



GAN - Theory and Applications

Emanuele Ghelfi

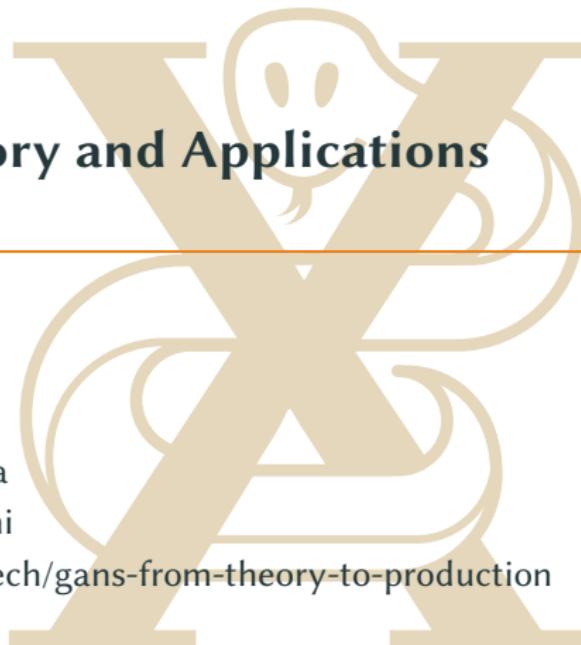
Paolo Galeone

Federico Di Mattia

Michele De Simoni

github.com/zurutech/gans-from-theory-to-production

April 30, 2019





Overview

1. Introduction
2. Models definition
3. GANs Training
4. Types of GANs
5. GANs Applications

Introduction

“Generative Adversarial Networks is the **most interesting idea in the last ten years in machine learning.**

Yann LeCun, Director, Facebook AI



Generative Adversarial Networks

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.

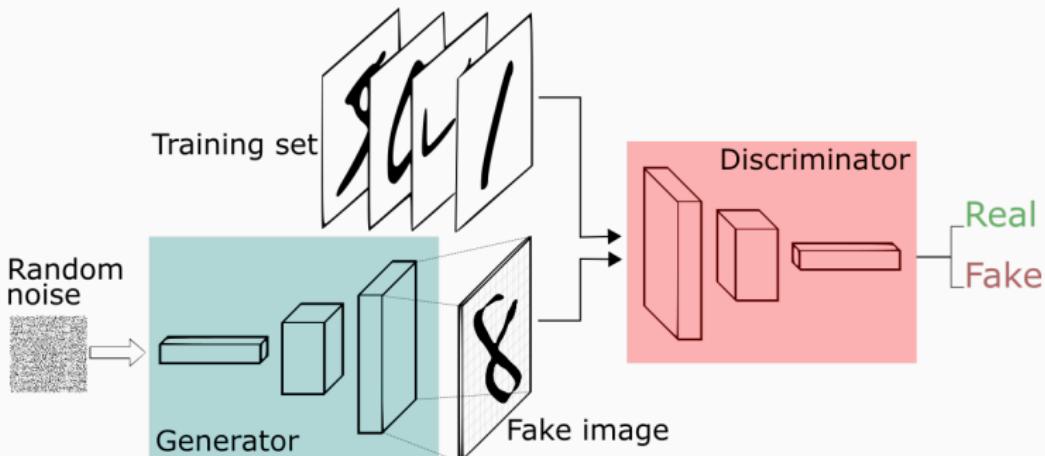


Figure 1: Credits: Silva

Generative Adversarial Networks

GANs game:

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

Generative Adversarial Networks

GANs game:

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

Generative Adversarial Networks

GANs game:

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

GANs - Discriminator

- **Discriminator** needs to:

- Correctly classify **real** data:

$$\max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)].$$

$$D(x) \rightarrow 1$$

- Correctly classify **wrong** data:

$$\max_D \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$

$$D(G(z)) \rightarrow 0$$

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



**YOU DON'T NEED TO
DESIGN A LOSS FUNCTION**

**IF A DISCRIMINATOR
DESIGNS ONE FOR YOU**

GANs - Generator

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

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

$$D(G(z)) \rightarrow 1$$

GANs - Generator

- **Generator** needs to **fool** the discriminator:

- Generate samples similar to the real one:

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

$$D(G(z)) \rightarrow 1$$

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

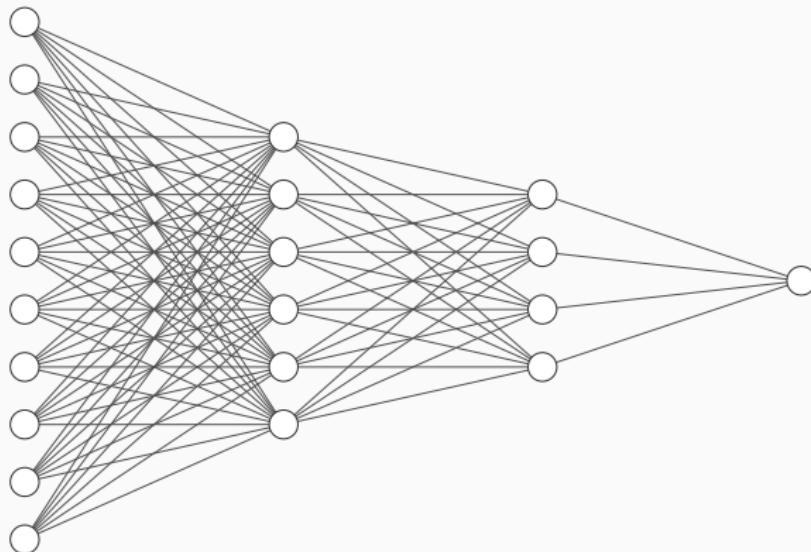
$$\max_G \mathbb{E}_{z \sim p_z(z)} [\log(D(G(z)))].$$

(2)

Models definition

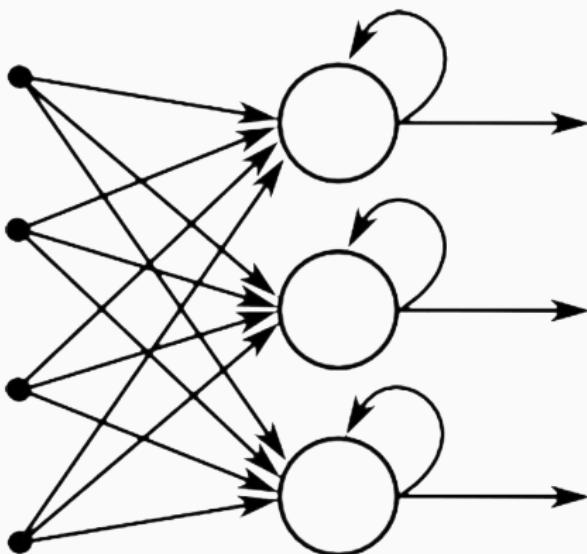
GANs - Models definition

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



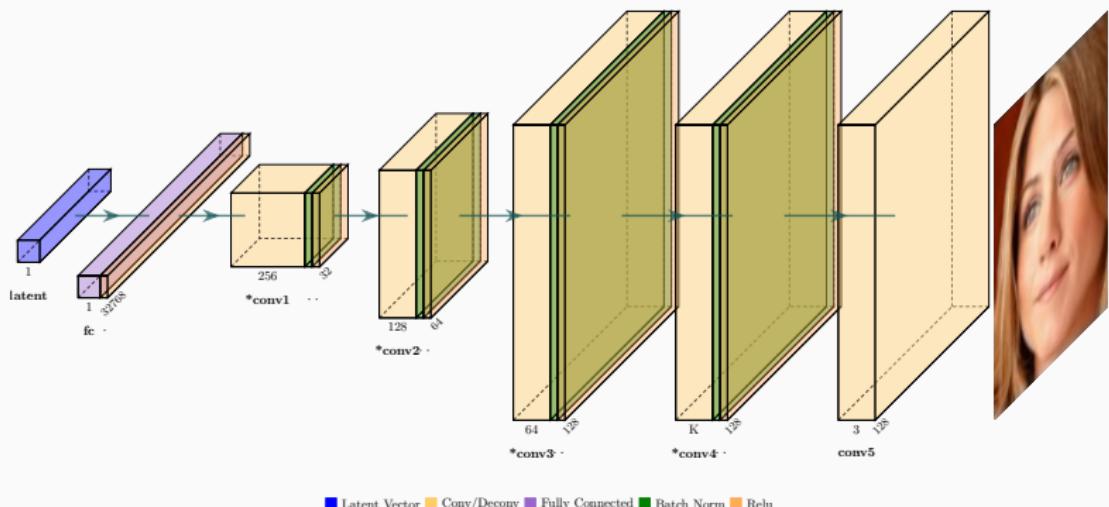
GANs - Models definition

- 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**.



GANs - Training - Discriminator

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

GANs - Training - Discriminator

How to **train the discriminator?**

Repeat from 1 to **k**:

1. Sample minibatch of m noise samples $z^{(1)}, \dots, z^{(m)}$ from $p_z(z)$
2. Sample minibatch of m examples $x^{(1)}, \dots, x^{(m)}$ from $p_{data}(x)$
3. Update the **discriminator**:

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

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

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 $z^{(1)}, \dots, z^{(m)}$ from $p_z(z)$

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 $z^{(1)}, \dots, z^{(m)}$ from $p_z(z)$
2. Update the **generator**:

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

$$\theta_g = \theta_g + \lambda \nabla_{\theta_g} J$$

GANs - Training - Considerations

- Optimizers: Adam, Momentum, RMSProp.
- **Arbitrary number** of steps or epochs.
- Training is completed when the discriminator is **completely fooled** by the generator.
- 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 the discriminator and generator.

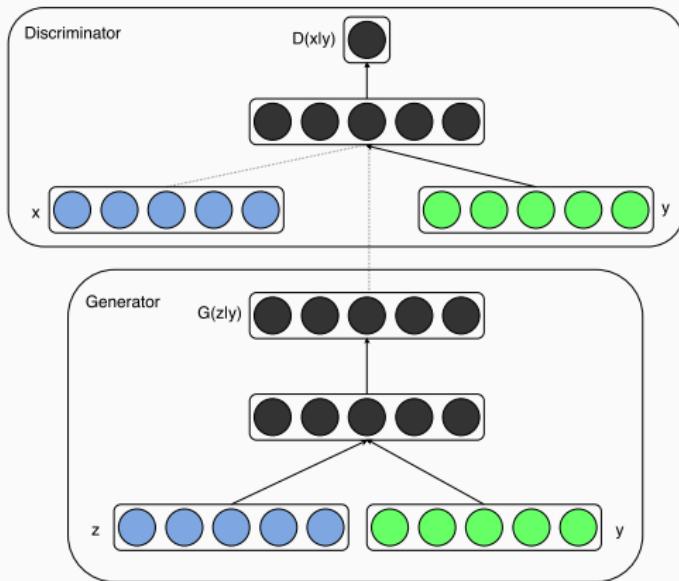


Figure 2: From Mirza and Osindero (2014)

Conditional GANs

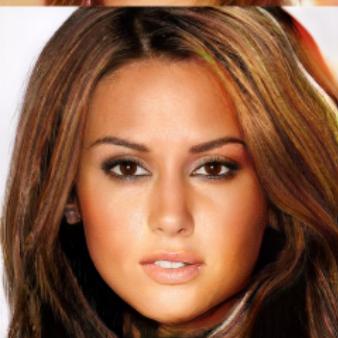
The GANs game becomes:

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

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

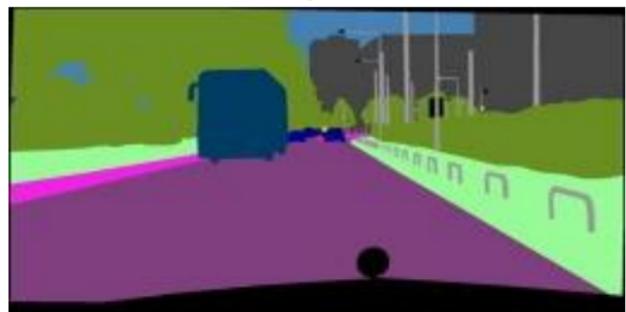
GANs Applications

Unconditional - Face Generation - Karras et al. (2017)



Conditional - Domain Translation - Isola et al. (2016)

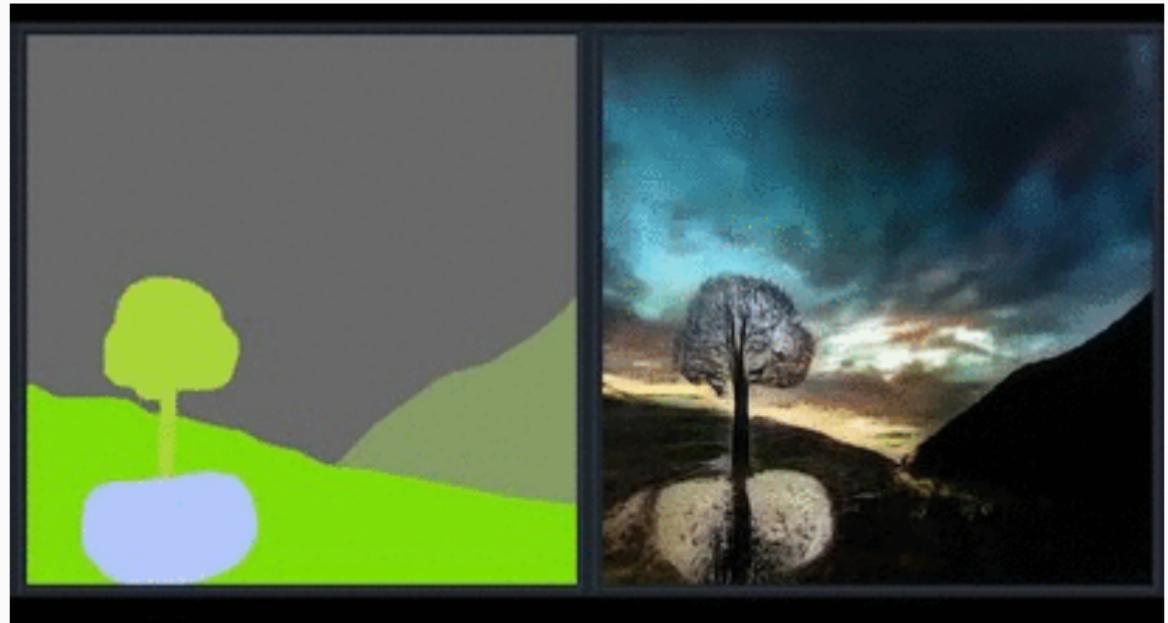
Input



Generated



Conditional - Semantic Image Synthesis - Park et al. (2018)



Conditional - Image Super Resolution - Ledig et al. (2016)



SRGAN



Thank you for your attention!
Questions?

Real-world GANs

- Semi-Supervised Learning (?)
- Image Generation (almost all GAN papers)
- Image Captioning
- Anomalies Detection (?)
- Program Synthesis (?)
- Genomics and Proteomics (?) (?)
- Personalized GANufactoring (?)
- Planning

References

- [Goodfellow et al. 2014] GOODFELLOW, Ian J. ; POUGET-ABADIE, Jean ; MIRZA, Mehdi ; Xu, Bing ; WARDE-FARLEY, David ; OZAIR, Sherjil ; COURVILLE, Aaron ; Bengio, Yoshua: Generative Adversarial Networks. (2014). – (2014)
- [Isola et al. 2016] ISOLA, Phillip ; ZHU, Jun-Yan ; ZHOU, Tinghui ; EFROS, Alexei A.: Image-to-Image Translation with Conditional Adversarial Networks. (2016). – (2016)
- [Karras et al. 2017] KARRAS, Tero ; AILA, Timo ; LAINE, Samuli ; LEHTINEN, Jaakko: Progressive Growing of GANs for Improved Quality, Stability, and Variation. (2017). – (2017)

[Ledig et al. 2016] LEDIG, Christian ; THEIS, Lucas ; HUSZAR, Ferenc ; CABALLERO, Jose ; CUNNINGHAM, Andrew ; ACOSTA, Alejandro ;AITKEN, Andrew ; TEJANI, Alykhan ; TOTZ, Johannes ; WANG, Zehan ; SHI, Wenzhe: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. (2016). – (2016)

[Mirza and Osindero 2014] MIRZA, Mehdi ; OSINDERO, Simon: Conditional Generative Adversarial Nets. (2014). – (2014)

[Park et al. 2018] PARK, Taesung ; LIU, Ming-Yu ; WANG, Ting-Chun ; ZHU, Jun-Yan: Semantic Image Synthesis with Spatially-Adaptive Normalization. (2018). – (2018)

[Silva] SILVA, Thalles: An Intuitive Introduction to Generative Adversarial Networks (GANs)