GAN - Theory and Applications

Michele De Simoni
Paolo Galeone
Emanuele Ghelfi
Federico Di Mattia

March 15, 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.

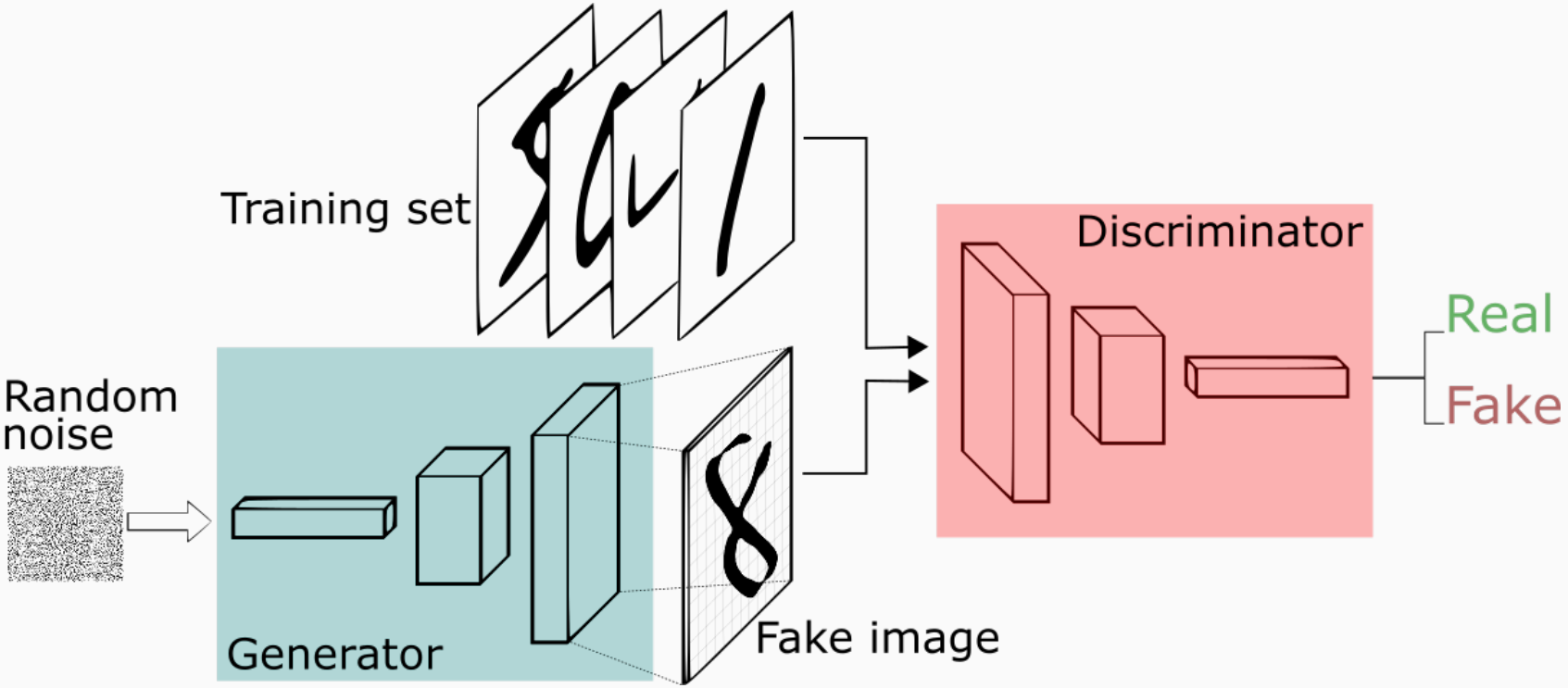


Figure 1: Credits: Reference

Generative Adversarial Networks

GANs game:

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

Generative Adversarial Networks

GANs game:

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

Generative Adversarial Networks

GANs game:

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

GANs - Discriminator

Intuitive explanation:

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

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

  - Correctly classify wrong data:

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

- The discriminator is an **adaptive loss function** that gets discarded once the generator has been trained.

GANs - Generator

Intuitive explanation:

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

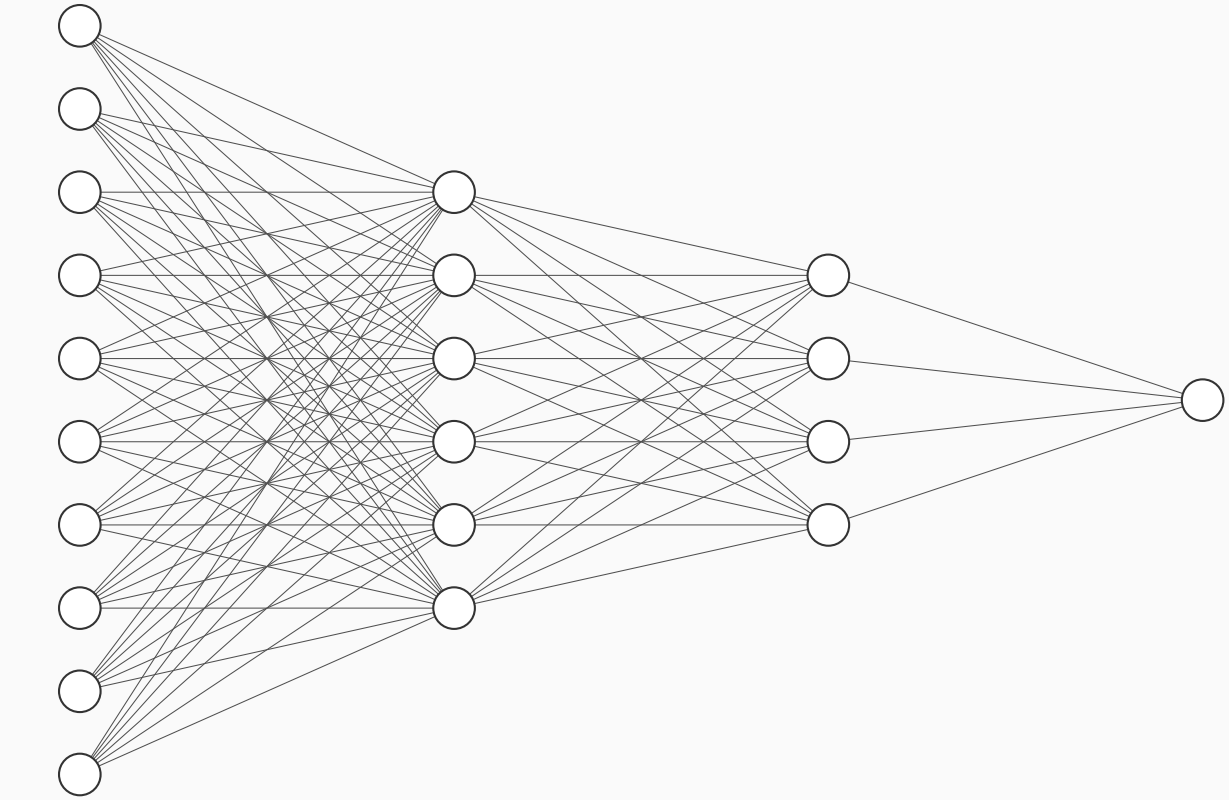
$$\min_G \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]. \tag{4}$$

  - Saturates easily Goodfellow et al. (2014).
  - Change loss for generator:

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

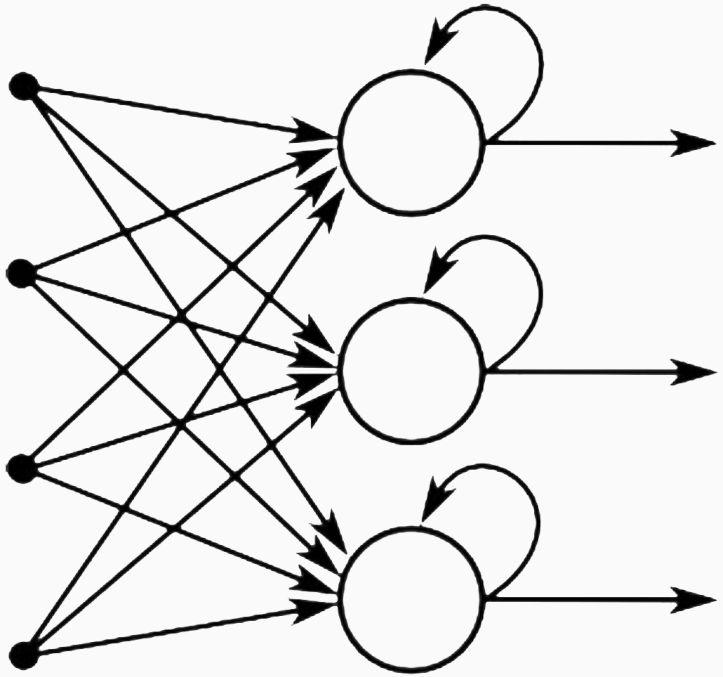
GANs - Models definition

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



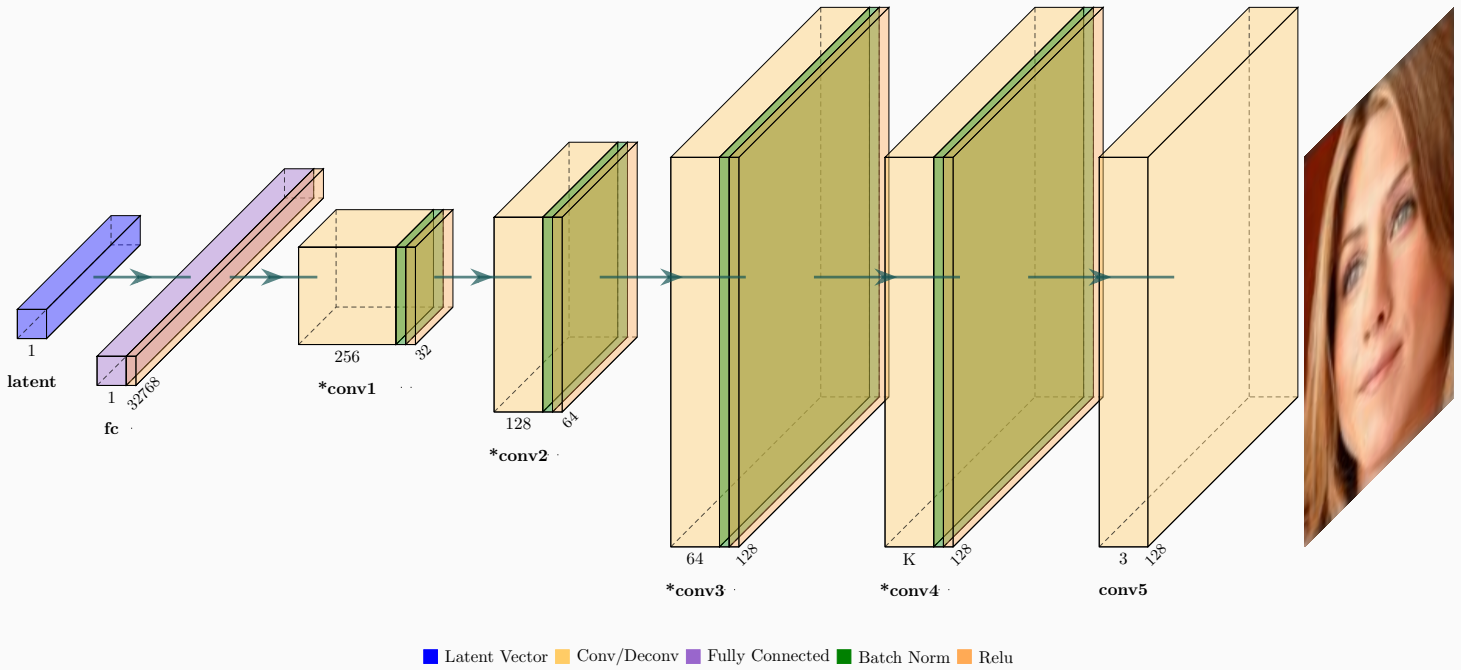
GANs - Models definition

- 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.
- **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**:

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

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**:

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

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**:

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

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**:

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

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**:

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

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**:

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

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.
- Goal: reach a **Nash Equilibrium** where the best D can do is random guessing.

Type of GANs

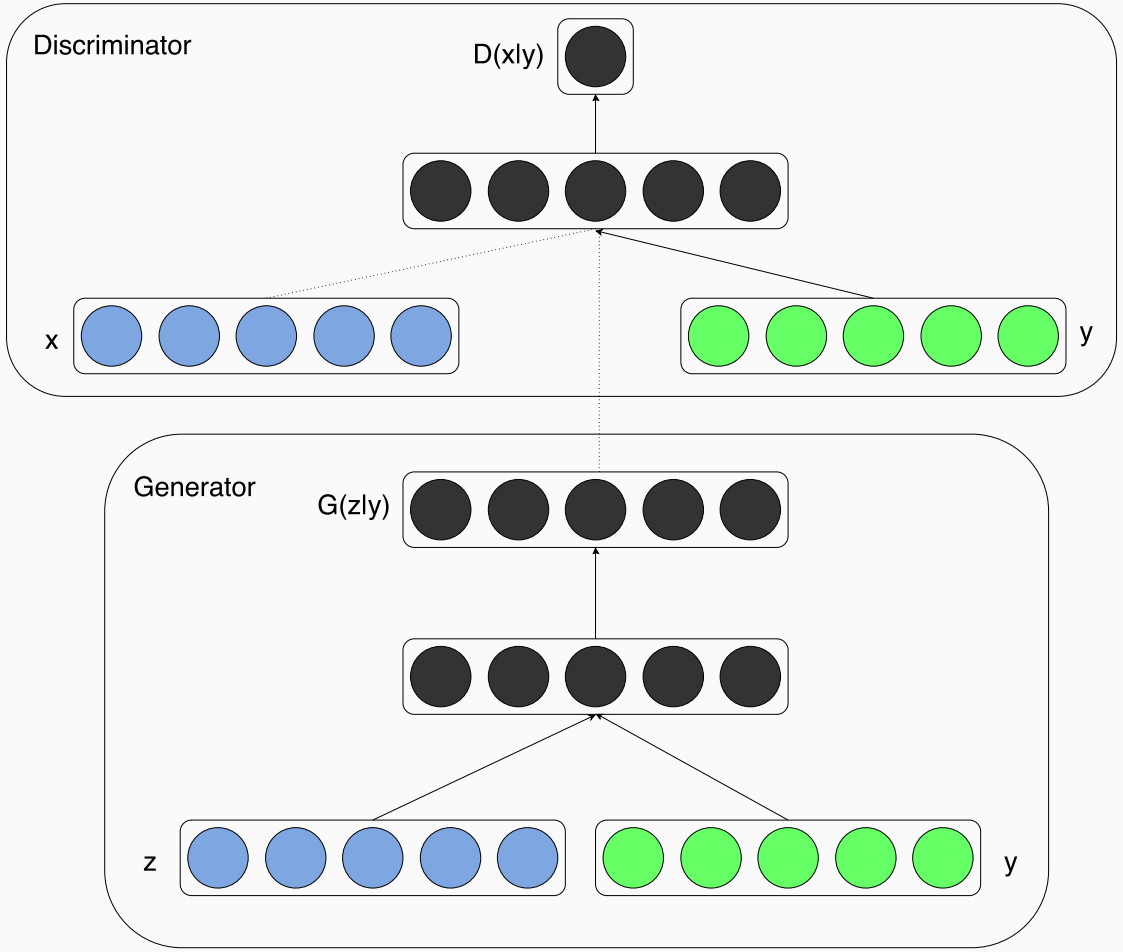
Types of GANs

Two big families:

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



Conditional GANs

The GANs game becomes:

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

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

GANs Applications

GANs Applications

GANs can be applied to lot of different tasks:

- Face generation

Figure 2: From Karras et al.

GANs Applications

GANs can be applied to lot of different tasks:

- Domain Translation

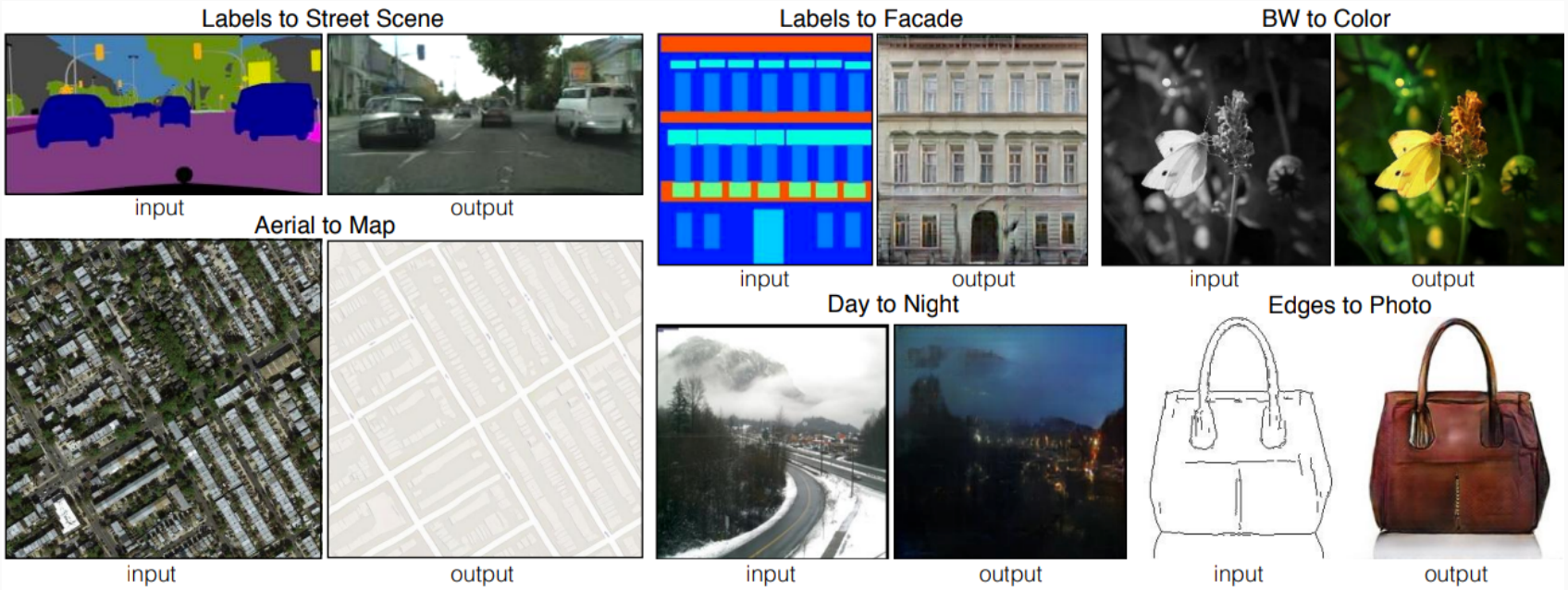


Figure 2: From Isola et al.

GANs Applications

GANs can be applied to lot of different tasks:

- Super resolution applications

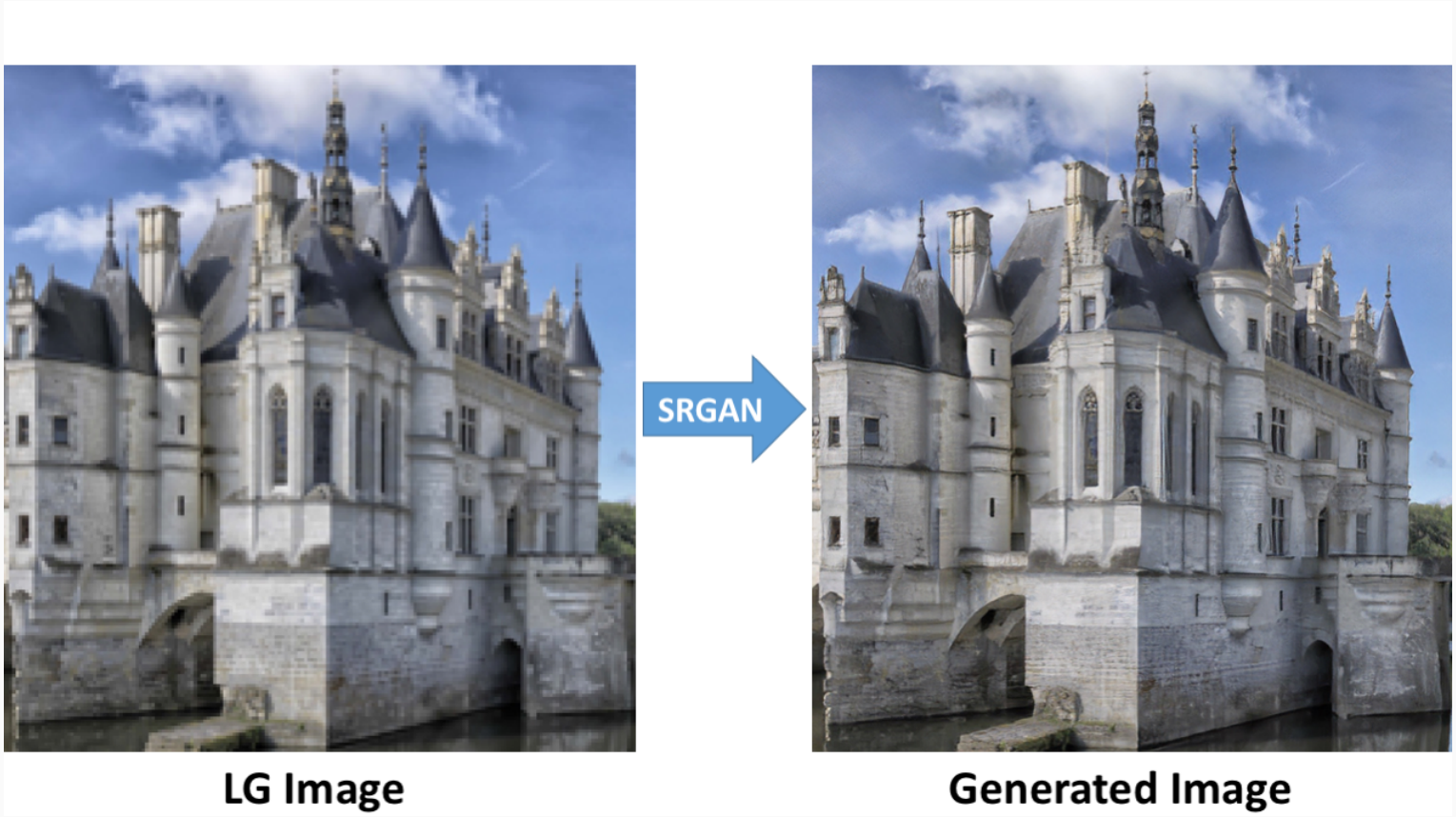


Figure 2: From Ledig et al.

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.] Isola, Phillip ; Zhu, Jun-Yan ; Zhou, Tinghui ; Efros, Alexei A.: Image-to-Image Translation with Conditional Adversarial Networks. . –

[Karras et al.] Karras, Tero ; Aila, Timo ; Laine, Samuli ; Lehtinen, Jaakko: Progressive Growing of GANs for Improved Quality, Stability, and Variation. . –

[Ledig et al.] 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. . –

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