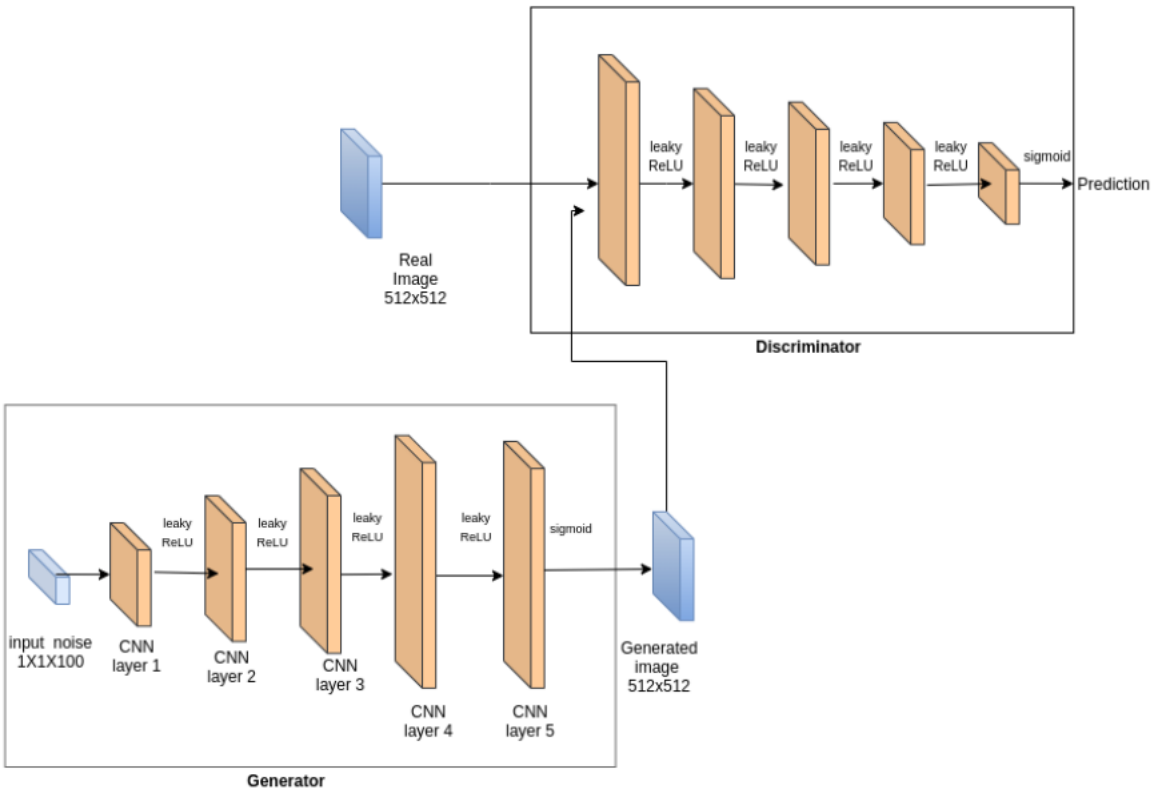
Summary

We attempted to use Generative Adversarial Networks to generate images with a given number of connected components. We tried various different architectures, regularization schemes, and training paradigms to achieve the task. We also created a synthetic data set to train our network. We created collections of "blobs" ranging between 11-18 in number in every image

Unsuccessful Attempts

Deep Convolutional Generative Adversarial Networks

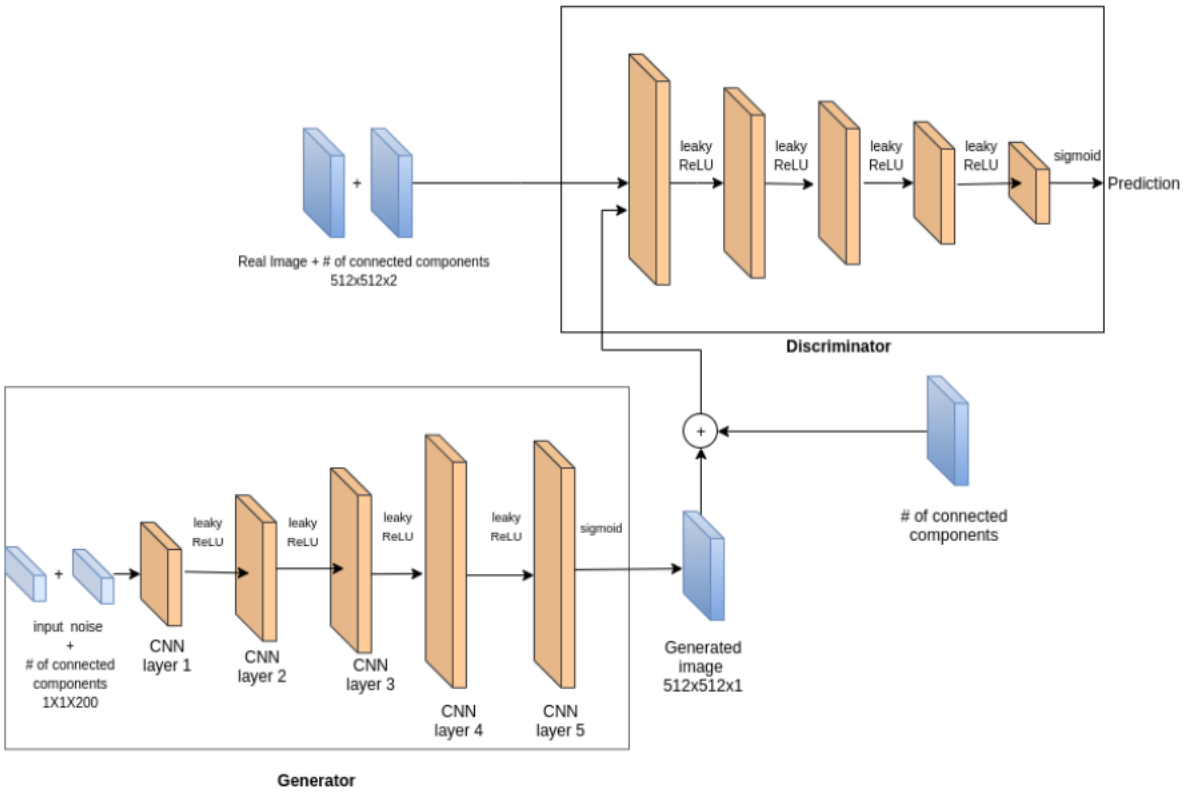
The first architecture that we implemented was a *Deep Convolutional Generative Adversarial* (DCGAN) network to generate the images with connected components. The architecture is shown below.



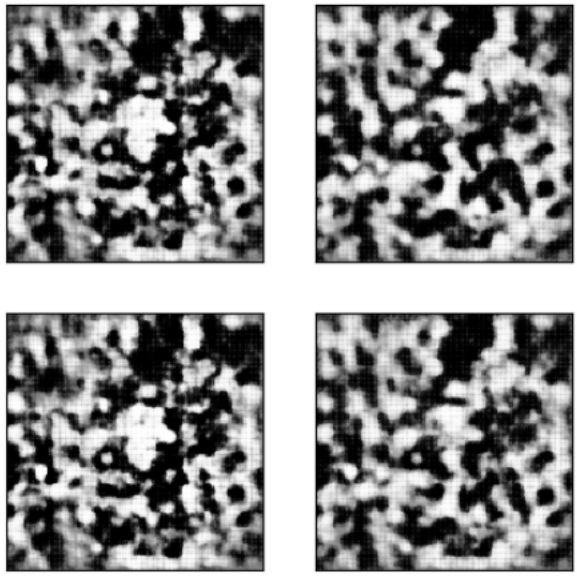
This network was able to generate images, however it did not capture the connected components in the image.

Conditional Deep Convolutional Generative Adversarial Networks

We tired to use a Conditional Deep Convolutional Generative Adversarial Networks (CDCGAN) to generate the images. CDCGAN are an extension of the vanilla DCGAN in which both the discriminator and the generator are conditioned on some extra information. The conditioning of the model on the extra information gives us control on the type of data that we want to generate. We condition the network on the number of connected components that we want in the generated image. The high level architecture of the CDCGAN is shown below.



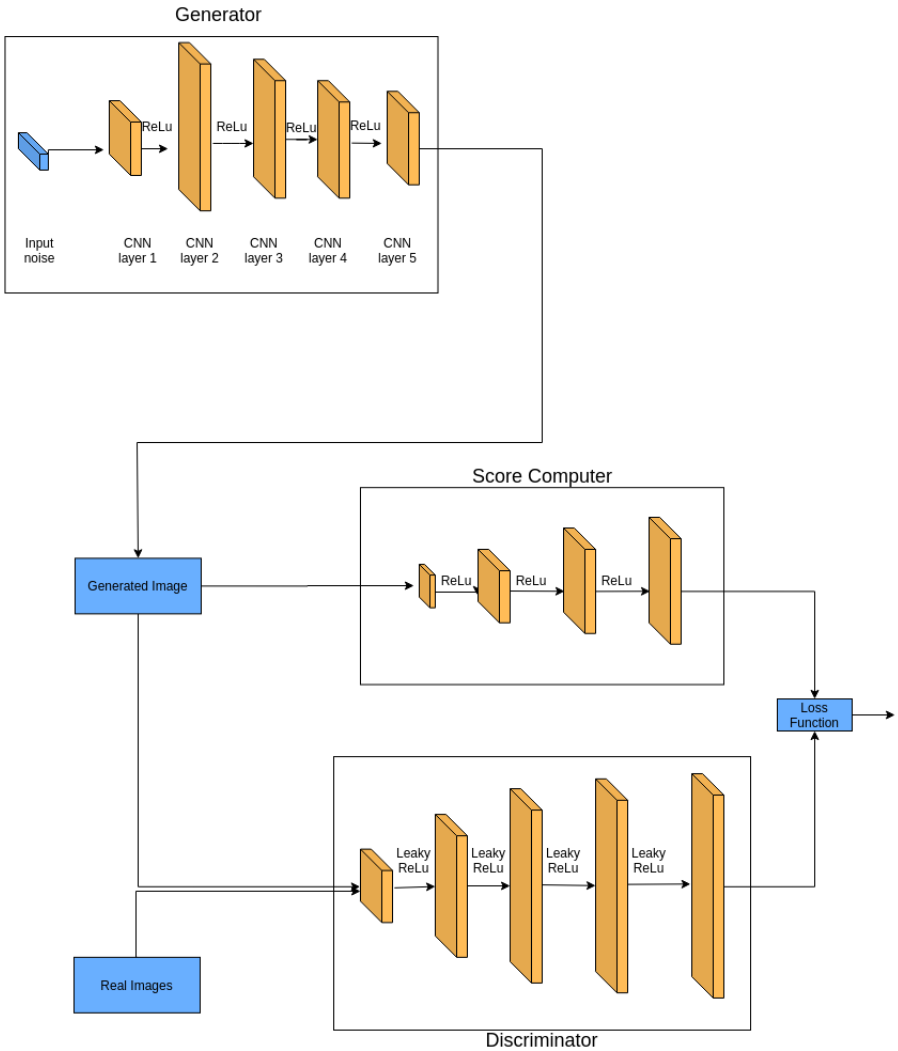
However the CDCGAN was not able to sufficiently capture the structures of the images and generated images as shown below.



Epoch 22

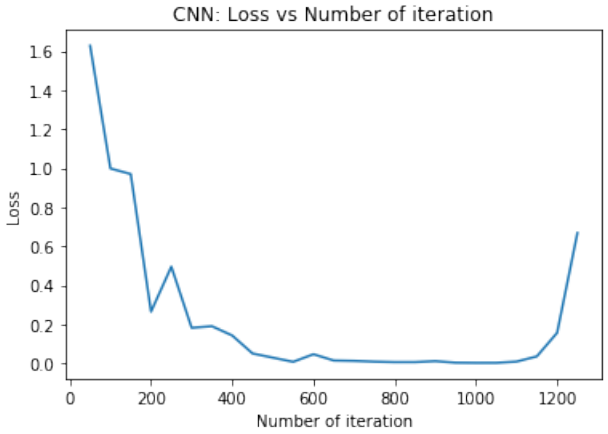
RegGAN

The RegGAN architecture consists of two discriminator and a single generator. The second discriminator is used to simulate the score function, which was designed by us. The first discriminator is used to differentiate between the images generated by the network and the ones from the dataset. The architecture can be seen below.



Pre-Training of Discriminator

We pre-train the discriminator to learn the score function. This is done so that the second discriminator has a good starting point for the actual training of the network. The discriminator is a CNN with four layers.
During pre-training we feed in the images from the data set to the network. The outputs from the network are then compared to the actual scores given by the score function. The results from the training of the network are shown below.



We then save the trained network and load it when we train the generator.

Training RegGan

The Generator and discriminator are also CNN both with 5 layers. We feed in noise to the generator and get outputs as images. These images are then fed into both the discriminators to compute the score and to compare it to the images of the actual data set.

We have two ways in which we back-propagate. In the first one we freeze the weights of the second discriminator and the gradient is only propagated through the the generator and the first discriminator. In the second method we pass the gradient through the second discriminator as well.

A sample of the images generated buy the network can be seen below.