2019

Challenge 2

GAN STANFORD DOG DATASET

MAX DE GOEDE

Abstract

In my plan, I describe that I want to learn about a new technique or become proficient in a technique that I learned during the minor. These being:

- Becoming proficient and gain more in-depth knowledge on CNN networks.
- Learn about GAN (General Adversarial Network), which I gained interest in during my previous challenge (Not become an expert on the algorithm but learn how it works and able to apply it)

I decided to work with GANS. This became a success, I learned how GANS worked and managed to make a network that was able to teach itself how to generate images from dogs by playing a tug of war with its counterpart which has the task to indicate whether the generated images were real or fake (discriminator).

I found a Kaggle competition that used GANS to generate dogs. With my goal set on learning about the algorithm I quickly looked through the competition notebooks to find what I could learn. Here I found a set of notebooks which were a beginner's course and introduction to anyone wanting to learn about GAN. I learned about both the generator, discriminator and what their place is within GAN. Any further work I looked into was to improve the output images (the generator results) or optimize the notebook runtime.

All of the notebooks were run in the KAGGLE cloud.

https://www.kaggle.com/pontiacboy/gan-testing-out

Introduction

This is my Machine Learning challenge 2.

For this, I worked with the Stanford dog dataset, which consists of labeled dog images (dog location labeled).

- Images/Annotation (Dog Location with xyMax and xyMin)
- 120 Categories (Breed)
- 20.580 Total images

With this dataset, a GAN network is trained. GAN trains by having two networks *"play"* against each other. Where one tries to figure out whether the images it receives are real or fake while the other creates these fake images.

During the plan of my challenge I indicated that I wanted to either learn about a new algorithm I had not yet worked with or become proficient on one I had learned during the minor. I decided to learn something new. During the first challenge, I learned how deepfake detection is trained by GANS. I decided to deepen my knowledge on this topic (In methods I describe more on this topic).

Contents

Methods

For the first challenge I decided to work on deepfake detection, here I learned that deepfake detection algorithms were trained. The detection (discriminator) received data input which could be videos that are fake or real and decided whether or not the input received was real. This was the first time I heard of GANs. Once the detection algorithm becomes better by backpropagation, the deepfake is forced to become better and adapt.

The example above is how I first came to hear of GANs. This is when I decided to dive deeper into GANs for my second challenge. Looking through Kaggle there were quite some examples of data sources which I would have liked to use:

1. **Generative Dog images**
    a. Data, 20.579 images with annotations.
        i. https://www.kaggle.com/c/generative-dog-images/overview
2. Dog Breed Identification
    a. Data, 120 Breeds about 150 images per breed. 20.580 in total
        i. https://www.kaggle.com/c/dog-breed-identification/overview
3. Ultrasound Nerve Segmentation
    a. Data,
        i. https://www.kaggle.com/c/ultrasound-nerve-segmentation/data
4. Google Landscape recognition
    a. Data,
        i. https://www.kaggle.com/c/landmark-recognition-2019/leaderboard

Not all the competitions shown above made use of GANs. But all made use of some kind of Object Detection/recognition.

GAN

GAN was designed to have 2 neural networks work against each other to constantly teach and improve one another. The original idea was for the usage of unsupervised learning. But the algorithm has shown great results with semi-supervised, supervised learning and reinforcement learning.

Few examples of applications:

- Fake videos,
    - Used in the creation of fake videos, improvement of deepfakes/detection.
- Fashion,
    - Used for fashion advertising no need to hire a whole crew with a model and create images.
- Games,
    - Creating higher resolution images by upscaling low resolution images and turning them into 4k.

Working of GAN[1]

Gan works with a generator that has the task to generate candidates (in our case images) while the discriminator evaluates and has to decide whether the candidate received is real or fake.

Generative network: learns to map from a latent space to a data distribution of interest.

Discriminative network: distinguishes candidates produced by the generator from the true data distribution.

The goal of the generative network is to fool the discriminator into thinking the images it creates are real.

The discriminator must have an initial dataset to train (learn) what a correct candidate (image) looks like. In our case, the discriminator is a CNN network tasked with learning patterns in the image to recognize dogs. While the generator learns by the success/failures of fooling the discriminative network. Backpropagation is used in both networks so both networks become better for each iteration (epoch).

In figure 1, one can see how the GAN works inside.

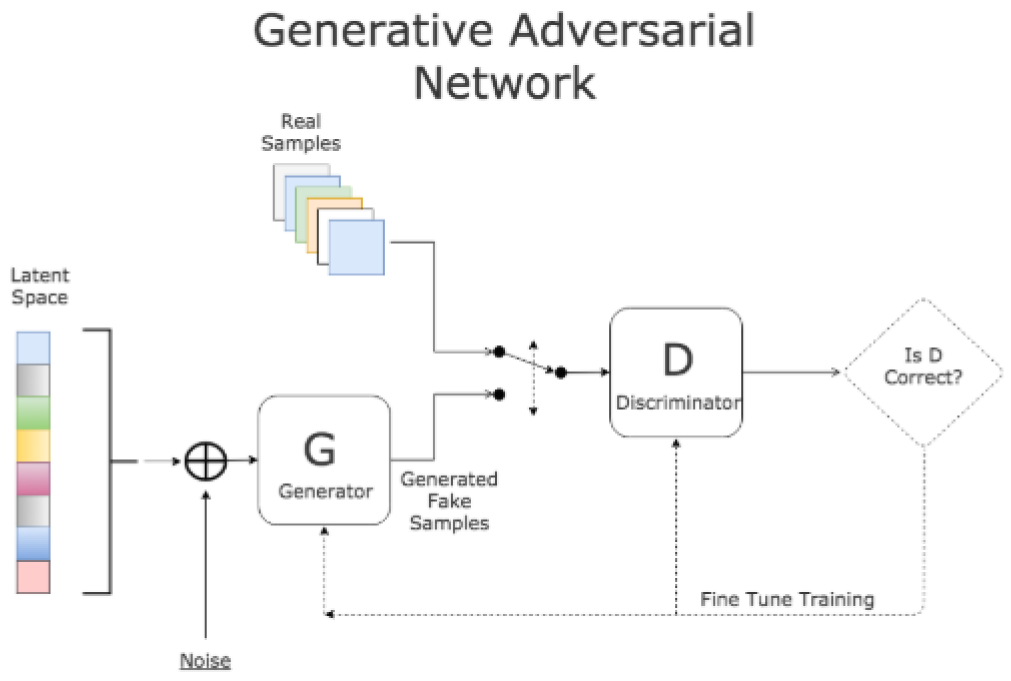

Figure 1, GAN WORKING

[1] Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio (2014) https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Results

First Phase

During the first phase of the project, my goal was to learn about GAN and how it works. I started by going through notebooks that others had made and tried to find something that could teach me the ABC's of GAN. I was able to find the following notebook, which is an introduction to GAN within the same competition. https://www.kaggle.com/jesucristo/gan-introduction

In this notebook I learn the following:

- Images
    - Image;
    - Annotation
- Discriminator
    - What it is (CNN network)
    - How it is made
    - How to use it
- Generator
    - What it is (CNN network)
    - How it is made
    - How to use it
- GAN interaction
    - How both networks interact with each other

The first phase only ran a total of 15 EPOCHS. The first epoch was much different than the rest, the first 100 iterations the loss is all over the place it makes it seem as if the discriminator is losing every time. But that is not the only thing, the generator is also performing badly. As the loss is around 3. The more iterations happen within the first epoch the result slowly starts getting better and better.

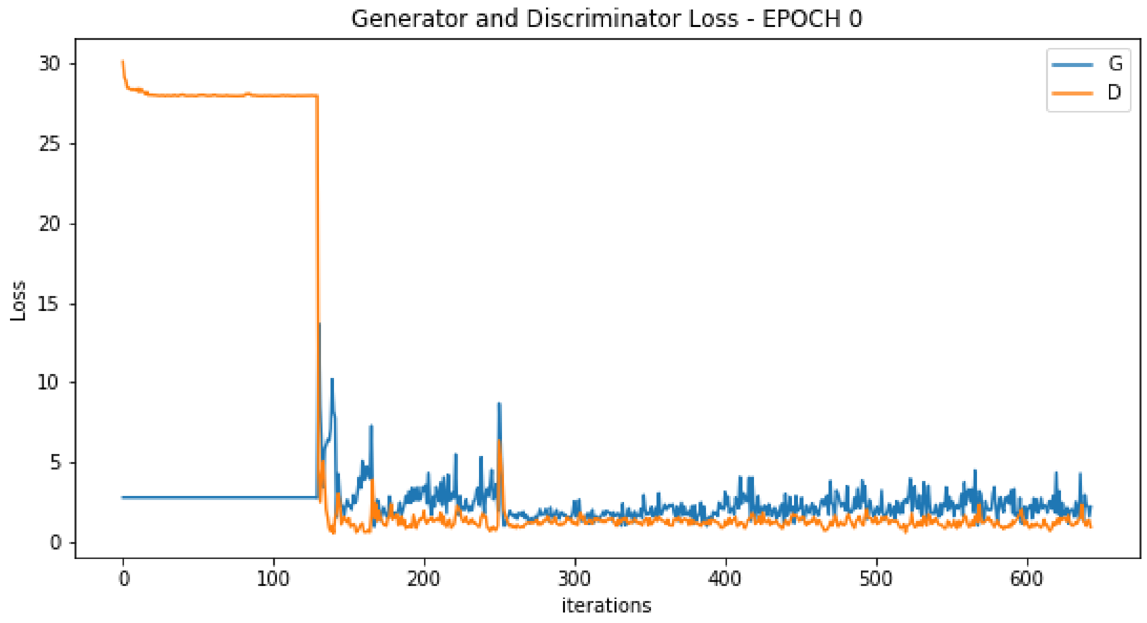

Figure 2 First Epoch loss results

Looking at the result from the first epoch (0), we see that the images look random lines across the canvas.

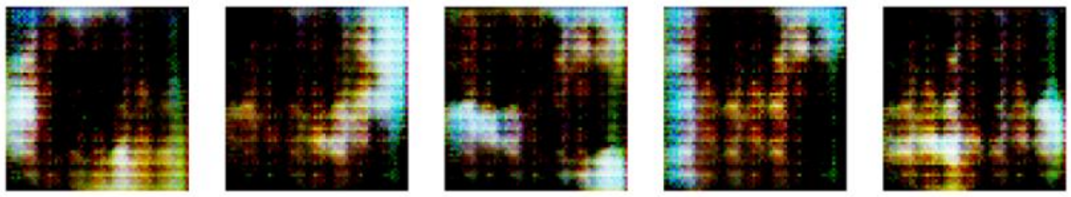

Figure 3 First Epoch generated images

10 epochs later we see that the start is much better, but the loss results are all over the place. It seems that the discriminator is performing decently. The generator is not doing too bad, but the loss is still on the higher side. Only seldom dropping under the loss of 1 (which would seem like the minimum requirement).

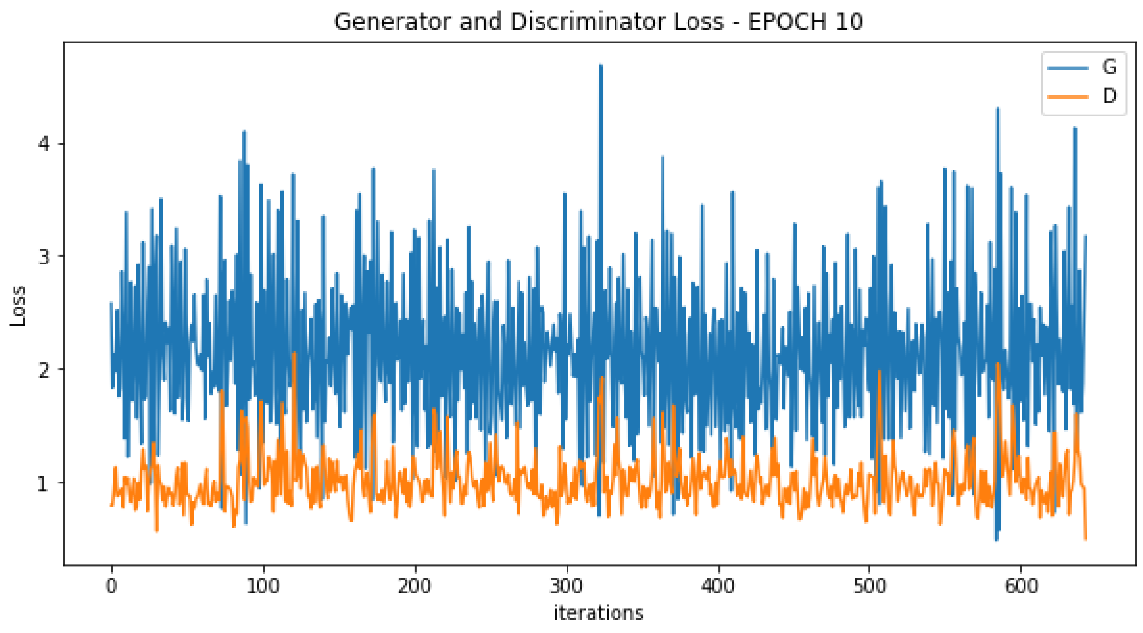

Figure 4 10th Epoch loss results

Looking at the results from epoch 10 we see images that look much more like dogs than the first batch. But this is far from we need/expect, as a first test run this feels like we can make something. I felt that if we could optimize the GAN better results are to come. By this point, it takes about a minute or two for an epoch to run. So it is hard to get good results.

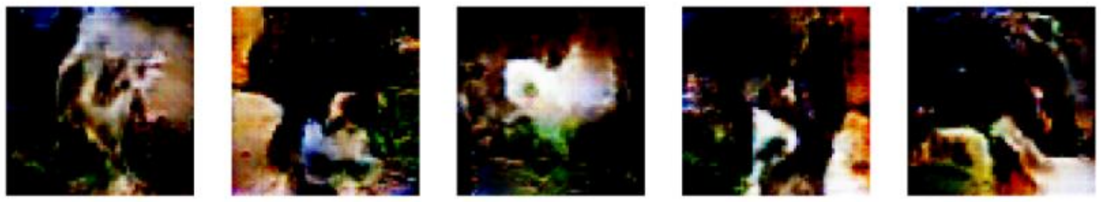

Figure 5 10th Epoch generated images

Second Phase

By now I already knew how GAN worked and wanted to improve the network more. The goals of this phase were to optimize the generator/discriminator and run more epoch and see if the results improve. I looked for more notebooks on the dog GAN competition, in the introduction notebook a reference was made to another notebook which acquired better results (using different generator/discriminator). I followed the steps on this notebook and got the following. https://www.kaggle.com/speedwagon/ralsgan-dogs.

Here we see the results of the loss, the loss here is much better. We can see that both losses are near the 1. The difference between the Discriminator and generator is smaller.

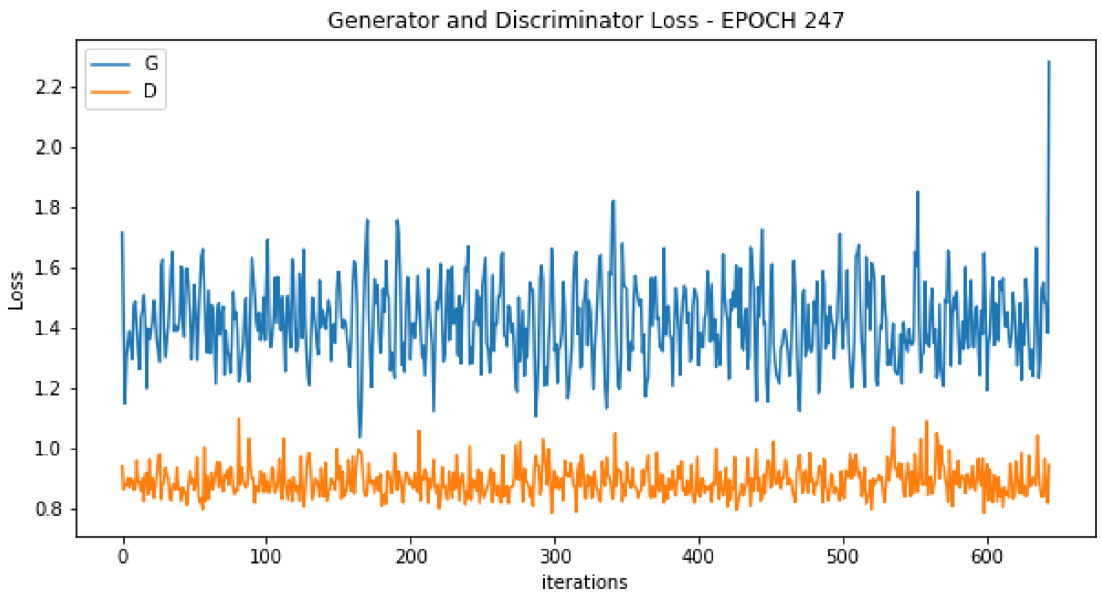

Figure 6 Second phase, Epoch 247 loss result

The results are very dark but looking much better than the previous images. Finally some images start looking like they could actually look like dogs.

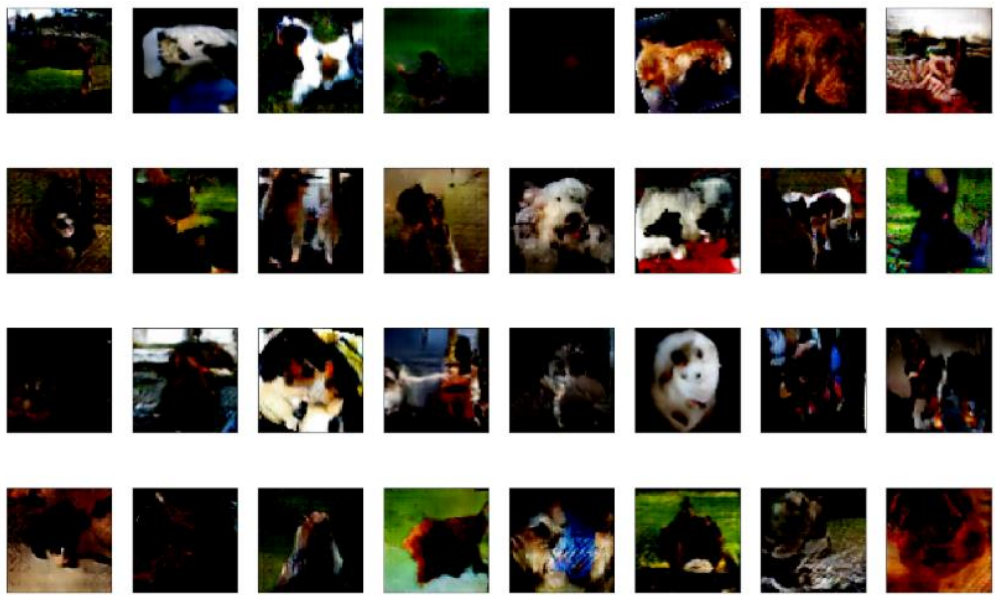

Figure 7 Second phase, Epoch 247 generated images

Third Phase

Now I knew how to work with GAN and the results have started to look like some creatures (not dogs, but shapes can be seen). By now there were a few more things that I wanted to run. I would like to have the average of each epoch result and show a graph towards the end. Have one data loader so data doesn't need to be loaded every epoch and it should be quicker to run each epoch. Further addition: random horizontal flip to some real loaded images.

The total results can be seen in the figure below. Here, the GAN performance reaches its peak around epoch 20. And that it slowly stabilizes once it reaches 60 epochs and stays in a constant line.

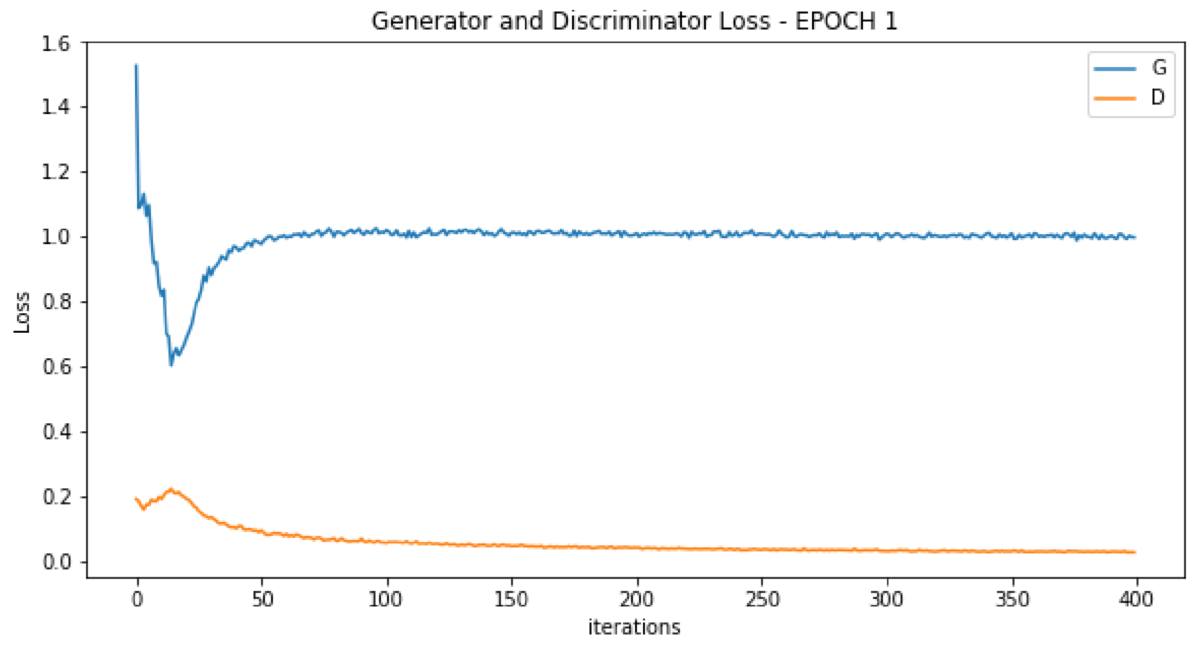

Figure 8 Third Phase, Average loss, per Epoch

The results are better there is one picture that somewhat looks like a dog. If I would get this image I would say that it looks like a dog. The images are still too dark, but I have no why this is the case.

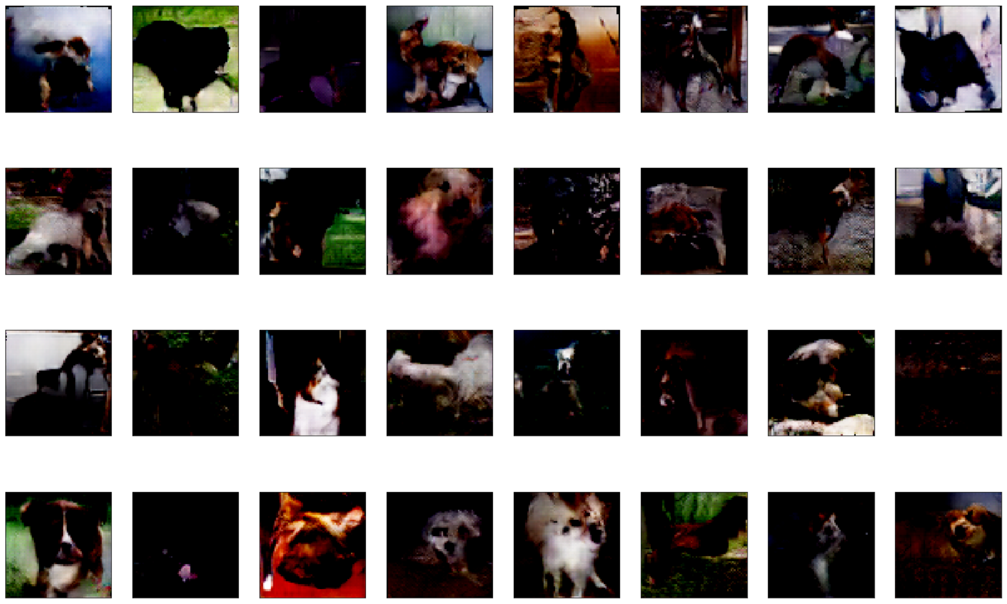

Figure 9 Third Epoch, generated images

Discussion

At the start of the project my goal was to learn about a new algorithm and I managed to achieve this. Looking at what I know now and have done through the challenge. I see this as a success I feel like I know quite a bit from this topic and I know how to work with it.