E0:270 - Machine Learning - Understanding Generative Adversarial Networks

Arpan Mangal [1] **Rajarshi Banerjee** [2]

Abstract

Using neural networks for generating synthetic data is a difficult task. Generative Adversarial Networks(GANs) provide an interesting approach to tackle this problem by pitting two networks to compete against each other. However training a GAN is an extremely difficult task. In this project we are trying to get an insight into the working of GANs and apply it to some interesting problem.

1. Problem statement

Understanding the various aspects of GAN, to get more insight as to why training GANs are so difficult. The idea is to illustrate this with the help of some toy examples and then apply GANs on some real life problems.

2. Motivation

GANs have been hugely successful in generating high quality synthetic data, and it is one the top contenders in generating data that is indistinguishable from its real counterparts. Researchers believe that GANs can be improved to create entire movies, design new architecture, generate stories and many more things. Thus given all the success that GANs have produced we are interested in exploring its various variants to understand how it works from the inside out.

3. Literature review

We had our humble beginnings from the (Ian J. Goodfellow, 2014) paper to understand the theory and the architecture of vanilla GAN.

After that we quickly moved on one the most successful architecture in GANs known as the Deep Convolution GAN as described in the paper (Alec Radford & Luke Metz, 2016)

From there we turned our attention as to why training a GAN is difficult and found that the paper (Martin Arjovsky, 2017a) gave a nice theoretical argument as to why in practical settings training a GAN can be very difficult without the prior knowledge of optimal hyper-parameters

After that we looked at the state of the art GAN network, the Wasserstein GAN (Martin Arjovsky, 2017b) to understand how they deal with problem of GAN convergence with their novel approach of using the Earth Mover's distance loss function.

4. Model description

In our toy examples to train GANs on 1-D Gaussian we used a very simple Linear network for both the generator and the discriminator.

We also trained DCGANs whose arcitecture is given in the (Alec Radford & Luke Metz, 2016) paper.

5. Dataset description

Till now the datasets on which we have trained our GANs are: MNIST and LSUN outdoor church datasets.

6. Preliminary results

We experimented with a toy example of a 1-D Gaussian, where we tried to train the generator to have the output probability distribution converge to a predefined Gaussian distribution. We found that after a certain number of iterations it slowly but surely converged to the distribution.

Next we modified the previous experiment a little bit by modifying the given distribution to be the mixture of 3 Gaussian distribution with equal variances whose means were separated by 2 units, each with equal probability. This time however the Model was unable to converge to the distribution and the output distribution was centered around only one Gaussian.

Below we have given some of the observations we had in training GANs:

[1]Department of Computer Science and Automation, Indian Insitute of Science, Bangalore [2]Department of Computer Science and Automation, Indian Insitute of Science, Bangalore. Correspondence to: Arpan Mangal <arpanmangal@iisc.ac.in>, Rajarshi Banerjee <rajarshib@iisc.ac.in>.

Midterm Project Report, E0:270, Machine Learning, 2018.

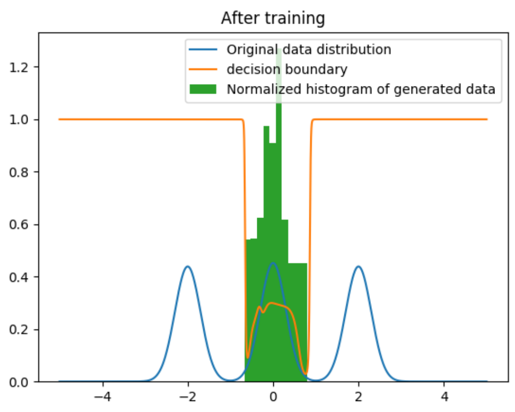

Figure 1. Effect on output distribution when discriminator is trained much more than generator, the generator collapses all outputs into a single Gaussian

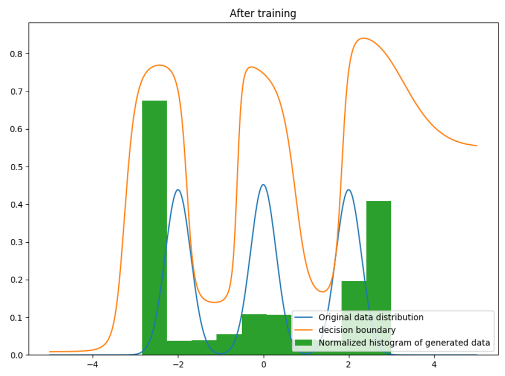

Figure 2. Effect on output distribution when generator is trained more than the discriminator, it was seen experimentally that the output distribution was very unstable and still collapsed outputs mostly to specific Gaussian means

7. Future Work

We next intent to implement the Wasserstein loss function in our GANs to see how the generated image quality changes, along with tuning some hyper parameters, we also intend to look into other practical applications that can be performed using GANs.

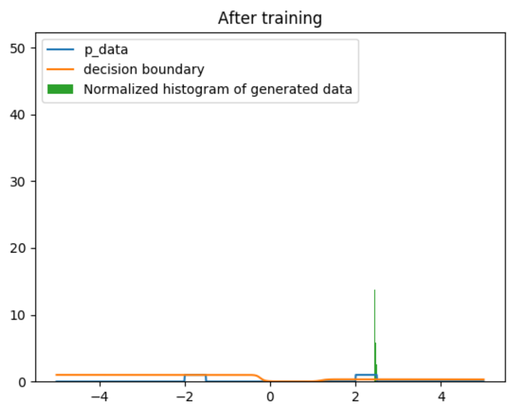

Figure 3. Effect on output distribution on training it on a mixture of two uniform distribution where generator is trained much more than the discriminator, here the output distribution collapses to a very small range. This scenario is known as mode collapse or helvetica

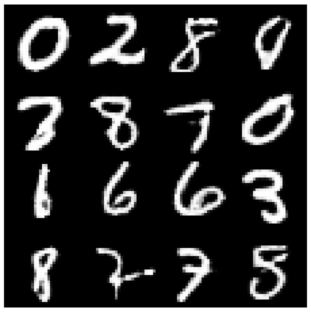

Figure 4. DCGAN output with infoGAN architecture trained on MNIST dataset after 25 epochs

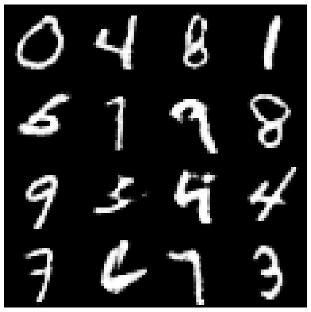

Figure 5. DCGAN output with infoGAN architecture (modified hyper-parameters) after 25 epochs

References

Alec Radford & Luke Metz, Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio. Generative adversarial nets, 2014.

Martin Arjovsky, Léon Bottou. Towards principled methods for training generative adversarial networks, 2017a.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein gan, 2017b.