Generative Adversarial Nets

The main idea is to develop a generative model via an adversarial process. We will discuss what is an adversarial process later. GAN consists of two model. The one is generative model G and the other is Dicrimninative model D. The purpose of a generative model is to generate the closest data as possible for give some input. The purpose of a discriminative model between two classes 0 and 1. 0 meaning the class belongs to Generative output and 1 meaning the class belongs to the true input sample from the original data.

This architecture corresponds to the minmax two-player game. One tries to create conflict over the other. Such networks are called adversarial networks. In the process of creating conflicts, both of them learn to be better and stronger than each other. When the discriminator makes an output of value ½ or 0.5, it implies that the discriminator is not able to disctinguish whether the value came from the generator output or the original sample.

Here, the G and D are defined by the multilayer perceptrons such that the entire system can be trained with backpropagation. The training of the discriminator and generator are done separately.

(image)

According to the paper, the generative model can be thought of as analogous to a team of couterfeiters who are trying to produce a fake currency and use them without getting caught.

While, the discriminative model can be thought of as analogous to the Police who are trying to detect the fake currency. Here, both the teams try to improve their methods until the currencies are indistinguishable from the original currency.

Adversarial Networks

Straight from the paper,

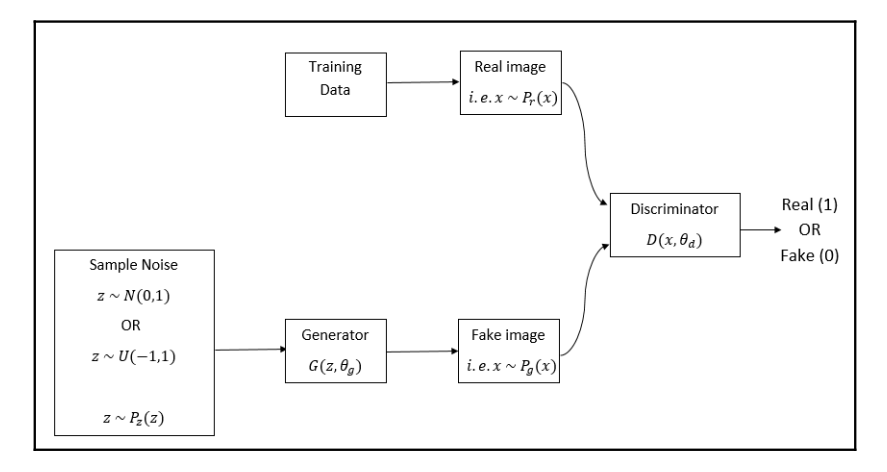
To learn the generator’s distribution Pg over data x, we define a prior on input noise variables Pz(z), then represent a mapping to data space as G(z; θg ).

where G is a differentiable function represented by a multilayer perceptron with parameters θ g .

We also define a second multilayer perceptron D(x; θd ) that outputs a single scalar.

Where D(x) represents the probability that x came from the data rather than Pg.

The architecture of GAN can be explained from the following figure.



Getting more into the Loss Function:

* z is the representation of the noise which is fed to the Input
* Pz represents the uniform or normal distribution from which the noise z is sampled
* x represents the inputs
* Pr is the representation of the real data from training set
* Pg is the representation of the fake data from the Generator

Here,

x~Pr(x) means x is sampled from real distribution Pr.

x~Pg(x) means x is sampled from generator disctribution Pg.

z~Pz(z) means z is sample from unifrom distribution Pz.

We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 − D(G(z))).

Hence, the D and G play the following two player min-max game with value function V(G,D):



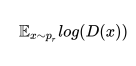
Understanding the value function

Discriminator LOSS

The loss function for the discriminator is:



Here, looking at the first term,



As already discussed, x~Pr(x) means we are sampling x from the real distribution. D(x) implying we are feeding input x to the discriminator. So, the discriminator will result the probability of the input being a real image. Eventually, since we have to maximize it, it is written as:



Again, looking at the second term,



As it is very clear that, the Generator outputs the fake image. When D(G(x)) is done, it means that the descriminator is fed the fake input generated by the generator. So, the discriminator will return the probability that the fake image being real image. But, since 1-D(G(x) is done, the discriminator returns the probability that the fake image being fake. So, eventuall we maximize it as:



Now, Combining both the equations, we obtain the discriminator value function as:



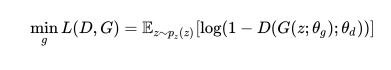
Generator Loss:

Since, we described the discriminator loss function as discriminator wants to minimize the probability of fake image being classified as real. And the equation is expressed as:

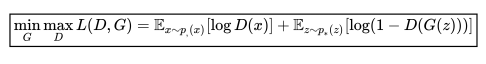


 →

But, the Generator’s primary objective is to fool the discriminator. The generator wants the fake image to be classied as real. So, the Generator minimizes the above probability as:



Now, summing up the loss from the Encoder and Decoder, the total loss is obtained as:



Why GANs ?

GANs provide an advancement in domain-specific data augmentation and solutions to problems that require generative solution such as image-to-image translations. They have ability to model high-dimensional data, handle missing data and the capacity to provide multi-modal outputs or multi-plausible answers.

Some most astounishing examples of GANS are:

1. New Faces generation on: <https://www.thispersondoesnotexist.com/>

2. Super Resolution Images

3. Image-to-Image Translation

4. Pose Guided Person Image Generation

5. Anime Characters Generations

6. Image Colorings, etc

The GANs have tons of applications each of which are able to amaze the whole world with their applications.

Facebook AI Directior, Scientist Yann LeChun says, “Generative Adversarial Networks is the most interesting idea in the last ten years in Machine Learning.”

GANs or VAEs:

GANs and VAEs are both generative model. They are similar in many aspects and also have a lot of differences from one other.

VAEs

* Easier to train, easier to implemetn and robust to hyperparameter choices
* The likelihood is tractable
* The network built is inference netwok. So, its reconstruction can be done.

GANs

* Much higher visual fidelity in generated samples
* Better with practical results