Generative Adversarial Networks

Ravi Kothari, Ph.D.
ravi.kothari@ashoka.edu.in

"A wise enemy is better than a foolish friend" – Unknown

Generative Modeling

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled
- We saw an important use-case of generative models when we covered variational auto-encoder *viz.* learning compact representations

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled
- We saw an important use-case of generative models when we covered variational auto-encoder *viz.* learning compact representations
- Other reasons include,

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled
- We saw an important use-case of generative models when we covered variational auto-encoder *viz.* learning compact representations
- Other reasons include,
  - Generate realistic samples e.g. paintings, art – possibly this is one element of creativity

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled
- We saw an important use-case of generative models when we covered variational auto-encoder *viz.* learning compact representations
- Other reasons include,
    - Generate realistic samples e.g. paintings, art – possibly this is one element of creativity
    - We can tell which data is more likely – something that can be very useful in speech recognition

Generative Modeling

- Our goal in generative modeling is to learn the distribution from which the true data was sampled
- We saw an important use-case of generative models when we covered variational auto-encoder *viz.* learning compact representations
- Other reasons include,
  - Generate realistic samples e.g. paintings, art – possibly this is one element of creativity
  - We can tell which data is more likely – something that can be very useful in speech recognition
  - Of course, learning compact representations remains an attractive reason

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)

- GANs are an *implicit generative model* in the sense that a GAN produces samples that confirms to the distribution from which the original data came from. Explicit generative models on the other hand attempt to estimate the actual distribution

GAN Architecture

GAN Architecture

- Generate a sample z from a simple fixed distribution e.g. $\mathcal{N}(0, 1)$

GAN Architecture

- Generate a sample z from a simple fixed distribution e.g. $\mathcal{N}(0, 1)$
- There is a generator network which produces $x = G(z)$

GAN Architecture

- Generate a sample z from a simple fixed distribution e.g. $\mathcal{N}(0, 1)$
- There is a generator network which produces $x = G(z)$
- In an implicit generative model, the produced samples are indistinguishable from the true (given unlabeled) data

GAN Architecture

- Generate a sample z from a simple fixed distribution e.g. $\mathcal{N}(0, 1)$
- There is a generator network which produces $x = G(z)$
- In an implicit generative model, the produced samples are indistinguishable from the true (given unlabeled) data
- So, GANs also train a discriminator network that tries to tell if the data came from the training set (true data) or whether it was generated from the generator network

GAN Architecture

- Generate a sample z from a simple fixed distribution e.g. $\mathcal{N}(0, 1)$
- There is a generator network which produces $x = G(z)$
- In an implicit generative model, the produced samples are indistinguishable from the true (given unlabeled) data
- So, GANs also train a discriminator network that tries to tell if the data came from the training set (true data) or whether it was generated from the generator network
- So, the discriminator network produces $D(x)$ – the probability that x is real

GAN Training

GAN Training

- So, the generator tries to produce something that is realistic enough to fool the discriminator and the discriminator tries to catch the generator. We can thus define a loss function as,

$$J_D = E_{x \sim D}[-\log D(x)] + E_z[-\log(1 - D(G(z)))] \tag{1}$$

GAN Training

- So, the generator tries to produce something that is realistic enough to fool the discriminator and the discriminator tries to catch the generator. We can thus define a loss function as,

$$J_D = E_{x \sim D}[-\log D(x)] + E_z[-\log(1 - D(G(z)))] \qquad (1)$$

- When this cross-entropy loss is low, it implies the discriminator can easily tell the real from fake. If the cross-entropy is high, it means the discriminator is not able to tell them apart

GAN Training

- So, the generator tries to produce something that is realistic enough to fool the discriminator and the discriminator tries to catch the generator. We can thus define a loss function as,

$$J_D = E_{x \sim D}[-\log D(x)] + E_z[-\log(1 - D(G(z)))] \tag{1}$$

- When this cross-entropy loss is low, it implies the discriminator can easily tell the real from fake. If the cross-entropy is high, it means the discriminator is not able to tell them apart
- So, for the generator the obvious cost function is to *maximize* the discriminator's cross-entropy, i.e.,

$$J_G = -J_D = \text{const} + E_z[\log(1 - D(G(z)))] \tag{2}$$

Since the generator has no control over the first term, it is just a constant

GAN Training

GAN Training

- So, the generator is trying to compute,

$$\arg \max_G \min_D J_D \tag{3}$$

GAN Training

- So, the generator is trying to compute,

$$\arg \max_G \min_D J_D \tag{3}$$

- This is also referred to as the minimax formulation and is an example of a zero-sum game

GAN Training

- So, the generator is trying to compute,

$$\arg \max_G \min_D J_D \tag{3}$$

- This is also referred to as the minimax formulation and is an example of a zero-sum game
- The discriminator and the generator are trained jointly on their respective cost-functions using back-propagation

GAN Training

GAN Training

- There is a difficulty with cross-entropy in this scenario. For example, assume the discriminator is doing well. $D(G(z))$ is close to 0 in that case and J_G is also close to 0 even though the scenario implies the generator was doing poorly

GAN Training

- There is a difficulty with cross-entropy in this scenario. For example, assume the discriminator is doing well. $D(G(z))$ is close to 0 in that case and J_G is also close to 0 even though the scenario implies the generator was doing poorly
- We thus often use the modified cost function for the generator given by,

$$J_G = E_z[-\log D(G(z))] \tag{4}$$

GAN Examples

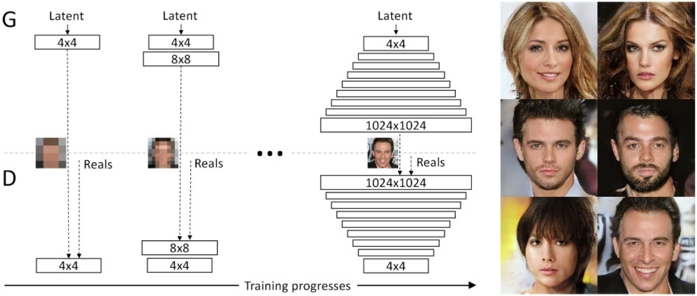

Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

From Karras et al., 2017

Demo

https://poloclub.github.io/ganlab/