GAN - Theory and Applications

Michele de Simoni
Paolo Galeone
Emanuele Ghelfi
Federico di Mattia

March 13, 2019

Generative Adversarial Networks

> "Adversarial Training (also called GAN for Generative Adversarial Networks) is the most interesting idea in the last 10 years of ML."
>
> — Yann LeCun

Generative Adversarial Networks

Two components, the **generator** and the **discriminator**:

- The **generator** G, aim is to capture the data distribution.
- The **discriminator** D, estimates the probability that a sample came from the training data rather than from G.

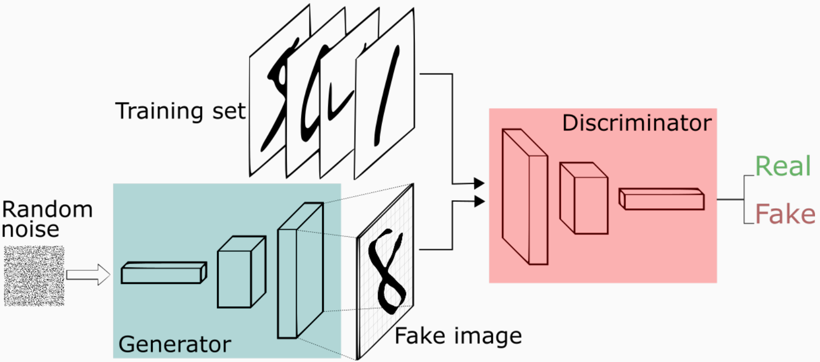


Figure 1: Credits: Reference

Generative Adversarial Networks

Generator and Discriminator compete against each other, playing the following **zero sum min-max game** with value function $V_{GAN}(D, G)$:

$$\min_G \max_D V_{GAN}(D, G) = \mathop{\mathbb{E}}_{x \sim p_{data}(x)}[\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

Generative Adversarial Networks

Generator and Discriminator compete against each other, playing the following **zero sum min-max game** with value function $V_{GAN}(D, G)$:

$$\min_G \max_D V_{GAN}(D, G) = \underbrace{\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)]}_{\text{real samples}} + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

Generative Adversarial Networks

Generator and Discriminator compete against each other, playing the following **zero sum min-max game** with value function $V_{GAN}(D, G)$:

$$\min_G \max_D V_{GAN}(D, G) = \underbrace{\mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x)]}_{\text{real samples}} + \underbrace{\mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]}_{\text{generated samples}} \tag{1}$$

GANs - Discriminator

Intuitive explanation:

- **Discriminator** needs to:
    - Correctly classify real data:

$$\boxed{\max_{D} \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x)].} \tag{2}$$

    - Correctly classify wrong data:

$$\boxed{\max_{D} \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))].} \tag{3}$$

GANs - Generator

Intuitive explanation:

- **Generator** needs to **fool** the discriminator:
  - Generate samples similar to the real one:

$$\boxed{\min_{G} \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))].} \tag{4}$$

GANs - Generator

Intuitive explanation:

- **Generator** needs to **fool** the discriminator:
  - Generate samples similar to the real one:

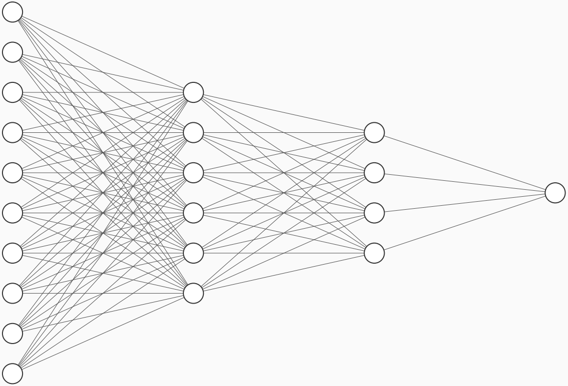
$$\boxed{\min_G \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))].} \tag{4}$$

  - Saturates easily (1).
  - Change loss for generator:

$$\boxed{\max_G \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(D(G(z)))].} \tag{5}$$

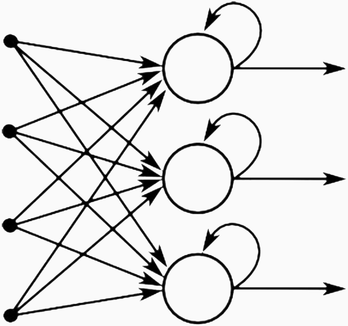
GANs - Models definition

- Both D and G can be parametrized functions (Neural Networks).
- Different architectures for different data types.
    - Tuple of numbers?



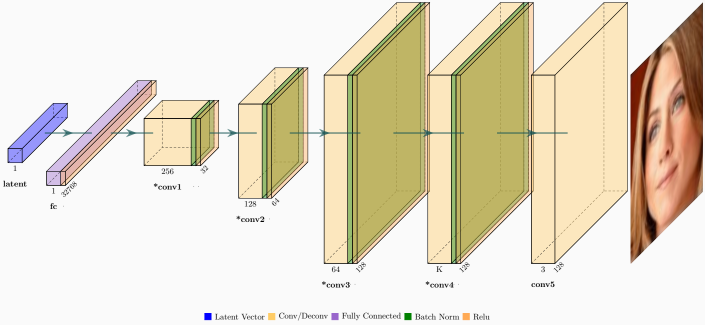
GANs - Models definition

- Both D and G can be parametrized functions (Neural Networks).
- Different architectures for different data types.
    - Text or sequences?



GANs - Models definition

- Both D and G can be parametrized functions (Neural Networks).
- Different architectures for different data types.
    - Images?



GAN Training

GANs - Training

- Discriminator and generator are **competing** against each other.
- How to train?
- **Alternating** execution of training steps.
- Use **minibatch stochastic gradient descent/ascent**.

GANs - Training - Discriminator

How to **train** the **discriminator**?
Repeat from 1 to **k**:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(z)$
2. Sample minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$ from data generating distribution $p_{data}(x)$
3. Train the **discriminator** by stochastic gradient **ascent**:

$$\Delta_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))$$

How to **train** the **discriminator**?

Repeat from 1 to **k**:

1. Sample minibatch of m noise samples $z^{(1)}, \dots, z^{(m)}$ from noise prior $p_g(z)$
2. Sample minibatch of m examples $x^{(1)}, \dots, x^{(m)}$ from data generating distribution $p_{data}(x)$
3. Train the **discriminator** by stochastic gradient **ascent**:

$$\Delta_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))}_{\text{Discriminator loss}}$$

GANs - Training - Discriminator

How to **train** the **discriminator**?
Repeat from 1 to **k**:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(z)$
2. Sample minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$ from data generating distribution $p_{data}(x)$
3. Train the **discriminator** by stochastic gradient **ascent**:

$$\underbrace{\Delta_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))}_{\text{Discriminator loss}}}_{\text{Loss estimation using m samples}}$$

GANs - Training - Generator

How to **train** the **generator**?

The update is executed **only once** and only after the turn of the discriminator is completed:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(z)$
2. Train the **generator** by stochastic gradient **ascent**:

$$\Delta_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D(G(z^{(i)})))$$

How to **train** the **generator**?
The update is executed **only once** and only after the turn of the discriminator is completed:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(z)$
2. Train the **generator** by stochastic gradient **ascent**:

$$\Delta_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\log(D(G(z^{(i)})))}_{\text{Generator loss}}$$

GANs - Training - Generator

How to **train** the **generator**?

The update is executed **only once** and only after the turn of the discriminator is completed:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from noise prior $p_g(z)$
2. Train the **generator** by stochastic gradient **ascent**:

$$\Delta_{\theta_g} \quad \underbrace{\frac{1}{m} \sum_{i=1}^{m} \underbrace{\log(D(G(z^{(i)})))}_{\text{Generator loss}}}_{\text{Loss estimation using m samples}}$$

GANs - Training - Considerations

- Optimizers: Adam, Momentum, RMSProp
- Training phase can last for an **arbitrary number** of steps or epochs
- Training is completed when the discriminator is **completely fooled** by the generator.
- The goal of GAN training is to reach a **Nash Equilibrium** where the best D can do is random guessing.

Type of GANs

Types of GANs

Two big families:

- **Unconditional** GANs (just described)
- **Conditional** GANs (2)

References

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

[2] Mehdi Mirza and Simon Osindero, *Conditional Generative Adversarial Nets*.