Deep Learning Lesson 4

Lecturer: Paolo Russo
Dip. Ingegneria Informatica, Automatica e Gestionale, Roma

Credits: PYCONX GAN Theory and Applications slides, by Ghelfi, Galeone, Di Mattia, De Simoni

SAPIENZA
UNIVERSITÀ DI ROMA

Overview

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

“*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.

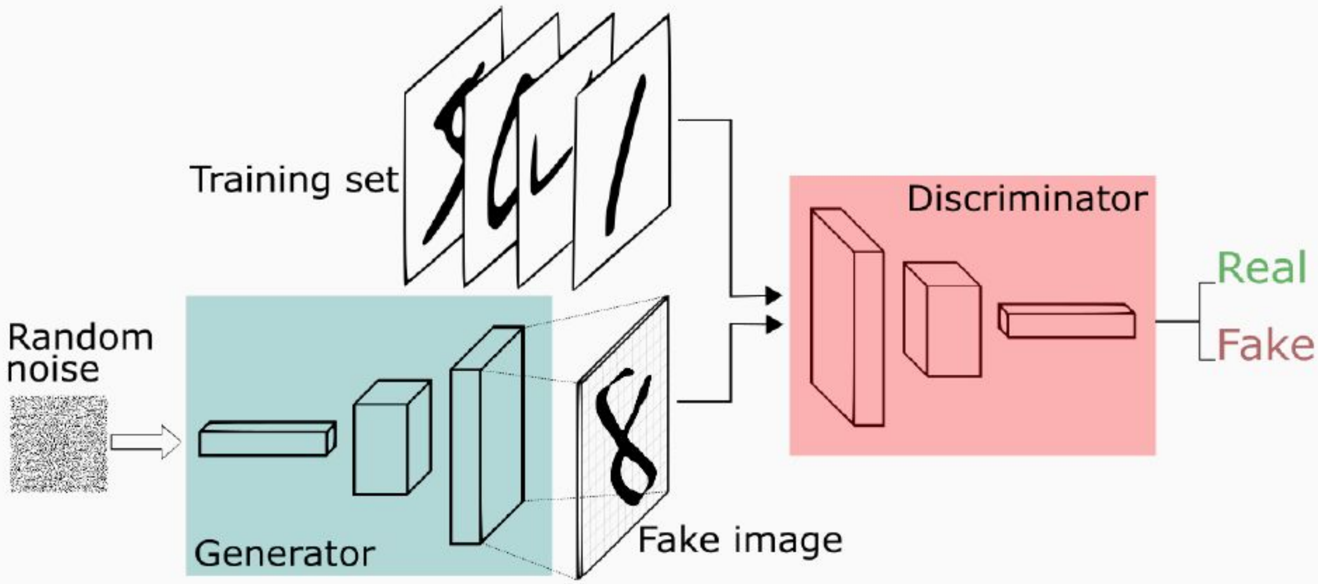


Figure 1: Credits: Silva

Generative Adversarial Networks

GANs game:

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

GANs - Discriminator

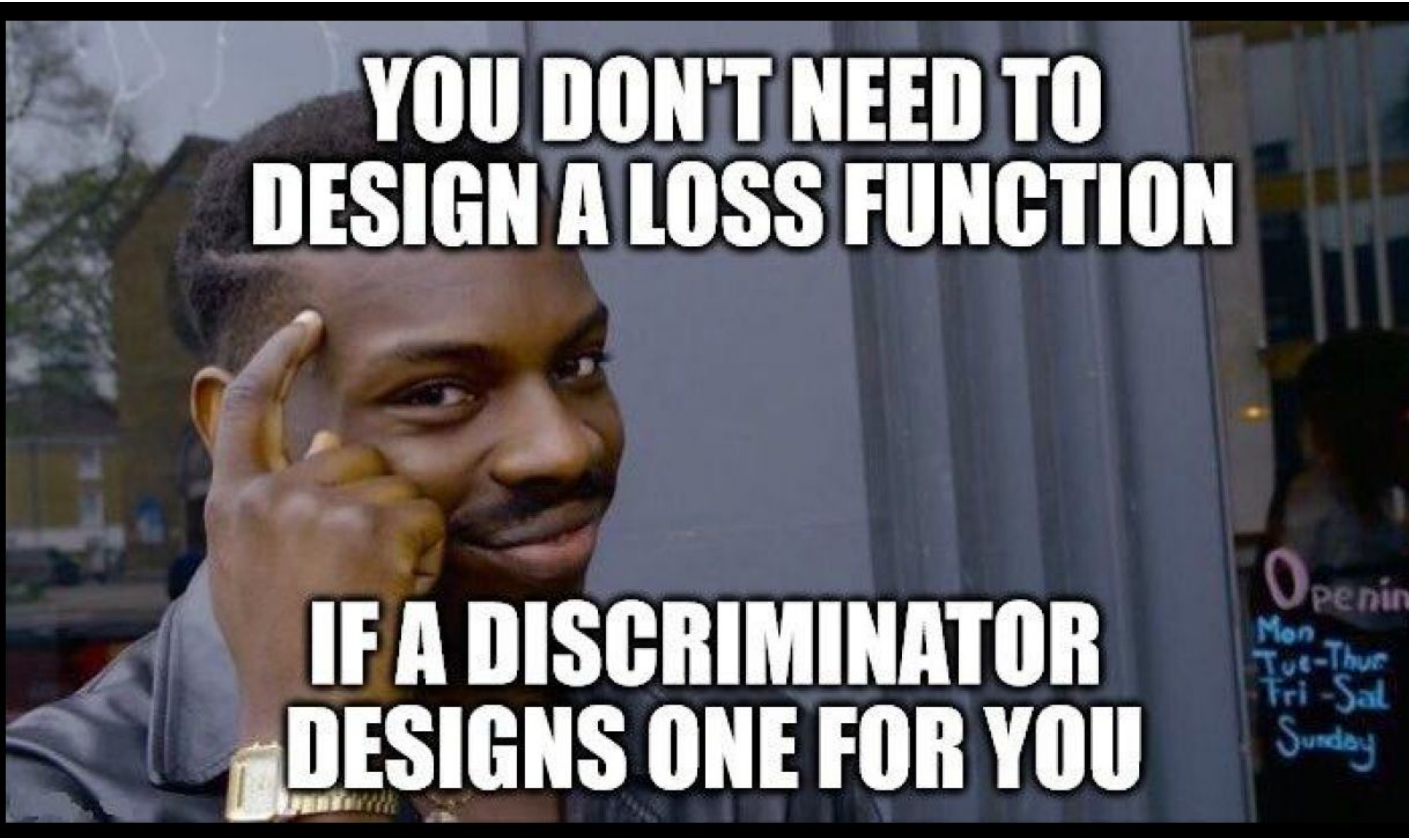
- **Discriminator** needs to:
  - Correctly classify real data:

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

  - Correctly classify wrong data:

$$\boxed{\max_D \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]} \qquad D(G(z)) \to 0$$

- 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 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:

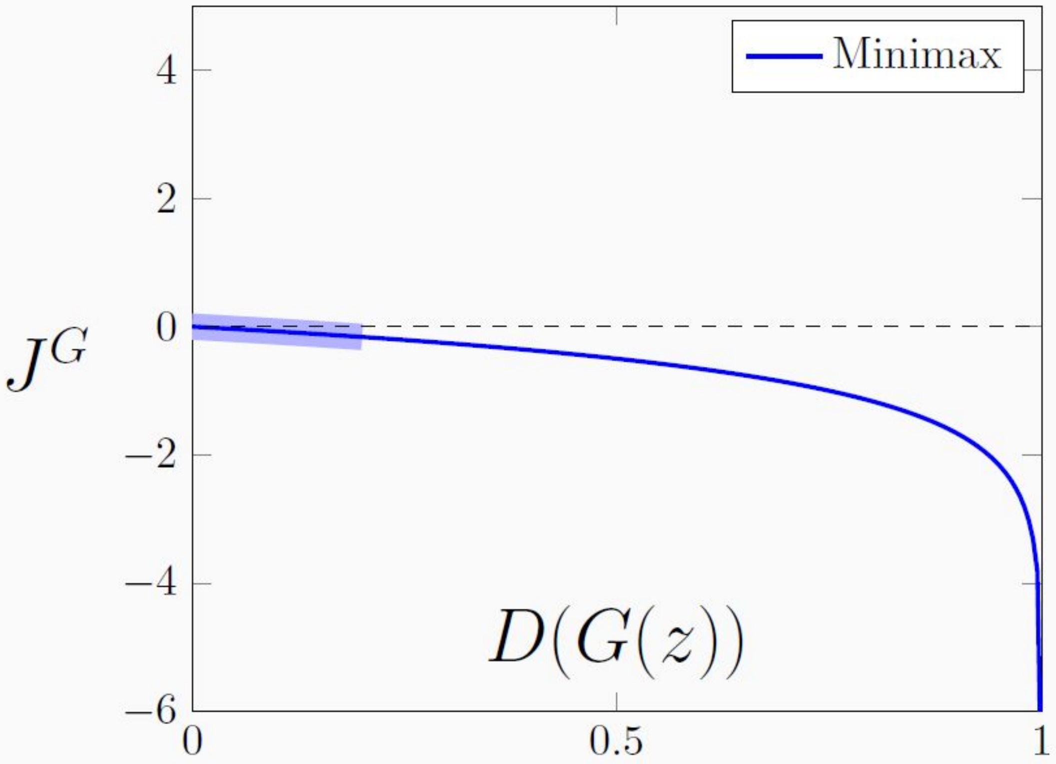
$$\boxed{\min_{G} \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]} \qquad D(G(z)) \to 1$$

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

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

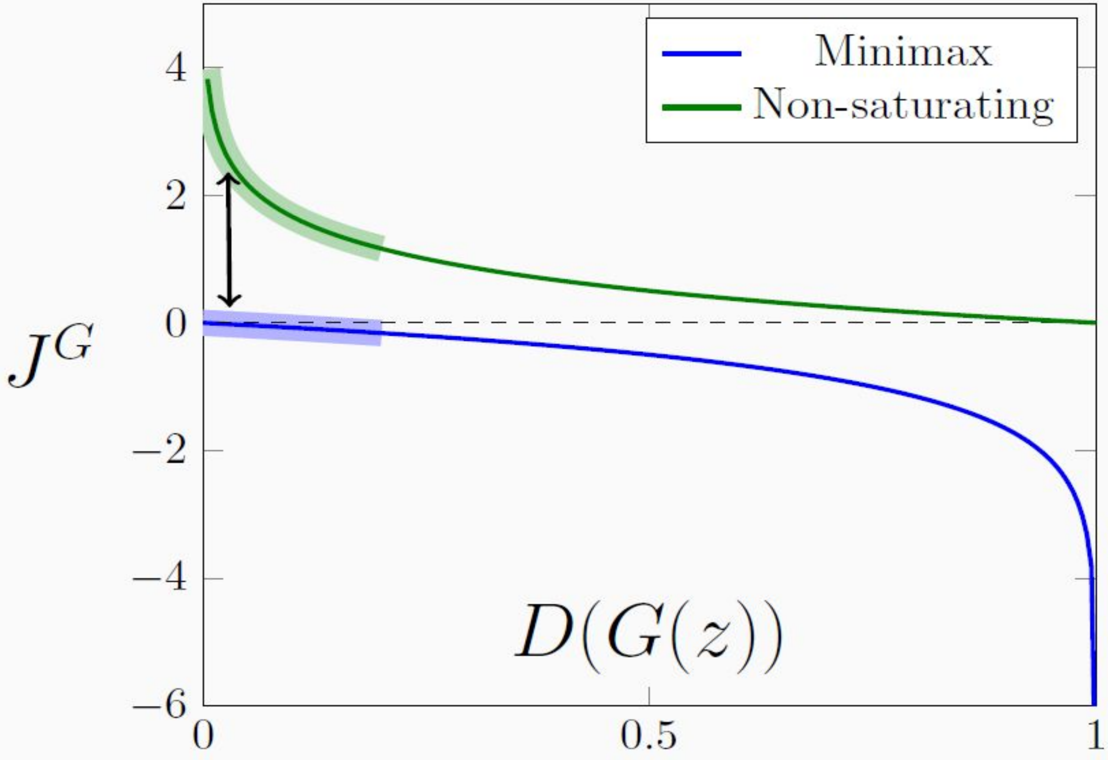
GANs - Generator Objectives

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



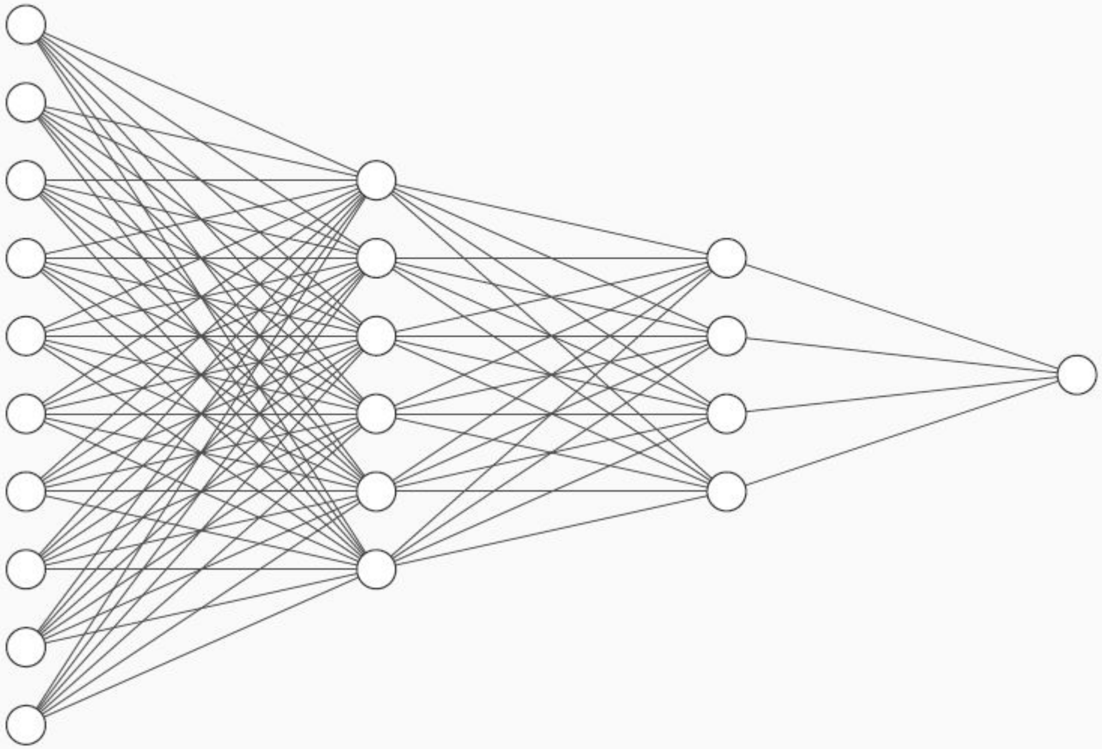
GANs - Generator Objectives

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



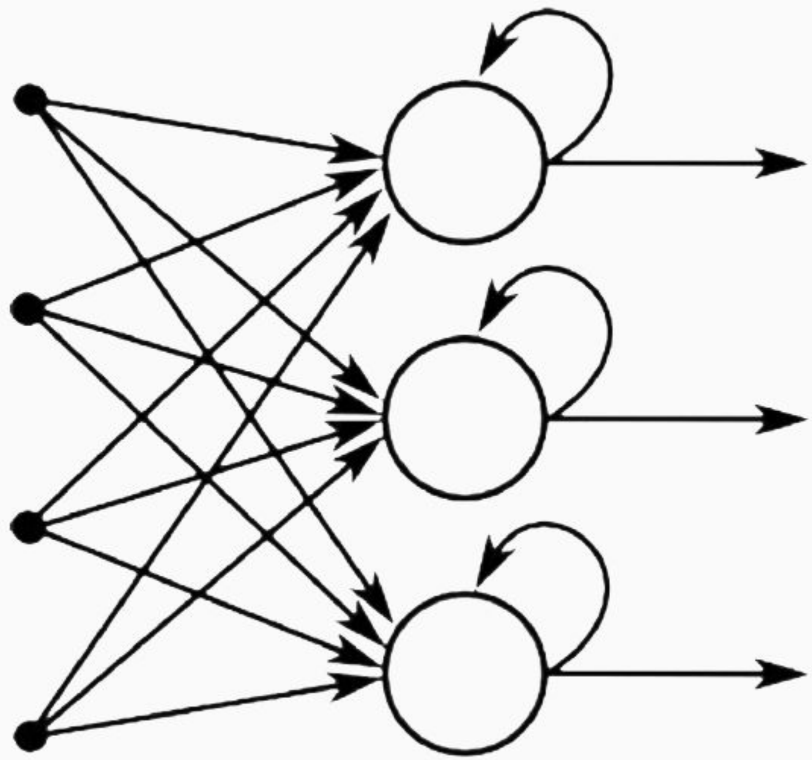
Models definition

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



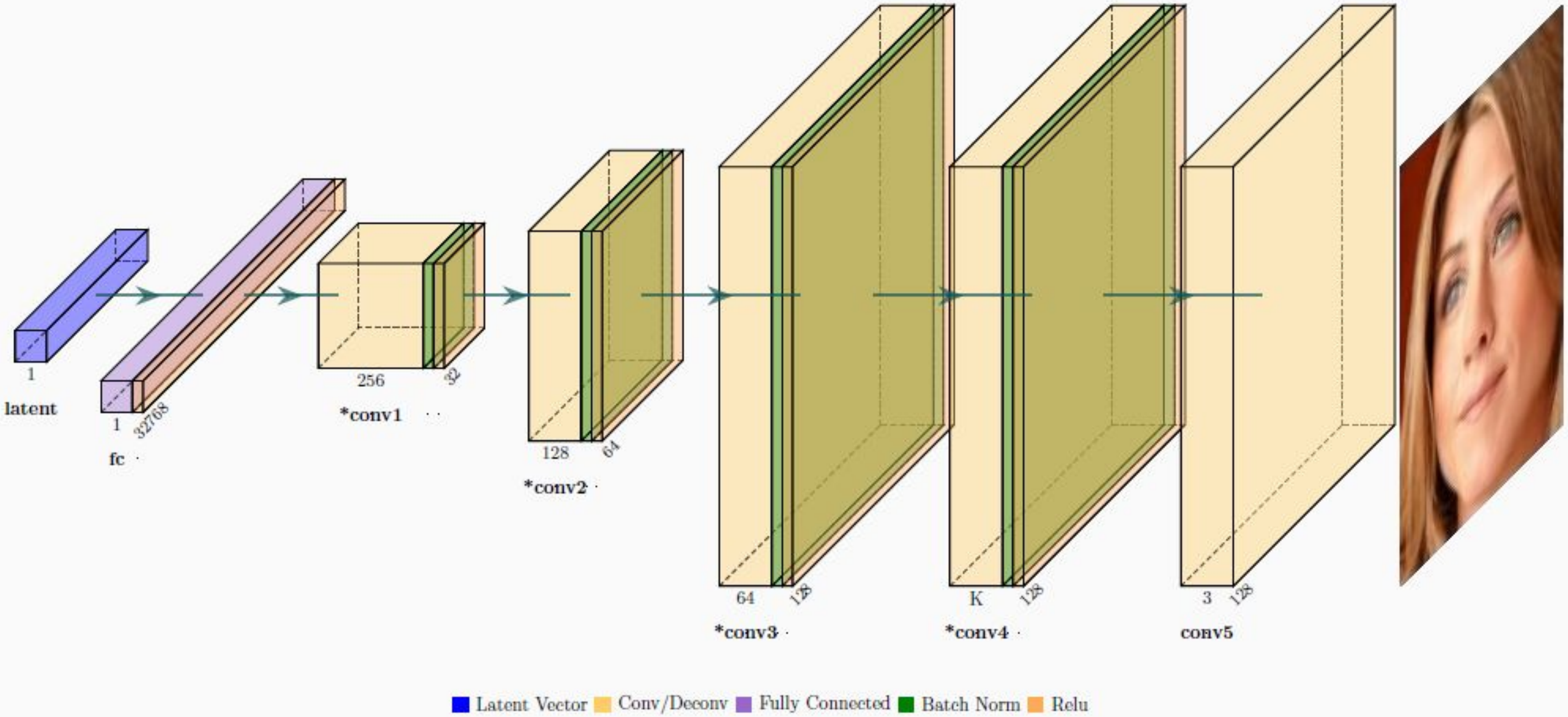
Models definition

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



Models definition

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



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 $z^{(1)}, \ldots, z^{(m)}$ 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**:

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

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

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 $p_z(z)$

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**:

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

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

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

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.

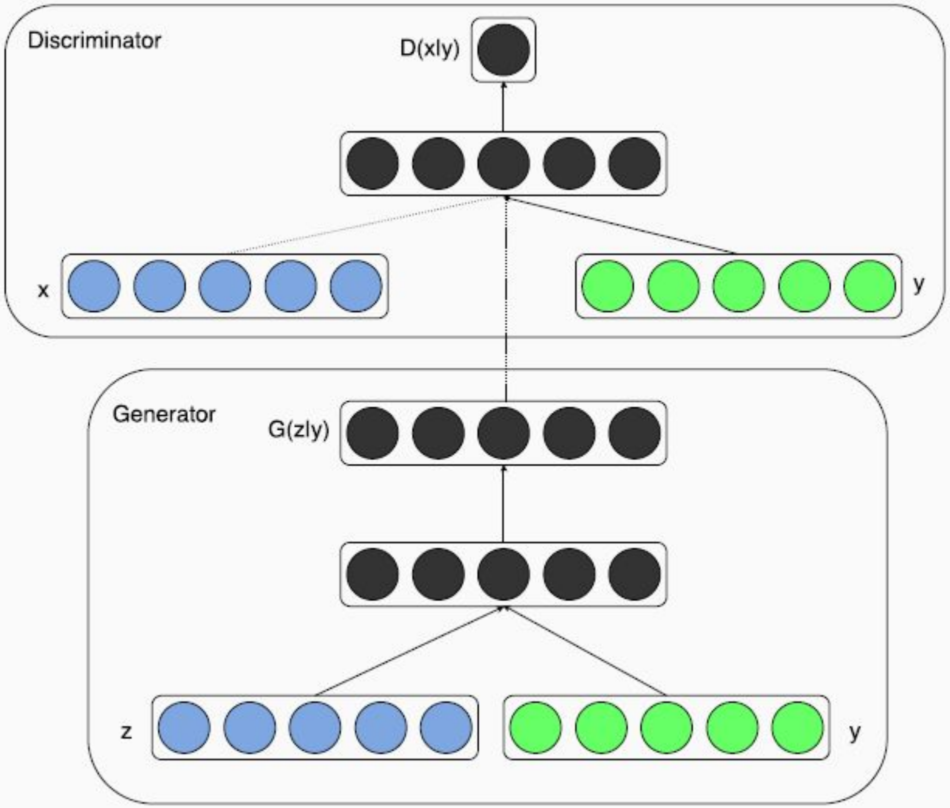


Figure 2: From Mirza and Osindero (2014)

Conditional GANs

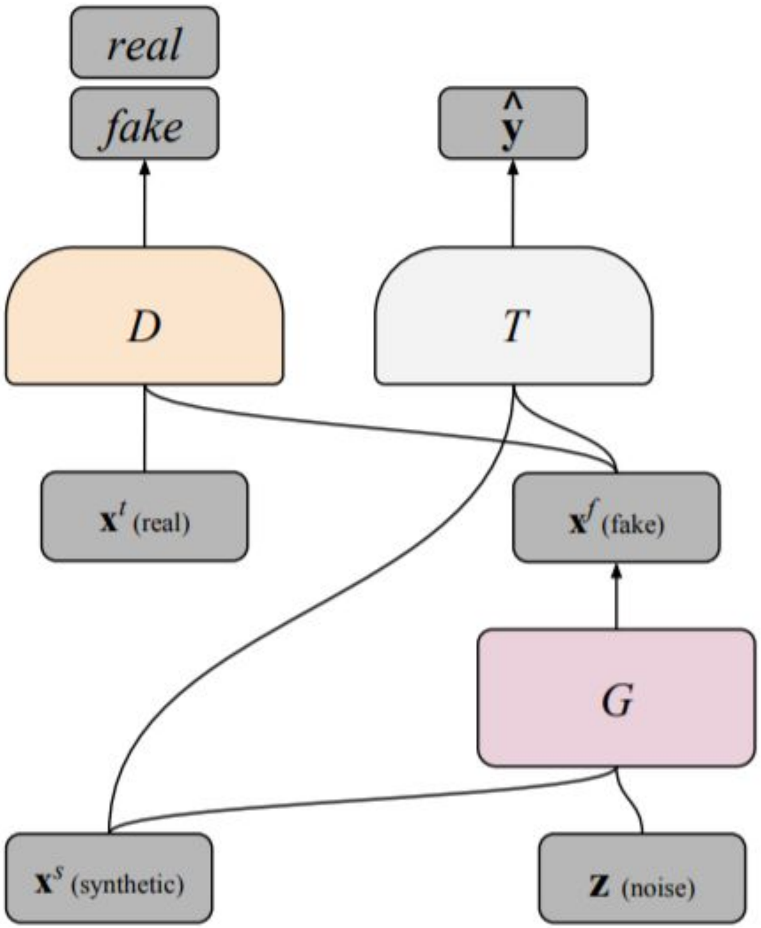
The GANs game becomes:

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

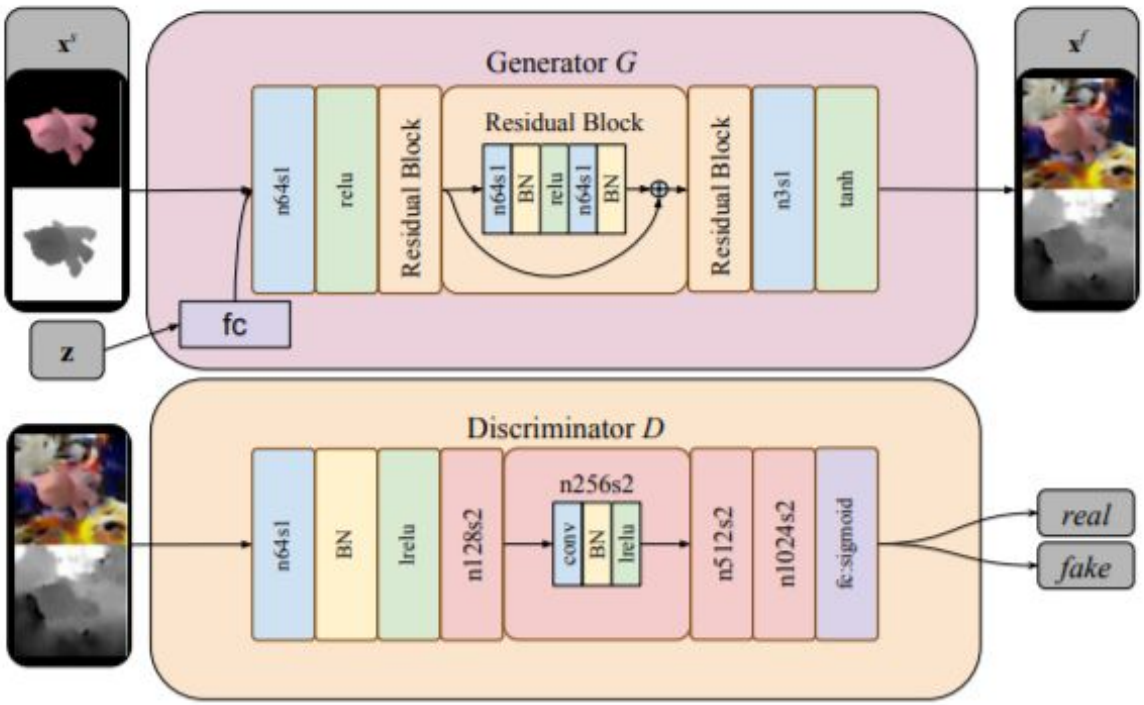
Notice: the same representation of the condition has to be presented to both network.

PixelDA

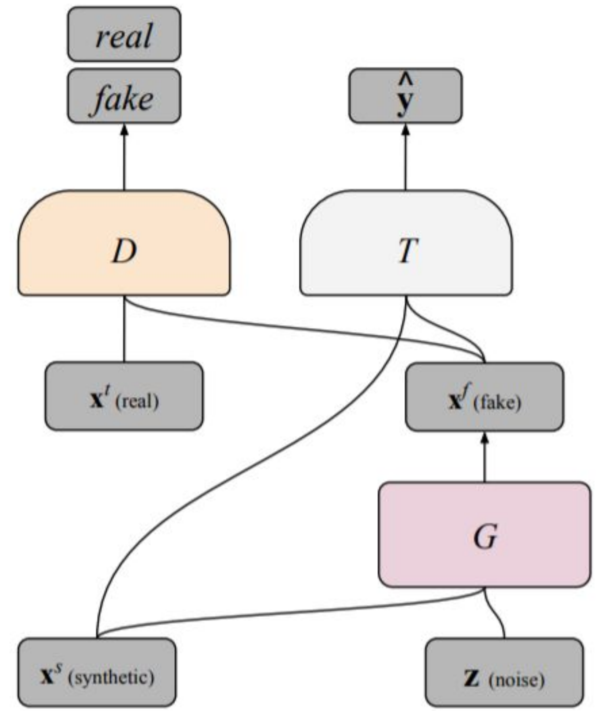
- *“Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks”*
- Developed by DeepMind (Google AI lab)
- It works by using a simple resnet-like generator and DCGAN-like discriminator, together with an auxiliar classifier!



PixelDA

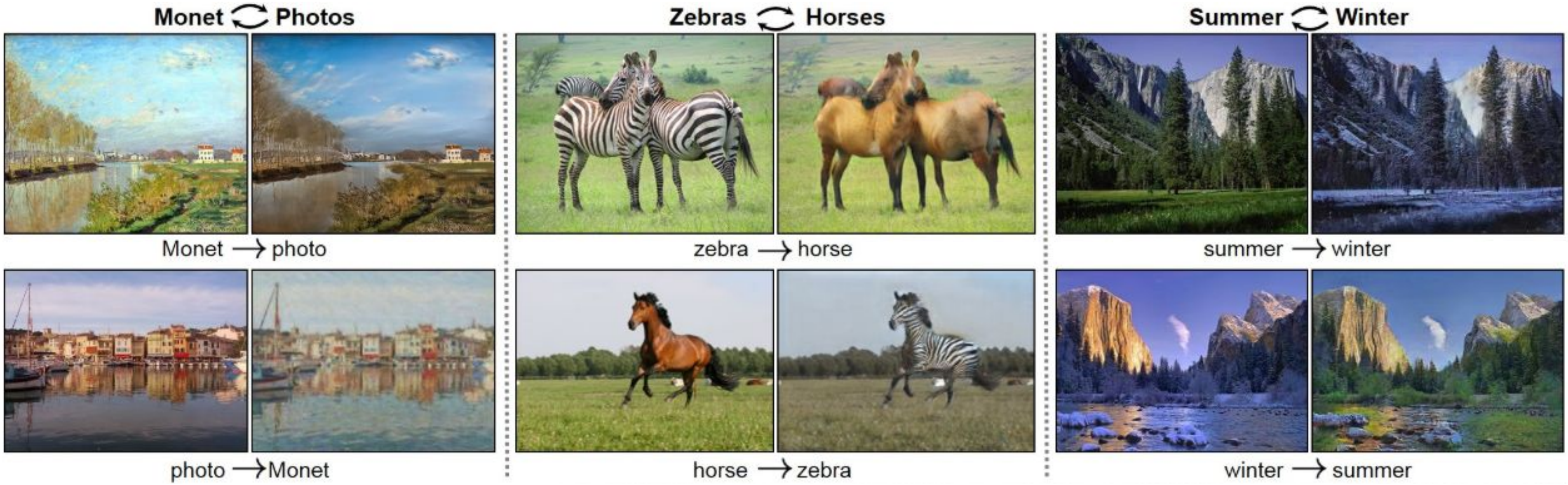


Generator G
Residual Block
n64s1
relu
BN
tanh
n3s1
fc
z
Discriminator D
n128s2
n256s2
conv
lrelu
n512s2
n1024s2
fc:sigmoid
real
fake



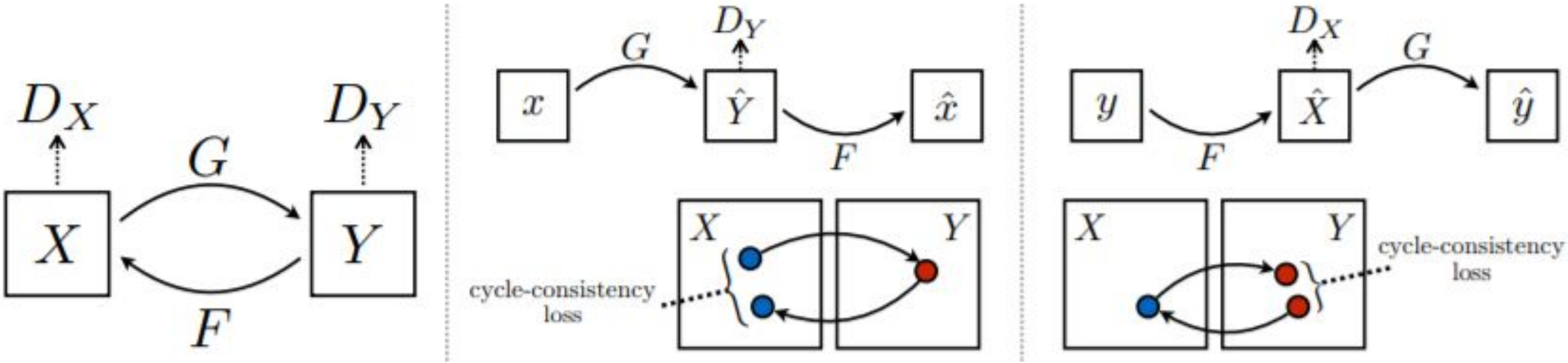
real
fake
D
T
G
(real)
(fake)
(synthetic)
(noise)

CycleGAN



- ICCV 2017 groundbreaking paper
- First work, together with DiscoGAN, that exploits 2 GAN models for source → target and target → source mapping at the same time!
- Used as baseline method for several other models

CycleGAN



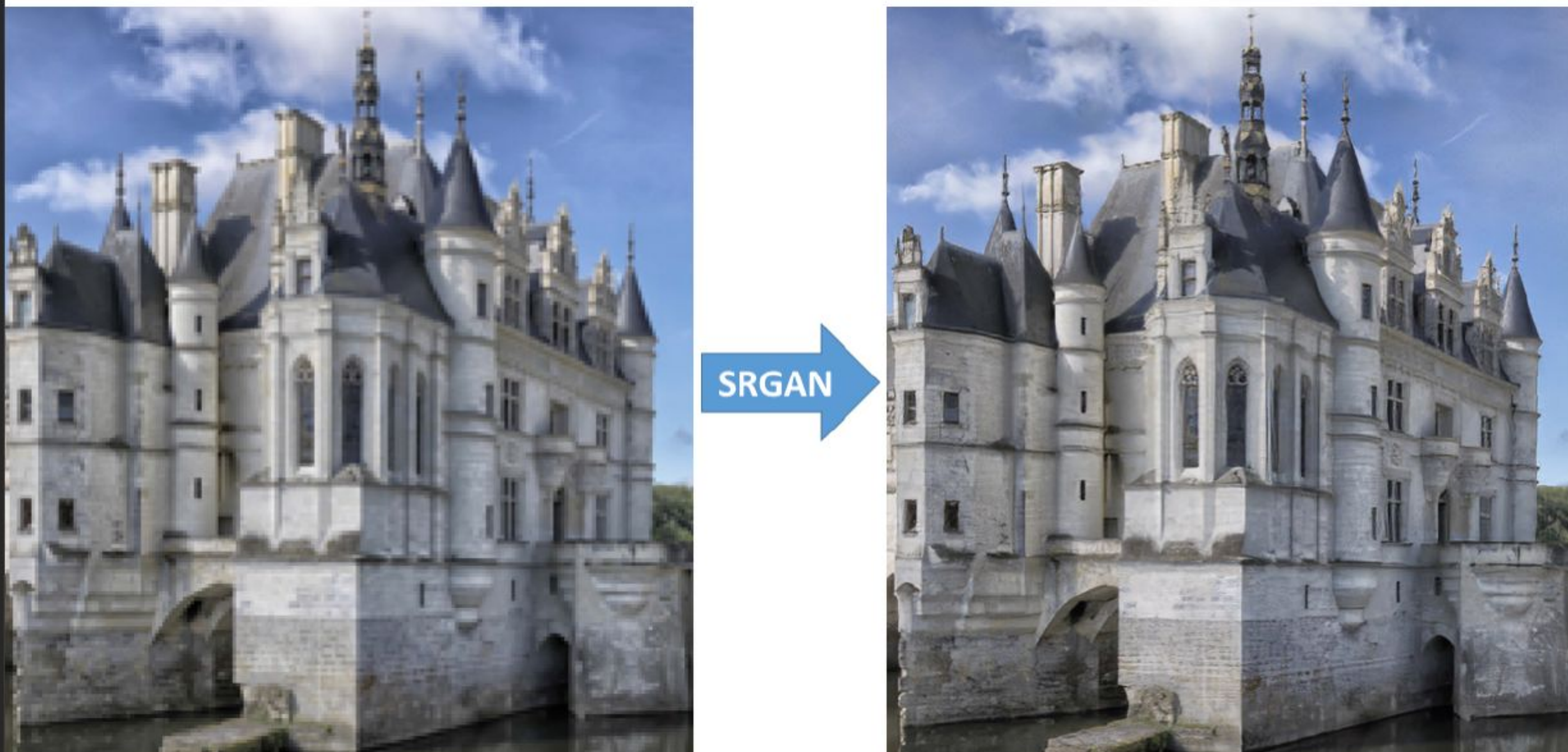
- One generator learns the function G from X to Y, another one learns the function F from Y to X. Each generator has a corresponding discriminator for real/fake traditional GAN training.
- In addition, a *cycle consistency loss* is added to the system:
  F(G(X)) ≈ X (and vice versa) → $|F(G(X))-X|_1$ to be minimized (L1 norm)

Conditional - Domain Translation
Isola et al. (2016)

Unconditional - Face Generation
Karras et al. (2017)

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

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



Project

- The project main theme is a standard training of a classifier
- If you own enough expertise and computational power, you can choose to implement and run a GAN method, or in general you can ask for more freedom of choice for a more complicated project.
- You choose the dataset for the training. Obviously, it must be labelled, at least 2 classes, and enough challenging for a deep learning run! If there is no training/test split, you do the splits (80/20).
- You have to run two separate experiments: one with a pretrained network (VGG, ResNet, Inception, or whatever you like!), and another one with a simple custom CNN you build (which will produce worse performance of course).

Project

- For each experiment, run a new training with 2-3 data augmentation techniques. If you want, try to play also with learning rate, with solvers,
- Write a presentation (slides) in which you tell us what you have done, what you was expecting, and what you got. Analyze the accuracy for each experiments and tell us your considerations – why this data augmentation technique worked so nicely? Why you chose to not use another particular data aug. ? Why your custom network failed so miserably? ;-)
- The project and presentation can be done in a group of 2 or 3 people. We encourage you to work in group, but if you are really a lone wolf, you can do it in solo.

Project

- Before starting the project, send me an email with your project proposal, in which you tell me which dataset you wanna use and a small description of your project – no project group can work on the same dataset! So, please send me a plan A (preferred) dataset, but also a plan B dataset if the first one have been already chosen by someone else!
- Do not, DO NOT, **DO NOT** copy code and experiments from other groups. It is a silly move, you will not learn anything good from that, and copy = failed exam. The main aim of this project is to make you learn how to work with deep learning ☺

That's all!