Conor Sweeney (cjs2201) and Roberto Valdez Ponce (rjv2119)
Deep Learning Systems Performance Fall 2020
Professor Dube
December 14, 2020
Final Project

Complex Representations in GANs

Executive Summary:

- **Goal:** To attempt to capture complex representations using a GAN
- **Approach:** Using simple GAN architecture on increasing complex image sets to see if the GAN is able to capture the intricacies of the data.
- **Benefit:** Improved GANs can be used to solve increasing complex problems.

Background:

GAN stands for Generative Adversarial Network. It is a zero-sum game where a generative neural network competes against a discriminative neural network. The generator creates images using the same statistical probabilities of the features in the training set in hopes of fooling the discriminator into classifying the fake image as real. This facilitates unsupervised training since both networks are updated dynamically as they compete.

When properly designed and trained, GANs can be extremely powerful and generate very lifelike images:

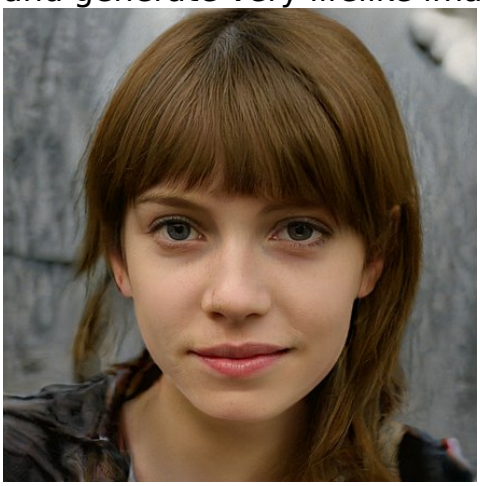


Figure 1: A generated image from SyleGAN based on portrait training images.

This concept was originally designed by Ian Goodfellow and his team in 2014 in their aptly named paper "Generative Adversarial Nets". Since then GANs have been an active field of AI research.

Technical Challenges:

We faced a lot of technical challenges with our project. There were a lot of general design decisions for our experiment which we needed to make such as: What type of GAN? Where do we get the data? How do we measure success? How do we generate a successful GAN in a short time frame?

More importantly, there are a lot of training challenges which we knew that we would need to face:

- The GAN suffers mode collapse. This is when the generator only outputs a small amount of outputs in order to trick the discriminator. Such as only outputting 8's in order to trick our number discriminator.
- The GAN suffers from vanishing gradients. This happens if a discriminator is too good then the generator can fail due to vanishing gradients. Vanishing gradients essentially mean that the weight stops updating its value.
- Nonconvergence and instability due to inappropriate design of network architecture.
- Selecting an appropriate objective function.
- Selecting the best optimization algorithm.

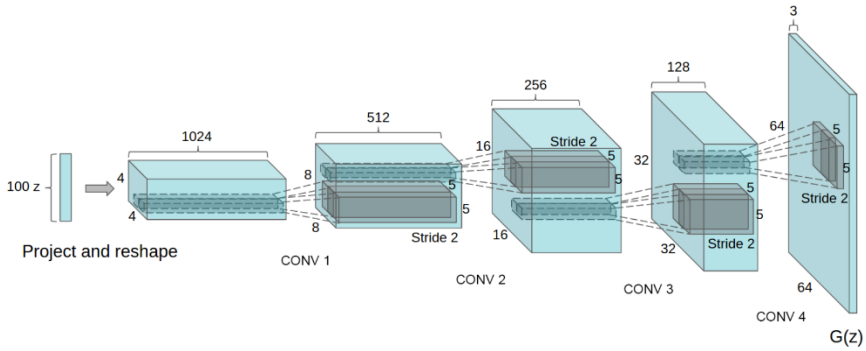
Approach:

We decided to choose a simple approach. We used a Vanilla GAN, measured performance based on loss and human visual analysis (we looked at the generated images and decided if they looked realistic or not), and 3 datasets:

- Simple: hand written numbers (http://yann.lecun.com/exdb/mnist/)
- Moderate: portrait images from CelebA (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)
- Complex: impressionistic art (https://www.wikiart.org/en/paintings-by-style/impressionism#!#filterName:all-works,viewType:masonry)

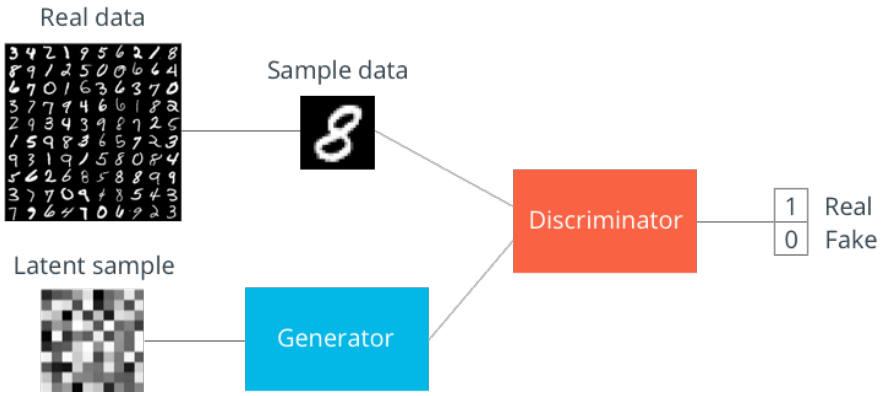
***Solution diagram/architecture*:**

Technically speaking, for the generator we are using a Deep Convolutional GAN:



Which is basicaly an inverted CNN. It starts with a z-vector of random noise and gradually builds up to an image.

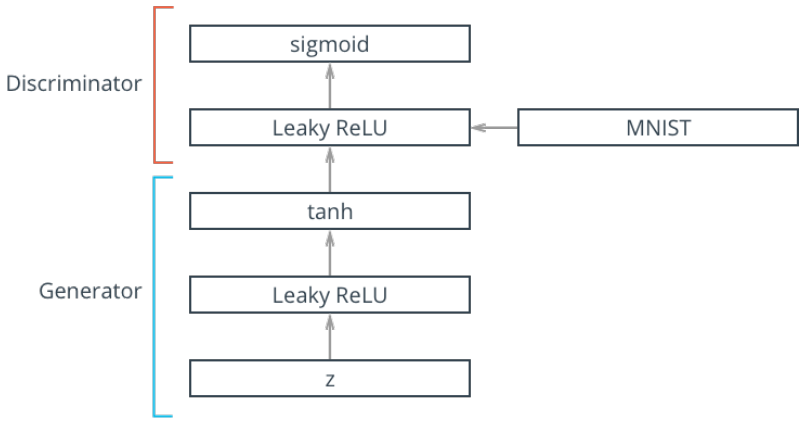
In the case of the Discriminator it is a CNN that ends up with a logit node to model the probability of being a real vs fake image.



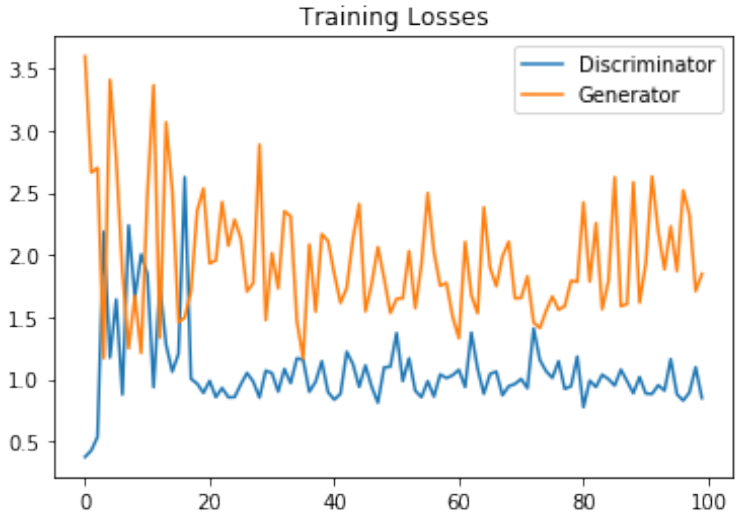
Implementation details*:*

We train the generator and discriminator at the same time. At each stage we feed both a batch of real images and fake (generated by the Generator) to the discriminator. For backpropagation, in the case of the fake images, the Generator is aware of the error outputed by the discriminator and the structure is trained as a whole. The discriminator is also trained alone on the errors obtained from the real data.

We ues a sigmoid activation functino in the last layer of the discriminator and hyperbolic tangent for the generator. We show how this structure looks like in the caseo f the MNIST data:



The training losses for both the generator and discriminator have big oscillations at first but then they converge to a lower state:

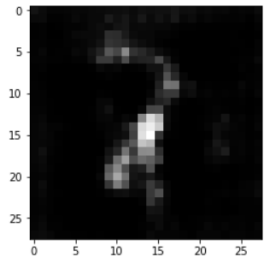


Demo/Experiment design flow:

We preprocessed this 3 datasets, and feed them to our GAN. While training the model printed the current output which let us experience the improvements that the model made on real time:

```
with tf.Graph().as_default():
    train(epochs, batch_size, z_dim, learning_rate, beta1, mnist_dataset.get_batches,
          mnist_dataset.shape, mnist_dataset.image_mode)
```

```
Epoch 1/4... Discriminator Loss: 0.9540... Generator Loss: 1.3431
```

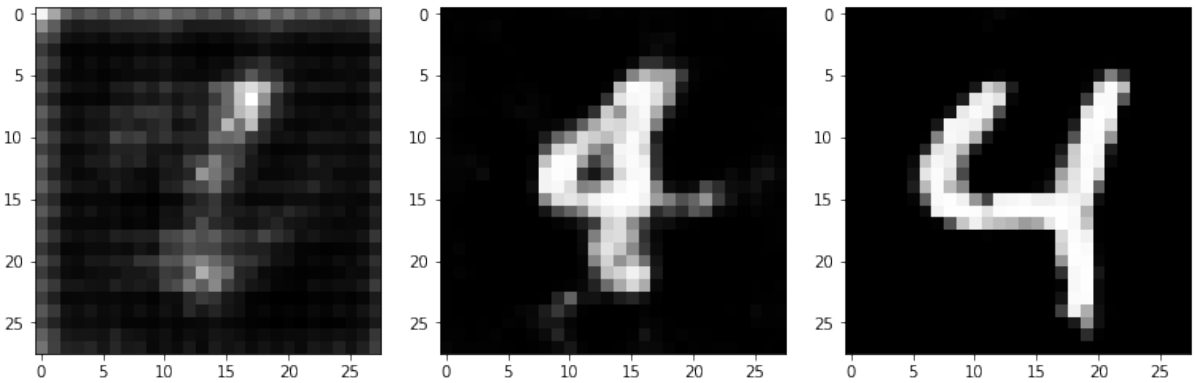


```
Epoch 1/4... Discriminator Loss: 0.9487... Generator Loss: 1.4490
Epoch 1/4... Discriminator Loss: 0.9468... Generator Loss: 1.1472
Epoch 1/4... Discriminator Loss: 0.9762... Generator Loss: 0.9686
```

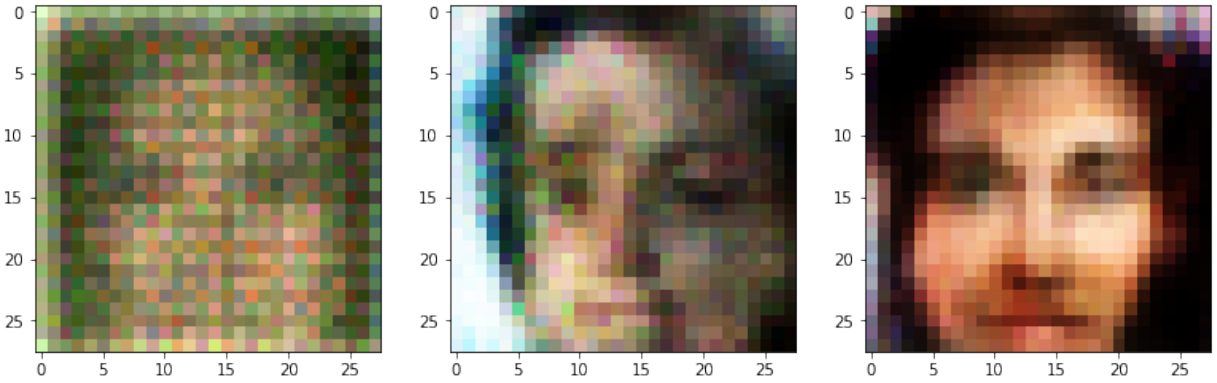
CelebA

We include this notebook in our Github. As a demo, we are going to show images generated by the GENERATOR at 3 selected moments during training

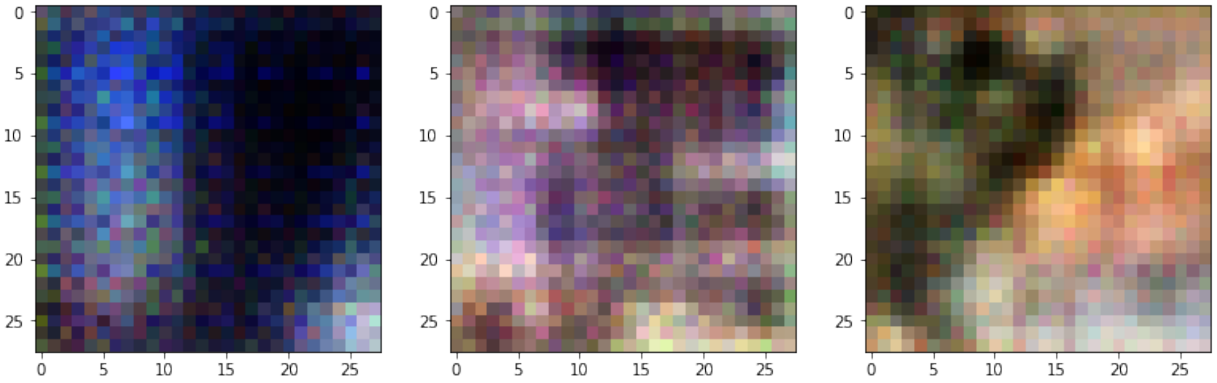
Results:
MNIST:



CELEB:



IMPRESSIONIST:



Experimental evaluation:

In the case of the MNIST dataset, only after a couple of epochs we could already see realistic digit images that could fool a human detector.

In the case of the Celeb dataset, the number of epochs required to see realistic faces was a lot higher compared to the MNIST. The results were good but the pixel detail of the generated image was not the best. Having said this, it was very interesting to see how the model started to improve and generate more realistic images.

Finally in the case of the Impressionist dataset the results were not as good as compared with the MNIST or Celeb. Because the original images had different structures and colors, the model did not settle into generating images that we could consider examples of impressionist paintings.

Conclusion:

In conclussion, GANs are a very interesting subject of study. The results are visual wich makes the process of learning very interesting. However, the process of training a GAN is exponentially more difficult compared with a regular Neural Network.

Because the process requires two models at the same time and because the Discriminator changes with time, it is very tricky to fine tune parameters or obtain substantial improvements.