Generative Adversarial Networks (GANs)

Akshan Sameullah

April 2020

1 Introduction

Imagine two people are locked in a world-class art museum alone together. Out of boredom, Person A decides to learn to paint with some supplies from the gift shop while Person B (whose artistic talent happens to be very limited) just walks around admiring the artwork. After some time, Person A becomes very good at painting, while Person B becomes very familiar with the subtlest details in the paintings created by many world-class artists. After meeting up, Person B decides to use their gained knowledge to help Person A paint a world-class painting. Person A first paints a picture based on what they guess could be world-class artwork. They then hand it to Person B, who takes another look at the paintings in the museum and then provides feedback and suggestions to Person A to make their painting look world-class (such as using brighter colors, less opacity, etc:-). This is similar to how **Generative Adversarial Networks (GAN)** work.

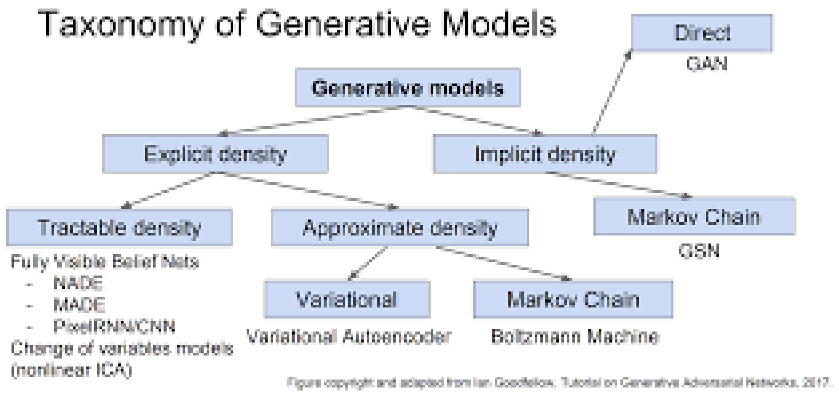
In the context of Generative Adversarial Networks, Person A in the analogy represents the **Generator**, while Person B is analogous to what is known as the **Discriminator**. The paintings that Person B studied represent training data, and the materials Person A used to gain art skills are analogous to **latent factors**, alternately referenced as the z vector.

2 Generative Models

In a broader scope, Generative Adversarial Networks are part of a class known as **Generative models** used for unsupervised learning. Generative models are models with the task of creating samples of a similar distribution as the training data.

Generative Models have a vast number of applications including generating artwork, simulation and planning for Reinforcement Learning, increasing image resolution, colorization, data augmentation, and countless more.

Generative Models are divided into two main groups, *Explicit Density Models* and *Implicit Density Models*. In short, Explicit Density Models specify the parameters of the distribution of a random variable for a function, while Implicit



$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Figure 1: Minimax objective function

Density Models follow a stochastic approach. To simplify, Explicit Density models pick a random variable and then compare that random variable's value in the data provided (e.g. plotting the degree that some images are green) and then tries to learn a function to estimate this nature. Implicit models, aim to generate samples to mimic the given ”real” data (training data).

3 Concept

Generative Adversarial Networks are currently the state-of-the-art sample generator. Generative Adversarial Networks are Implicit Density Models and do not use Explicit Density Functions, unlike many other Generative Models. Instead, they use an adversarial or game-theoretic approach.

GANs are made up of two networks, the **Generator network** and the **Discriminator Network**. The Generator Network produces fake images with the goal of having them look as realistic as the training data. The Discriminator Network is then fed these fake images from the Generator Network and predicts if the image is real or fake with respect to the training data.

GANs are trained in a game between the Discriminator and the Generator, in which the Generator tries to produce quality data that seems so realistic, the Discriminator cannot determine whether the data was created by the generator or not.

The goal of the Discriminator in this game, is to pick up on any mistakes or artifacts that the Generator made that differentiate what is real, and what is generated. This joint game is called the **minimax game**. The objective function for the minimax game is as follows:

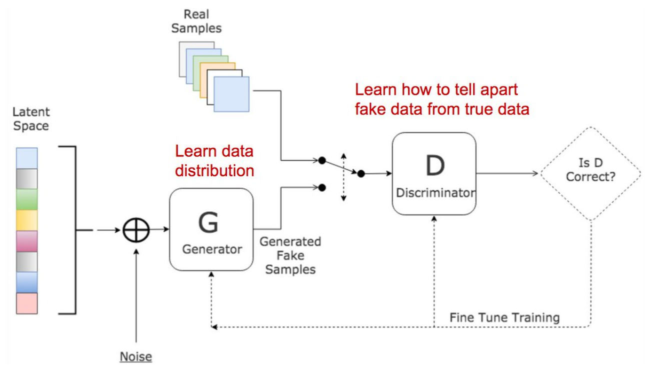


Figure 2: Joint process between Discriminator and Generator

In short, we find the minimum parameters for the Generator for the maximum parameters for the Discriminator of the confidence of a real image being real (the first term in brackets) and a fake image being fake (the second term in brackets). In the Minimax game, both networks ”take turns”, as in the Discriminator doesn’t update parameters while the Generator is training, neither does the Generator update parameters from the objective function while the Discriminator is training.

4 The Discriminator

In a GAN, the Discriminator is fed **Real Data** (Positive Training Examples received from Training Data) and **Generated Fake Data** (Negative Training Examples produced by Generator). The Discriminator classifies the Generator output it receives and then calculate **both** the loss of the Generator and the loss of the Discriminator. It then penalizes itself **only using its own calculated loss**. It then concludes its turn with back-propagation and updating its weights.

The **Discriminator** performs **Gradient Ascent**, or in other words, tries to increase the objective function so that *D(x)* (the Discriminator’s prediction probability of a sample from the training data is real) is high and *D(G(z))* (the Discriminator’s prediction probability of a sample of generated data is real) is low.

5 The Generator

The Generator takes in **random sampled noise from the target distribution** as the input (the z vector), and then outputs a sample using its weights to feed into the Discriminator. After receiving a classification of the output from the Discriminator (does it look real or fake), we calculate loss and back-propagate through **both** the Discriminator and the Generator but only update the Generator weights. That concludes the Generator’s turn in the game.

The **Generator** as per the minimax function above, performs **Gradient Descent** or in other words, it tries to decrease the objective function so that *1-D(G(z))* (the Discriminator's prediction probability of a sample of generated data is real) is low. With this objective, the gradient would generally be lowered in an exponential fashion. However, in practice, it is found that it helps to instead perform **Gradient Ascent** on *D(G(z))* so instead of trying to simply avoid the Discriminator being correct, it tries to make the Discriminator predict a false positive. This is essentially the same idea, but now the discriminator will learn more from the first few samples which are generally bad samples because of the initialization of the Generator being random. An update to the original GAN paper was added to incorporate this principle.

6 Putting it Together

Using the steps we saw above, we now have a general idea of making a GAN from scratch. Below is a pseudo-code (Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014) to reinforce understanding:

```
for number of iterations:
    for k steps: #it is currently unclear whether k must
                 # be 1
        sample minibatch of m noise samples
                        (z vector) from prior noise
        sample minibatch of m examples (x) from data
                        generating distribution
        Update Discriminator by ascending its stochastic
                        Gradient

    sample minibatch of m noise samples (z)
                        from prior noise
    Update Generator by ascending its stochastic
                        Gradient
```

7 The *z* vector

It is important to keep in mind that GANs are a form of unsupervised learning, and so they take in unlabeled data. We talked about how the Generator takes in random noise (**The *z* vector**) from the training data. To illustrate why this works, consider the following scenario.

A family keeps a pet dog in their house. Ever since the dog was born, it has been watching movies and television shows. It notices that when the dogs on the screen in movies do certain things, it makes the owner laugh. The dog then tries to combine some random actions from the dogs in movies and notices how much attention it receives from the owner. It then tries to combine some other actions from the figures in the movie to see if that increases attention.

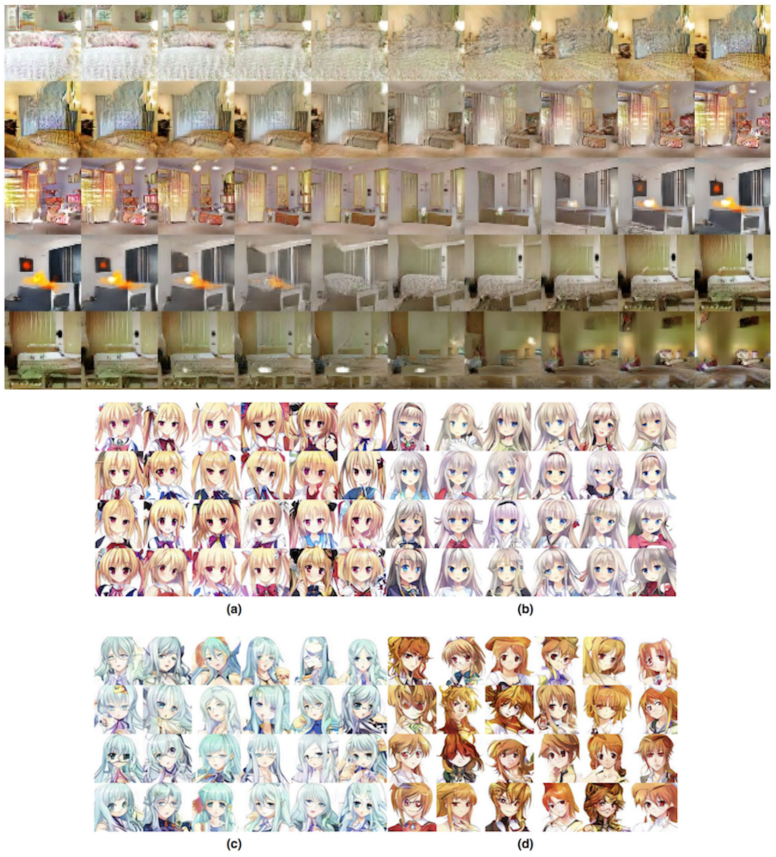


Figure 3: We can connect between random points in the latent space to form images with varying degrees of underlying variables

After many tries, the dog would eventually be able to generate actions that are just as attention-seeking as the actions of dogs in movies. In the analogy, the dog was not given any labeled data, but was still able to generate data that was indistinguishable from the television. This ties to the larger definition of Artificial Intelligence, which is to mimic the human brain.

8 CycleGANs, Deep Convolution GANs, and other recent advances

In this lecture, we talked about the plain vanilla GAN, which can produce generate new data given unlabeled data:

However, plain vanilla GANs generally aren't realistic enough to fool a human or are so close to the training example they aren't exactly new.

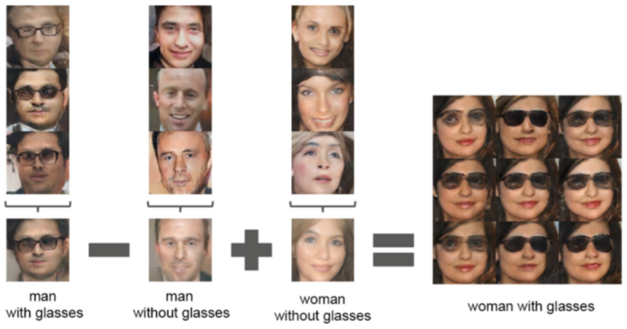


Figure 4: These Vector representations shown can be used in a sort of 'arithmetic' to produce new images

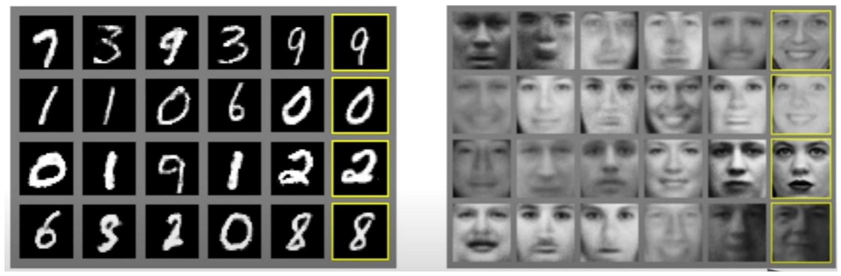


Figure 5: These images were generated by a GAN, except for the right column, which is the nearest neighbor from the training set

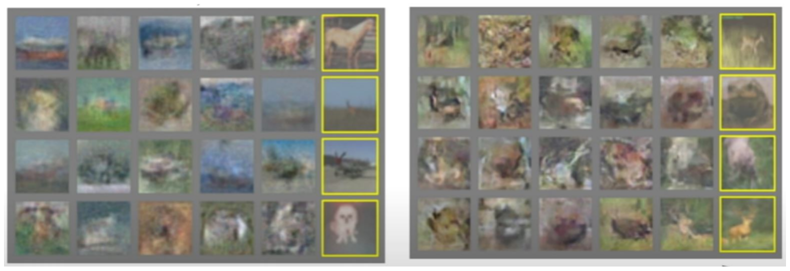


Figure 6: These images were generated by a basic GAN using the CIFAR-10 Dataset

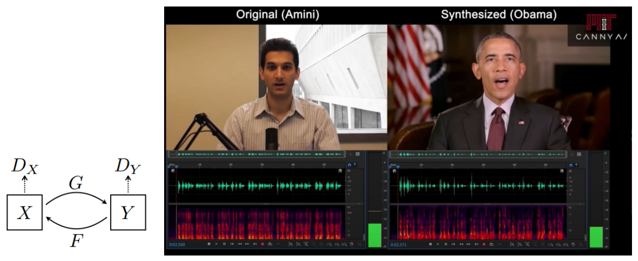


Figure 7: For a course introduction, MIT students used a CycleGAN to transform their introduction's image and voice to match that of Obama's. Notice the CycleGAN changes the background as well to match the training data

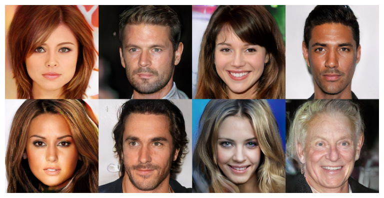


Figure 8: These faces were generated by a Deep Convolution GAN and appear photorealistic

As a result, many developments have been done by adding on to this idea. This includes the **CycleGAN**, which can perform transformations between domains. This became popular in the media, through applications such as Snapchat filters and deepfakes. Essentially, instead of generating new samples given training data, it takes an input and applies the distribution from a target to that input.

Another improvement (most commonly used) is the **Deep Convolution GAN**(Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", 2016). This idea uses a Convolutional Architecture, for both the Discriminator and the Generator (The generator uses fractionally strided convolutions, which makes sense based on what we learned about Generators) instead of a simple neural network. The results of this, are spectacularly realistic.

These GANs also follow a few guidelines (Radford et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", 2016) which include:

- Replace Pooling Layers with strided convolutions (fractional for the Generator)
- Use batch normalization in Discriminator and Generator
- Remove Fully Connected Hidden Layers for deeper architectures

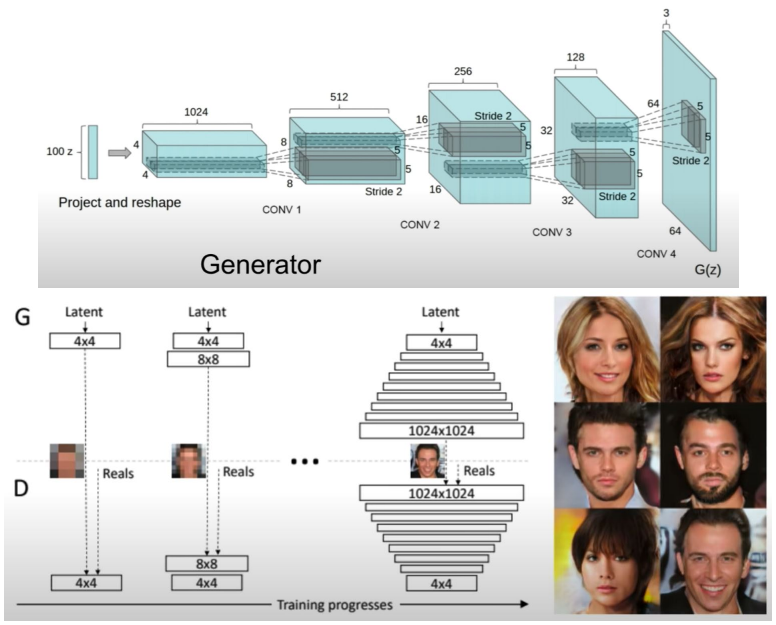


Figure 9: Progressive growing of Convolutional GANs (NVIDIA)

- Use ReLU for all layers in generator, except for the output, which uses Tanh
- Use LeakyReLU for all Discriminator Layers

Work on GANs accelerated during the past few years with many advancements such as the two mentioned. Advancements have been made on training, overall generation, speed, etc:-. Many applications have risen that use GANs all around. Examples used in the real world are incorporated in most industries. Examples include Using GANs to model clothing, or generate high-quality images from text or crude drawings for the need of stock images.

9 Credits and Resources

For more types of GANs and research papers visit https://github.com/hindupuravinash/the-gan-zoo

Stanford University School of Engineering Lecture 13 CS231n Generative Models

MIT 6.S191 Deep Generative Modeling

Carnegie Mellon University Deep Learning Lectures 23-24

Ian Goodfellow's Lectures, Google Brain ICCV17

IBM IT Infrastructure Generative Adversarial Networks

Apple Machine Learning Journal: GAN

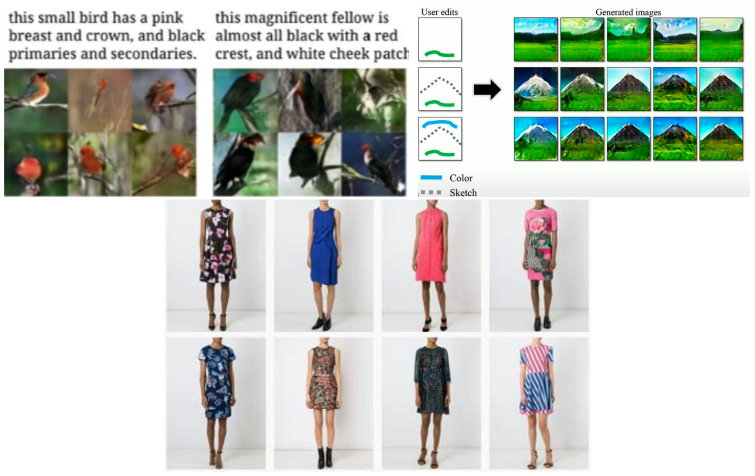

this small bird has a pink breast and crown, and black primaries and secondaries.
this magnificent fellow is almost all black with a red crest, and white cheek patch
User edits
Generated images
Color
Sketch

Figure 10: Some of the many applications of GANs. Text Synthesis (Akata et al. 2017)

Tensorflow Core
National Research University Higher School of Economics