Overview

Introduction

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

- The **generator** G needs to capture the data distribution.
- The **discriminator** D estimates the probability that a sample comes from the training data rather than from G.

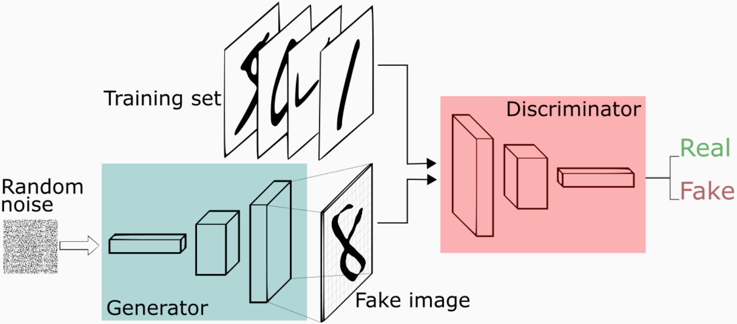


Figure 1: Credits: Silva

GANs game:

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

Generative Adversarial Networks

GANs game:

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

Generative Adversarial Networks

GANs game:

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

GANs - Discriminator

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

$$\boxed{\max_{D} \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x)]} \qquad D(x) \rightarrow 1$$

  - Correctly classify wrong data:

$$\boxed{\max_{D} \mathop{\mathbb{E}}_{z \sim p_{z}(z)} [\log(1 - D(G(z)))]} \qquad D(G(z)) \rightarrow 0$$

- The discriminator is an **adaptive loss function**.

GANs - Generator

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

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

GANs - Generator

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

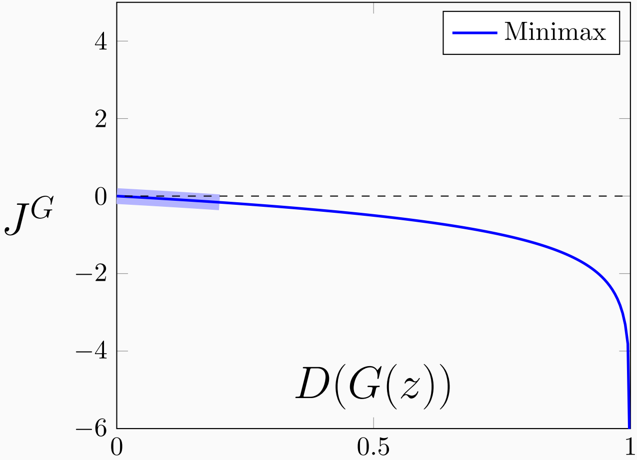
$$\boxed{\min_G \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]} \qquad D(G(z)) \to 1$$

  - Non saturating objective (Goodfellow et al., 2014):

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

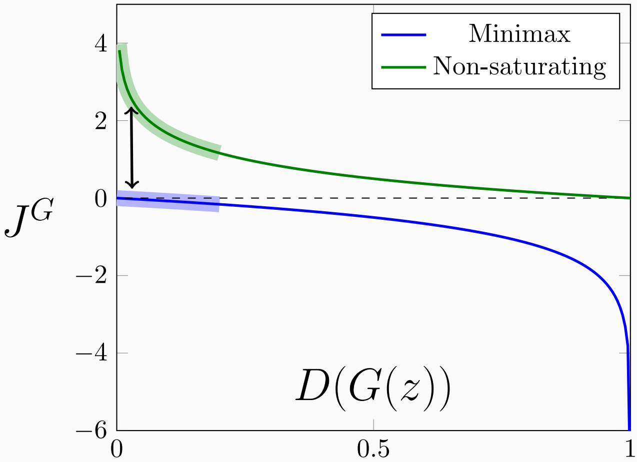
GANs - Generator Objectives

- Minimax: $\log(1 - D(G(z)))$



GANs - Generator Objectives

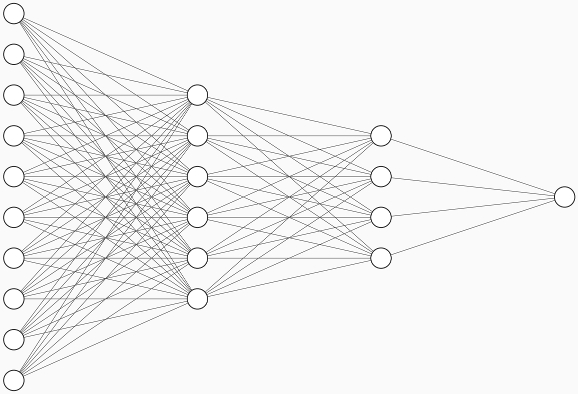
- Minimax: $\log(1 - D(G(z)))$
- Non-saturating: $-\log(D(G(z)))$



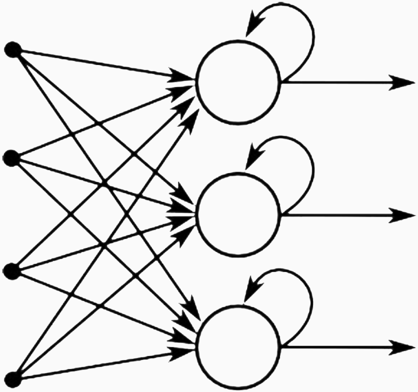
Models definition

GANs - Models definition

- Different architectures for different data types.
  - Tuple of numbers? Fully Connected Neural Networks

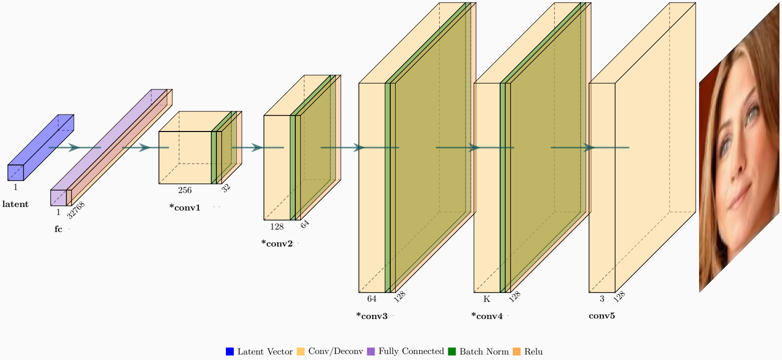


- Different architectures for different data types.
    - Text or sequences? Recurrent Neural Networks



GANs - Models definition

- Different architectures for different data types.
  - Images? Convolutional Neural Networks



GANs Training

GANs - Training

- D and G are **competing** against each other.
- **Alternating** execution of training steps.
- Use **minibatch stochastic gradient descent/ascent**.

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

Repeat from 1 to **k**:

1. Sample minibatch of m noise samples $z^{(1)}, \dots, z^{(m)}$ from $p_z(z)$

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 $p_z(z)$
2. Sample minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$ from $p_{data}(x)$

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 $p_z(z)$
2. Sample minibatch of m examples $x^{(1)}, \ldots, x^{(m)}$ from $p_{data}(x)$
3. Update **D**:

$$\mathbf{J} = \underbrace{\frac{1}{m}\sum_{i=1}^{m} \log \mathbf{D}(x^{(i)}) + \log(1 - \mathbf{D}(\mathbf{G}(z^{(i)})))}_{\text{D performance}}$$

$$\theta_d = \theta_d + \lambda \nabla_{\theta_d} \mathbf{J}$$

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

Update executed **only once** after **D** updates:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from $p_z(z)$

GANs - Training - Generator

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

Update executed **only once** after **D** updates:

1. Sample minibatch of m noise samples $z^{(1)}, \ldots, z^{(m)}$ from $p_z(z)$
2. Update **G**:

$$\mathbf{J} = \underbrace{\frac{1}{m} \sum_{i=1}^{m} \log(\mathbf{D}(\mathbf{G}(z^{(i)})))}_{\text{G performance}}$$

$$\theta_{\mathrm{g}} = \theta_{\mathrm{g}} + \lambda \nabla_{\theta_{\mathrm{g}}} \mathbf{J}$$

GANs - Training - Considerations

- Optimizers: Adam, Momentum, RMSProp.
- **Arbitrary number** of steps or epochs.
- Training is completed when D is **completely fooled** by G.
- Goal: reach a **Nash Equilibrium** where the best D can do is random guessing.

Types of GANs

Types of GANs

Two big families:

- **Unconditional** GANs (just described).
- **Conditional** GANs (Mirza and Osindero, 2014).

Conditional GANs

- **Both** G and D are **conditioned** on some extra information **y**.
- In **practice**: perform conditioning by feeding **y** into D and G.

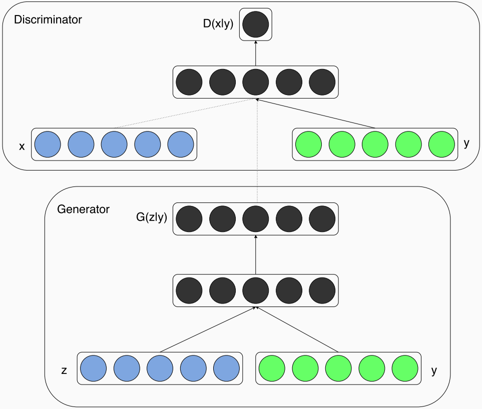


Figure 2: From Mirza and Osindero (2014)

Conditional GANs

The GANs game becomes:

$$\min_G \max_D \mathop{\mathbb{E}}_{x \sim p_{data}(x|\mathbf{y})}[\log D(x, \mathbf{y})] + \mathop{\mathbb{E}}_{z \sim p_z(z)}[\log(1 - D(G(z|\mathbf{y}), \mathbf{y}))]$$

Notice: the same representation of the condition has to be presented to both network.

Real-world GANs

- Semi-Supervised Learning (Salimans et al., 2016)
- Image Generation (almost all GAN papers)
- Image Captioning
- Anomalies Detection (Zenati et al., 2018)
- Program Synthesis (Ganin et al., 2018)
- Genomics and Proteomics (Killoran et al., 2017) (De Cao and Kipf, 2018)
- Personalized GANufactoring (Hwang et al., 2018)
- Planning