Generative Adversarial Networks
(GAN Families)

Ayush Raina

Indian Institute of Science

October 6, 2024

Generative Models

1. Generative Models usually learn the joint or conditional probability distribution of the data.
2. Restricted Boltzmann Machines learn $p(X, H)$, Variational Autoencoders learn $p(z|X)$ and Autoregressive Models learn $p(X)$ with the help of a neural network.

Now we are only interested in sampling from the true data distribution. We don't need $p(X)$.

Sampling

Given a image dataset $X = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \mathbb{R}^d$, we want to generate more images that look like they are from the same distribution as X.

What does GAN do?

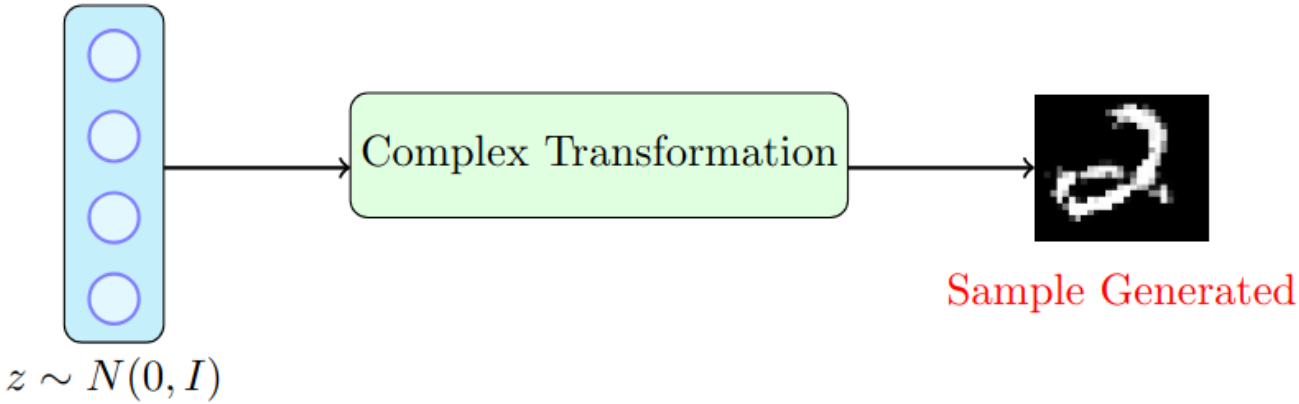

Figure 1: Real vs Fake Images

We will sample $z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})$ and pass it through a neural network (Generator), and train it in such a way that output images looks similar to training images.

Architecture

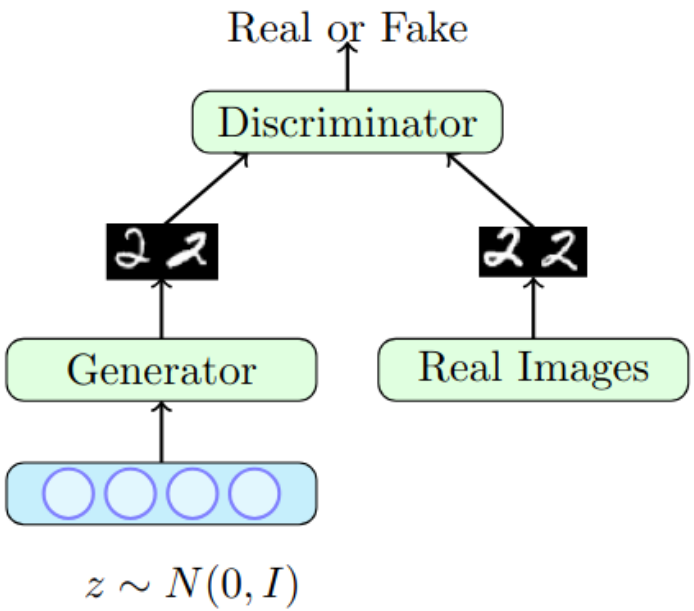

Figure 2: GAN Architecture

To achieve what we just saw, another neural network is introduced, called the Discriminator.

Architecture

GAN's architecture consists of two neural networks:

1. Generator $G(z; \phi)$
2. Discriminator $D(x; \theta)$

where ϕ and θ are the parameters of the generator and discriminator respectively.

Generator and Discriminator

The task of generator and discriminator is as follows:

1. Generator takes a random noise vector $z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})$ and generates a image $X = G(z; \phi)$.
2. Discriminator takes a image $X \sim p_{\text{data}}$ or $X = G(z; \phi)$ and outputs a probability $D(X; \theta)$ that X is from the real data distribution.

For convenience, we will denote $G(z; \phi)$ as $G_\phi(z)$ and $D(x; \theta)$ as $D_\theta(x)$.

Types of GAN's

There are many types of GAN's:

1. DCGAN: Deep Convolutional GAN
2. WGAN: Wasserstein GAN
3. CGAN: Conditional GAN
4. Pix2Pix: Image to Image Translation
5. CycleGAN: Cycle Consistent GAN

We will be discussing the some of those in results section.

Generator Objective Function

We know that higher score from the discriminator means that the image is more likely to be from the real data distribution.

Hence for some $z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})$, Generator objective can be written as:

$$\max_{\phi} \quad log(D_\theta(G_\phi(z))) \tag{1}$$

or we can also write it as:

$$\min_{\phi} \quad log(1 - D_\theta(G_\phi(z))) \tag{2}$$

Generator Objective Function

Hence we can write the generator objective function as:

$$\min_{\phi} \quad \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})} \left[log(1 - D_\theta(G_\phi(z))) \right] \tag{3}$$

or equivalently:

$$\max_{\phi} \quad \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})} \left[log(D_\theta(G_\phi(z))) \right] \tag{4}$$

Discriminator Objective Function

Discriminator has to give high scores to images $\sim p_{\text{data}}$ and low scores to images $\sim G_\phi(z)$. Hence the objective function for discriminator can be written as:

$$\max_\theta \quad \mathbb{E}_{x \sim p_{\text{data}}} \left[log(D_\theta(x)) \right] + \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})} \left[log(1 - D_\theta(G_\phi(z))) \right] \tag{5}$$

Overall Objective Function

The overall objective function for GAN can be written as:

$$\min_{\phi} \max_{\theta} \quad \mathbb{E}_{x \sim p_{\text{data}}} [log(D_\theta(x))] + \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})} [log(1 - D_\theta(G_\phi(z)))] \tag{6}$$

1. Discriminator tries to maximize 2nd term where as Generator tries to minimize it (Adversarial Training).
2. 1st term is independent of ϕ and hence can be ignored while training the generator.

Training GAN's

Training GAN's is done by alternating between the following steps:

1. Train Discriminator: Since we have to maximize w.r.t θ, we use Gradient Ascent on:

$$\mathbb{E}_{x\sim p_{\text{data}}}\left[log(D_\theta(x))\right]+\mathbb{E}_{z\sim\mathcal{N}(0,\mathbb{I}_{d\times d})}\left[log(1-D_\theta(G_\phi(z)))\right] \quad (7)$$

2. Train Generator: Since we have to minimize w.r.t ϕ, we use Gradient Descent on:

$$\mathbb{E}_{z\sim\mathcal{N}(0,\mathbb{I}_{d\times d})}\left[log(1-D_\theta(G_\phi(z)))\right] \quad (8)$$

Algorithm for GAN Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Generator Training

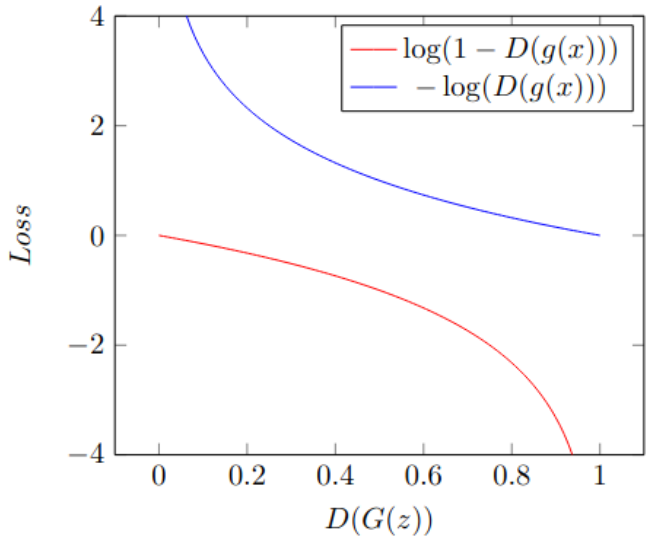

Figure 4: Generator Training

In practice $log(1 - D_\theta(G_\phi(z)))$ may not provide enough gradient for the generator to learn because it will be close to 0 initially.

New Training Objective For Generator

We can use the following objective function for training the generator:

$$\max_{\phi} \quad \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_{d \times d})} \left[log(D_\theta(G_\phi(z))) \right] \tag{9}$$

Now in this case Gradients will flow better and the generator will learn faster.

Generated Handwritten Digits

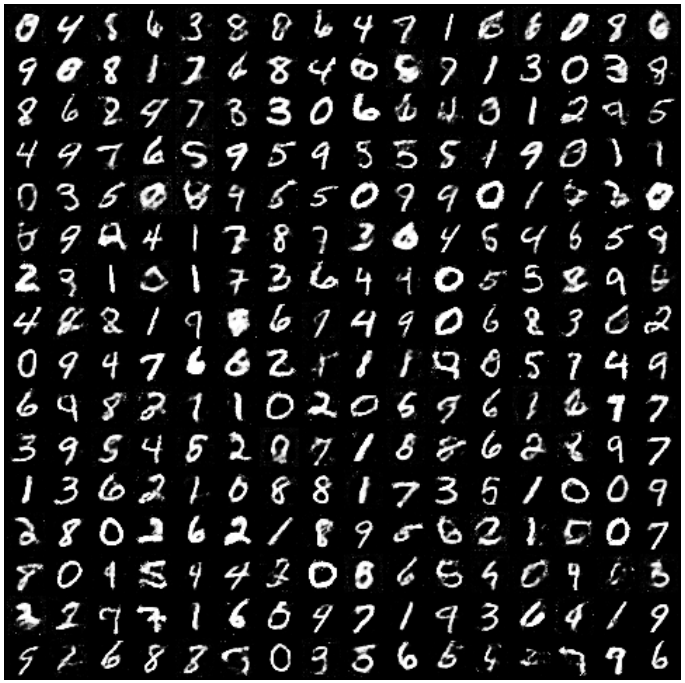

Figure 5: Generated Handwritten Digits

In this case both the generator and discriminator are feed forward neural networks.

Generated Face Images

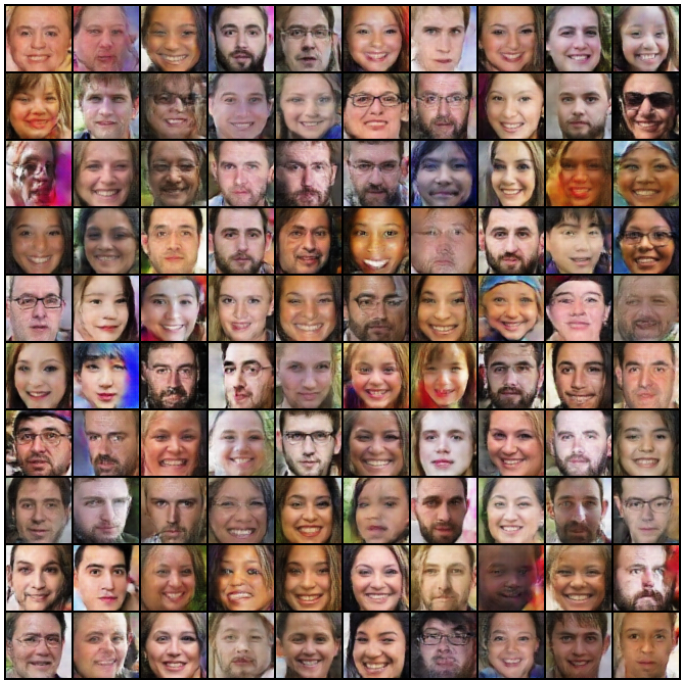

Figure 6: Generated Face Images

In this case both the generator and discriminator are Convolutional Neural Networks.

Generated Anime Characters

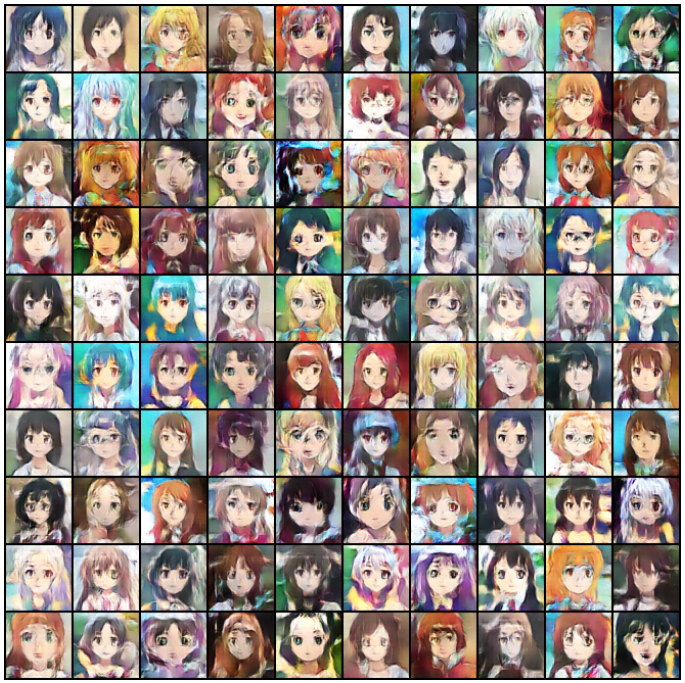

Figure 7: Generated Anime Characters

In this case both the generator and discriminator are Convolutional Neural Networks

Generated Animal Images

Figure 8: Generated Animal Images

In this case both the generator and discriminator are Convolutional Neural Networks.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio *Generative Adversarial Networks*. arXiv:1406.2661, 2014.

[2] Alec Radford, Luke Metz, Soumith Chintala *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv:1511.06434, 2015.

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen *Improved Techniques for Training GANs*. arXiv:1606.03498, 2016.

[4] Martin Arjovsky, Soumith Chintala, Léon Bottou *Wasserstein GAN*. arXiv:1701.07875, 2017.

Thank You!

Thank You!