

# Deep Learning & Applied AI

Adversarial training

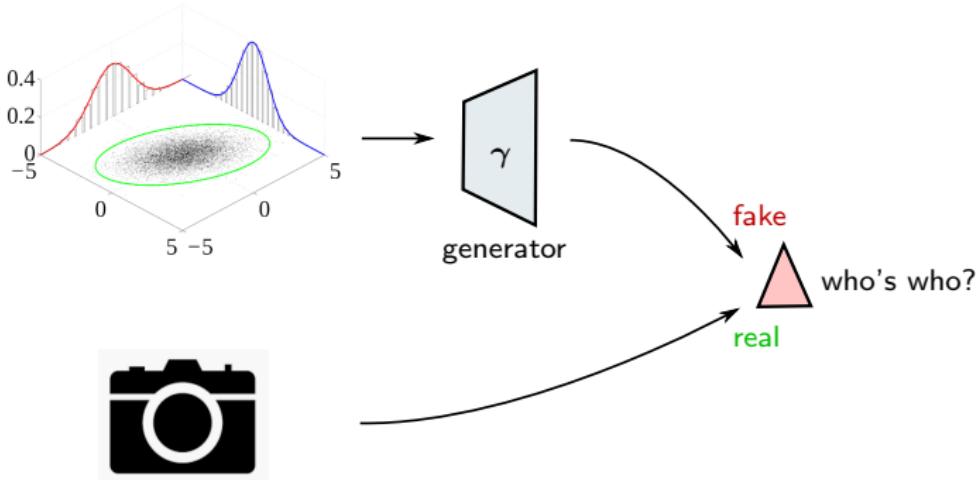
Emanuele Rodolà  
[rodola@di.uniroma1.it](mailto:rodola@di.uniroma1.it)



SAPIENZA  
UNIVERSITÀ DI ROMA

# Generative adversarial networks (GAN)

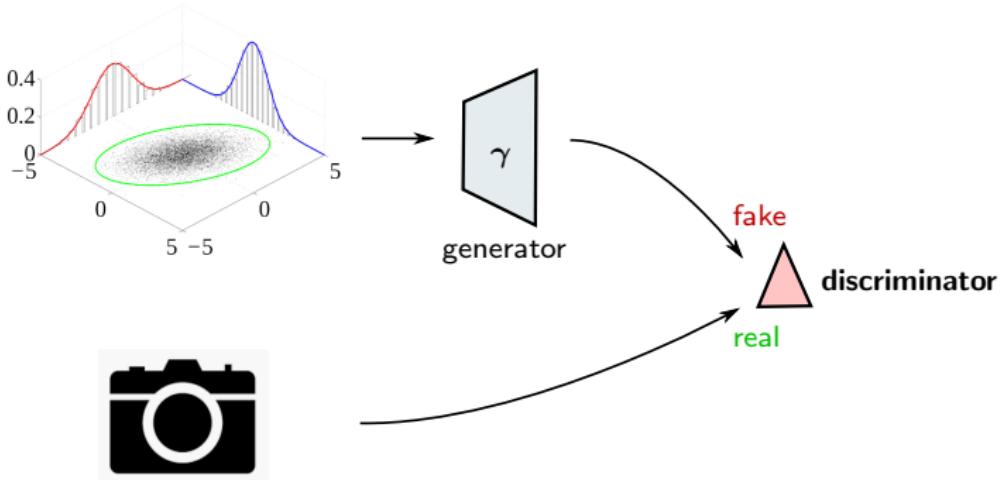
Synthesize data points from a given **generator** (e.g. a VAE decoder), and sample real data from the **actual** data distribution:



Given instances of fake and real data, a generative model is “good” when you can not **distinguish** between the two.

# Generative adversarial networks (GAN)

Synthesize data points from a given **generator** (e.g. a VAE decoder), and sample real data from the **actual** data distribution:



Let us train a **discriminator**  $\Delta$  with parameters  $\delta$ , whose output is the **probability** that the given data is real.

# Generative adversarial networks (GAN)

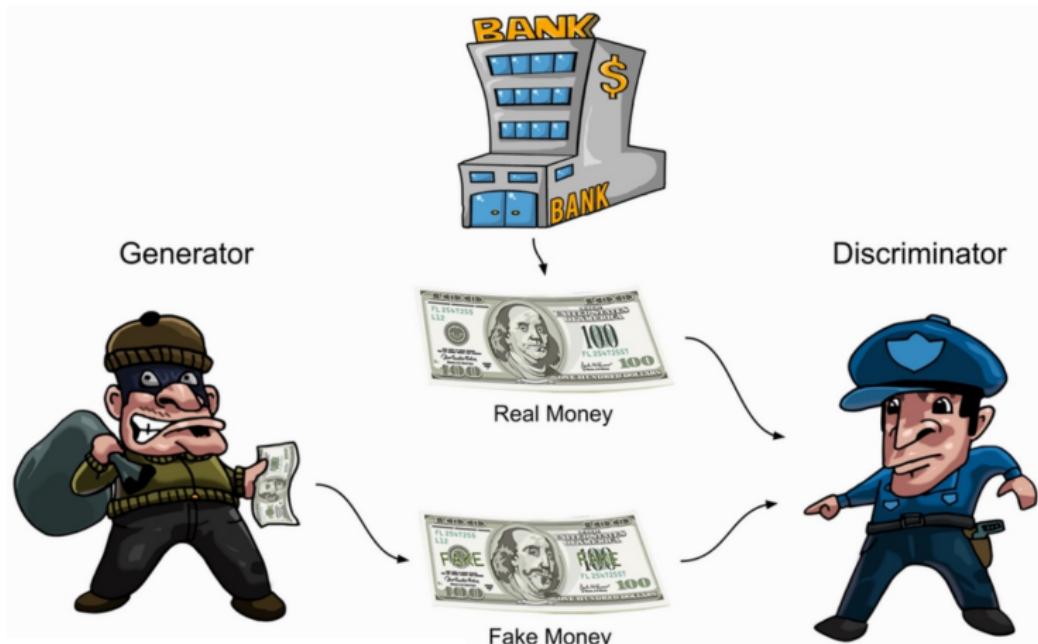


Figure: Rowel Atienza

# Generative adversarial networks (GAN)

Is this idea mathematically grounded?

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

# Generative adversarial networks (GAN)

$p_g$  è la distribuzione dei campioni generati, mentre  $p_{\text{real}}$  è la distribuzione reale

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

valore atteso = media pesata  $\underbrace{\mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x})}_{\text{real data}}$

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

$$\underbrace{\mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x})}_{\text{real data}} \quad \underbrace{\mathbb{E}_{\mathbf{z} \sim \mathcal{N}} \log(1 - \Delta_\delta(\mathbf{x}'))}_{\text{fake data}}$$

 punteggio che esprime quanto bene il discriminatore va su dati reali (alto punteggio = sa riconoscere bene i dati reali da quelli finti)

 punteggio che esprime quanto bene il discriminatore va su dati finti (alto punteggio = sa riconoscere bene i dati finti da quelli reali)

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

$$\underbrace{\mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x})}_{\text{real data}} \quad \underbrace{\mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))}_{\text{fake data}}$$

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

$$\underbrace{\mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x})}_{\text{real data}} + \underbrace{\mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))}_{\text{fake data}}$$

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

$$\mathbb{E}_{\mathbf{x}} \log \underbrace{\Delta_\delta(\mathbf{x})}_{\approx 1} + \mathbb{E}_{\mathbf{z}} \log \underbrace{(1 - \Delta_\delta(D_\gamma(\mathbf{z})))}_{\approx 0} \text{ in forma di punteggio...}$$

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

Mathematically, we can use the following score:

$$\mathbb{E}_{\mathbf{x}} \log \underbrace{\Delta_\delta(\mathbf{x})}_{\approx 1} + \mathbb{E}_{\mathbf{z}} \log \underbrace{(1 - \Delta_\delta(D_\gamma(\mathbf{z})))}_{\approx 1 \text{ ...deve essere il più vicino a 1}}$$

fino a questo punto però il delta è dato.  
Noi invece vogliamo trovare il miglior  
discriminatore tra tutti quelli possibili. Quindi  
dobbiamo massimizzare il problema sui parametri  
del discriminatore

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

We train a **classifier** to distinguish between generated and real data:

$$\max_{\delta} \mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))$$

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

We train a **classifier** to distinguish between generated and real data:

$$\max_{\delta} \mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))$$

In contrast, the **generator** should make this value as small as possible:

$$\text{loss} = \min_{\gamma} \max_{\delta} \mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))$$

perché l'obiettivo di un buon generatore è quello di realizzare un dato che sia quanto più indistinguibile da uno originale. Pertanto vogliamo minimizzare il valore ottenuto

questo è un caso di doppia ottimizzazione (spesso chiamata **min-max optimization**). E' molto frequente.

# Generative adversarial networks (GAN)

- $\mathbf{x}$ : sample from the **real** distribution
- $\mathbf{x}' = D_\gamma(\mathbf{z})$ : **generated** sample, good if  $p_g(\mathbf{x}') \approx p_{\text{real}}(\mathbf{x})$

A good discriminator should yield  $\Delta_\delta(\mathbf{x}) \approx 1$  on the **real** instances and  $\Delta_\delta(\mathbf{x}') \approx 0$  on the **fake** instances.

We train a **classifier** to distinguish between generated and real data:

$$\max_{\delta} \mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))$$

In contrast, the **generator** should make this value as small as possible:

$$\min_{\gamma} \max_{\delta} \mathbb{E}_{\mathbf{x}} \log \Delta_\delta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}} \log(1 - \Delta_\delta(D_\gamma(\mathbf{z})))$$

The generator competes against the **adversarial** discriminator and tries to minimize its **success rate**.

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

Let us try to maximize the discriminator score given a generator  $\textcolor{red}{G}$ :

$$J(\textcolor{red}{G}) = \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x}))$$

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

Let us try to maximize the discriminator score given a generator  $\textcolor{red}{G}$ :

$$\begin{aligned} J(\textcolor{red}{G}) &= \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x})) \\ &= \max_{\delta} \int [\log \Delta_{\delta}(\mathbf{x}) p_{\text{real}}(\mathbf{x}) + \log(1 - \Delta_{\delta}(\mathbf{x})) p_g(\mathbf{x})] d\mathbf{x} \end{aligned}$$

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

Let us try to maximize the discriminator score given a generator  $\textcolor{red}{G}$ :

$$\begin{aligned} J(\textcolor{red}{G}) &= \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x})) \\ &= \max_{\delta} \int [\log \Delta_{\delta}(\mathbf{x}) p_{\text{real}}(\mathbf{x}) + \log(1 - \Delta_{\delta}(\mathbf{x})) p_g(\mathbf{x})] d\mathbf{x} \end{aligned}$$

For any given  $\mathbf{x}$ , we want to maximize  $\Delta_{\delta}(\mathbf{x}) = a$ ; let's rename for simplicity  $p_{\text{real}}(\mathbf{x}) \equiv p$  and  $p_g(\mathbf{x}) \equiv q$ , we get to:

$$\max_a p \log \textcolor{green}{a} + \textcolor{red}{q} \log(1 - \textcolor{green}{a})$$

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

Let us try to maximize the discriminator score given a generator  $\mathcal{G}$ :

$$\begin{aligned} J(\mathcal{G}) &= \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x})) \\ &= \max_{\delta} \int [\log \Delta_{\delta}(\mathbf{x}) p_{\text{real}}(\mathbf{x}) + \log(1 - \Delta_{\delta}(\mathbf{x})) p_g(\mathbf{x})] d\mathbf{x} \end{aligned}$$

For any given  $\mathbf{x}$ , we want to maximize  $\Delta_{\delta}(\mathbf{x}) = a$ ; let's rename for simplicity  $p_{\text{real}}(\mathbf{x}) \equiv p$  and  $p_g(\mathbf{x}) \equiv q$ , we get to:

$$\max_a p \log a + q \log(1 - a)$$

This is maximized when the derivative w.r.t.  $a$  is zero:

$$\frac{p}{a} - \frac{q}{1-a} = 0$$

perché questa funzione sopra è concava, quindi la soluzione si trova ponendo la derivata = 0

# Generative adversarial networks (GAN)

Assume  $\mathbf{x} \sim p_g$  for the generated data, and  $\mathbf{x} \sim p_{\text{real}}$  for the real data.

Let us try to maximize the discriminator score given a generator  $\mathcal{G}$ :

$$\begin{aligned} J(\mathcal{G}) &= \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x})) \\ &= \max_{\delta} \int [\log \Delta_{\delta}(\mathbf{x}) p_{\text{real}}(\mathbf{x}) + \log(1 - \Delta_{\delta}(\mathbf{x})) p_g(\mathbf{x})] d\mathbf{x} \end{aligned}$$

For any given  $\mathbf{x}$ , we want to maximize  $\Delta_{\delta}(\mathbf{x}) = a$ ; let's rename for simplicity  $p_{\text{real}}(\mathbf{x}) \equiv p$  and  $p_g(\mathbf{x}) \equiv q$ , we get to:

$$\max_a p \log a + q \log(1 - a)$$

This is maximized when the derivative w.r.t.  $a$  is zero:

$$a = \frac{p}{p + q}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\textcolor{green}{a} = \frac{p}{p + \textcolor{red}{q}}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

quindi questo è il  
discriminante ottimale  
con generatore fissato

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back into the main functional:

$$J(\mathbf{G}) = \max_{\delta} \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \Delta_{\delta}(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\delta}(\mathbf{x}))$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back into the main functional:

$$J(G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back into the main functional:

$$J(\mathbf{G}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Let us define the distribution  $\rho = \frac{1}{2}p_{\text{real}} + \frac{1}{2}p_g$ . We get:

$$J(\mathbf{G}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{2\rho(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{2\rho(\mathbf{x})}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back into the main functional:

$$J(\mathbf{G}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Let us define the distribution  $\rho = \frac{1}{2}p_{\text{real}} + \frac{1}{2}p_g$ . We get:

$$J(\mathbf{G}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{\rho(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{\rho(\mathbf{x})} + \text{const.}$$

# Generative adversarial networks (GAN)

We thus have a closed-form solution for the optimal discriminator:

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back into the main functional:

$$J(\mathbf{G}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{p_{\text{real}}(\mathbf{x}) + p_g(\mathbf{x})}$$

Let us define the distribution  $\rho = \frac{1}{2}p_{\text{real}} + \frac{1}{2}p_g$ . We get:

$$\begin{aligned} J(\mathbf{G}) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} \log \frac{p_{\text{real}}(\mathbf{x})}{\rho(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim p_g} \log \frac{p_g(\mathbf{x})}{\rho(\mathbf{x})} + \text{const.} \\ &= KL(p_{\text{real}} \parallel \rho) + KL(p_g \parallel \rho) + \text{const.} \end{aligned}$$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$J(\textcolor{red}{G}) = KL(p_{\text{real}}\|\rho) + KL(\textcolor{red}{p}_g\|\rho) + \text{const.}$$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$\min_{p_g} KL(p_{\text{real}}\|\rho) + KL(p_g\|\rho) + \text{const.}$$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$\min_{p_g} KL(p_{\text{real}}\|\rho) + KL(p_g\|\rho)$$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$\min_{p_g} \underbrace{KL(p_{\text{real}}\|\rho) + KL(p_g\|\rho)}_{\substack{2 \times \text{Jensen-Shannon divergence} \\ \text{between } p_{\text{real}} \text{ and } p_g}}$$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$\min_{p_g} \underbrace{KL(p_{\text{real}}\|\rho) + KL(p_g\|\rho)}_{\approx JS(p_{\text{real}}\|p_g)} \quad \text{qui è il discriminatore ad essere fissato (lo abbiamo trovato prima)}$$

Property:  $p_{\text{real}} = p_g \Leftrightarrow JS(p_{\text{real}}\|p_g) = 0$

# Generative adversarial networks (GAN)

Therefore, the optimal GAN **generator** is found by minimizing:

$$\min_{p_g} \underbrace{KL(p_{\text{real}}\|p) + KL(p_g\|p)}_{\approx JS(p_{\text{real}}\|p_g)}$$

Property:  $p_{\text{real}} = p_g \Leftrightarrow JS(p_{\text{real}}\|p_g) = 0$

Within the GAN paradigm, the globally optimal generator has a data distribution exactly equal to the **real** distribution of the data.



i dati di allenamento. Non conosciamo la reale distribuzione di tutti i dati, ma ci auspiciamo di impararla con quello che abbiamo

## Adversarial training

Training the generator on data produced by an adversary is an example of a more general concept called [adversarial training](#).

The generated data samples used for training are [adversarial examples](#).

# Adversarial training

Training the generator on data produced by an adversary is an example of a more general concept called [adversarial training](#).

The generated data samples used for training are [adversarial examples](#).

Adversarial examples can be used [maliciously](#).

quindi lo scopo di allenare una rete con samples "infetti" serve per migliorare la robustezza della rete stessa e quindi renderla meno vulnerabile agli attacchi



"speed limit 50mph"

# Adversarial attacks



Szegedy et al, "Intriguing properties of neural networks", 2013

# Adversarial attacks

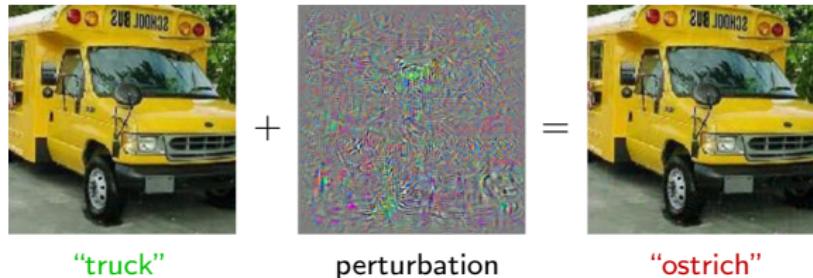


“truck”



“ostrich”

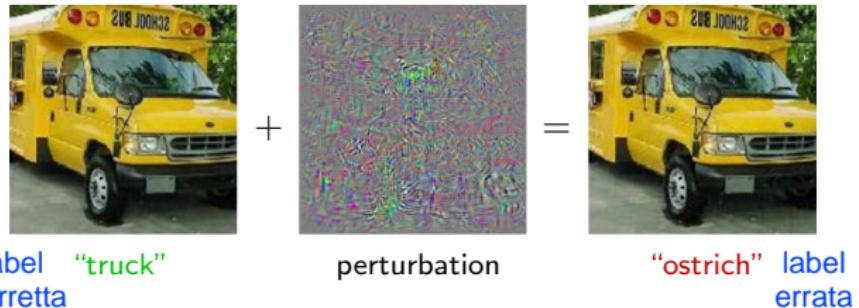
# Adversarial attacks



The perturbation can be explicitly [optimized](#) for.

Adversarial attacks can cause a system to take unwanted actions.

# Adversarial attacks



The perturbation can be explicitly [optimized](#) for.

Adversarial attacks can cause a system to take unwanted actions.

How to construct [undetectable](#) adversarial examples?

# Types of attack

Distinction based on the amount of information available to the attacker:

- Black-box attack:

Can only query the target model.

# Types of attack

Distinction based on the amount of information available to the attacker:

- **Black-box attack:**

Can only query the target model.

- **Gray-box attack:**

Access to partial information (only the features, architecture, etc.).

# Types of attack

Distinction based on the amount of information available to the attacker:

- **Black-box attack:**

Can only query the target model.

- **Gray-box attack:**

Access to partial information (only the features, architecture, etc.).

- **White-box attack:**

Complete access to the network (architecture, parameters, etc.).

# Types of attack

Distinction based on the amount of information available to the attacker:

- **Black-box attack:**

Can only query the target model.

- **Gray-box attack:**

Access to partial information (only the features, architecture, etc.).

- **White-box attack:**

Complete access to the network (architecture, parameters, etc.).

You are **not allowed** to touch the network weights.

## Targeted attacks

Given an input sample  $\mathbf{x}$ , a classifier  $C$ , and a **target** class  $t$ , consider:

$$\begin{aligned} \min_{\mathbf{x}' \in [0,1]^n} \quad & \|\mathbf{x} - \mathbf{x}'\|_2^2 \\ \text{s.t. } & C(\mathbf{x}') = t \end{aligned}$$

## Targeted attacks

Given an input sample  $\mathbf{x}$ , a classifier  $C$ , and a **target** class  $t$ , consider:

$$\begin{aligned} \min_{\mathbf{x}' \in [0,1]^n} \quad & \|\mathbf{x} - \mathbf{x}'\|_2^2 \\ \text{s.t. } & C(\mathbf{x}') = t \end{aligned}$$

Relax the difficult constraint to a penalty term:

$$\min_{\mathbf{x}' \in [0,1]^n} \|\mathbf{x} - \mathbf{x}'\|_2^2 + c L(\mathbf{x}', t)$$

where  $L$  is the cross-entropy loss.

# Targeted attacks

$t$  è fisso (non generico)

Given an input sample  $\mathbf{x}$ , a classifier  $C$ , and a **target** class  $t$ , consider:

$$\begin{aligned} \min_{\mathbf{x}' \in [0,1]^n} \|\mathbf{x} - \mathbf{x}'\|_2^2 \\ \text{s.t. } C(\mathbf{x}') = t \end{aligned}$$

quindi il generatore deve minimizzare la differenza (in questo caso data dalla norma  $L_2$ ) tra il sample originale  $\mathbf{x}$  e uno generato  $\mathbf{x}'$  tale che il classificatore  $C$  etichetti  $\mathbf{x}'$  con  $t$

Relax the difficult constraint to a penalty term:

$$\min_{\mathbf{x}' \in [0,1]^n} \|\mathbf{x} - \mathbf{x}'\|_2^2 + c L(\mathbf{x}', t)$$

where  $L$  is the cross-entropy loss. se  $\mathbf{x}'$  viene classificato come  $t$ , la cross entropy da' un valore piccolo

$c > 0$  is a trade-off parameter that is chosen as small as possible; it can be found via **line search**.

$c$  bilancia il fatto di trovare un'immagine simile a  $\mathbf{x}$  e allo stesso tempo che venga classificata come  $t$

## Targeted attacks

A more general approach is given by:

$$\min_{\delta \in [0,1]^n} d(\mathbf{x}, \mathbf{x} + \delta) \quad \text{s.t. } C(\mathbf{x} + \delta) = t$$

where the **perturbation**  $\delta$  appears explicitly, and  $d$  is the **data term** (e.g. a distance) that depends on the specific task.

## Targeted attacks

A more general approach is given by:

$$\begin{aligned} \min_{\delta \in [0,1]^n} \quad & d(\mathbf{x}, \mathbf{x} + \delta) \\ \text{s.t. } & f(\mathbf{x} + \delta) \leq 0 \end{aligned}$$



where the **perturbation**  $\delta$  appears explicitly, and  $d$  is the **data term** (e.g. a distance) that depends on the specific task.

$f$  is such that  $C(\mathbf{x} + \delta) = t$  if and only if  $f(\mathbf{x} + \delta) \leq 0$ .

## Targeted attacks

A more general approach is given by:

$$\min_{\boldsymbol{\delta} \in [0,1]^n} d(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}) + c f(\mathbf{x} + \boldsymbol{\delta})$$

where the **perturbation**  $\boldsymbol{\delta}$  appears explicitly, and  $d$  is the **data term** (e.g. a distance) that depends on the specific task.

$f$  is such that  $C(\mathbf{x} + \boldsymbol{\delta}) = t$  if and only if  $f(\mathbf{x} + \boldsymbol{\delta}) \leq 0$ .

# Targeted attacks

A more general approach is given by:

$$\min_{\delta \in [0,1]^n} d(\mathbf{x}, \mathbf{x} + \delta) + c f(\mathbf{x} + \delta)$$

where the **perturbation**  $\delta$  appears explicitly, and  $d$  is the **data term** (e.g. a distance) that depends on the specific task.

$f$  is such that  $C(\mathbf{x} + \delta) = t$  if and only if  $f(\mathbf{x} + \delta) \leq 0$ .

Possible instance of the problem above:

$$\min_{\delta \in [0,1]^n} \|\delta\|_p + c \left( \max_{\substack{\text{classe } i \\ \text{classe } t}} \{F(\mathbf{x} + \delta)_i - F(\mathbf{x} + \delta)_t\}^+ \right)$$

where  $F : \mathbf{x} \mapsto [0, 1]^k$  is the full neural network yielding a **probability distribution** over all  $k$  classes for a given input  $\mathbf{x}$ , and  $(a)^+ = \max(a, 0)$ .



## Untargeted attacks

If there is no specific target toward which to misclassify, consider the closed-form expression for a given input  $\mathbf{x}$  with ground-truth label  $\ell_{\text{gt}}$ :

$$\mathbf{x}' = \mathbf{x} + \alpha \underbrace{\text{sign}(\nabla L(\mathbf{x}, \ell_{\text{gt}}))}_{\text{perturbation}}$$

which adds a **perturbation** maximizing the cost.

## Untargeted attacks

If there is no specific target toward which to misclassify, consider the closed-form expression for a given input  $\mathbf{x}$  with ground-truth label  $\ell_{\text{gt}}$ :

$$\mathbf{x}' = \mathbf{x} + \alpha \underbrace{\text{sign}(\nabla L(\mathbf{x}, \ell_{\text{gt}}))}_{\text{perturbation}}$$

which adds a **perturbation** maximizing the cost.

For better control on the perturbation, one can apply the iterates:

$$\mathbf{x}'_{(i)} = \text{clip}_{\epsilon} \left( \mathbf{x}'_{(i-1)} + \alpha \text{sign} \left( \nabla L(\mathbf{x}'_{(i-1)}, \ell_{\text{gt}}) \right) \right)$$

with  $\mathbf{x}'_{(0)} = \mathbf{x}$ .

The **clip** operation projects back into an  $\epsilon$ -neighborhood from  $\mathbf{x}$ .

## Untargeted attacks

If there is no specific target toward which to misclassify, consider the closed-form expression for a given input  $\mathbf{x}$  with ground-truth label  $\ell_{\text{gt}}$ :

$$\mathbf{x}' = \mathbf{x} + \alpha \underbrace{\text{sign}(\nabla L(\mathbf{x}, \ell_{\text{gt}}))}_{\text{perturbation}}$$

$\ell_{\text{gt}}$  = label groundtruth di  $\mathbf{x}$   
 $\alpha$  = la forza di movimento (come nel SGD)  
 $\text{sign}$  = garantisce che la perturbazione venga applicata nella direzione che massimizza la perdita.

which adds a **perturbation** maximizing the cost.

For better control on the perturbation, one can apply the iterates:

$$\mathbf{x}'_{(i)} = \text{clip}_{\epsilon} \left( \mathbf{x}'_{(i-1)} + \alpha \text{sign} \left( \nabla L(\mathbf{x}'_{(i-1)}, \ell_{\text{gt}}) \right) \right)$$

with  $\mathbf{x}'_{(0)} = \mathbf{x}$ .

 assicura che la perturbazione rimanga nel limite di epsilon

The **clip** operation projects back into an  $\epsilon$ -neighborhood from  $\mathbf{x}$ .

This method is designed to be **fast**, since 1 iteration  $\approx$  1 backprop step.

## Example: Targeted attack for adversarial training

Using adversarial examples as **training** data improves the **robustness** of the attacked learning model:



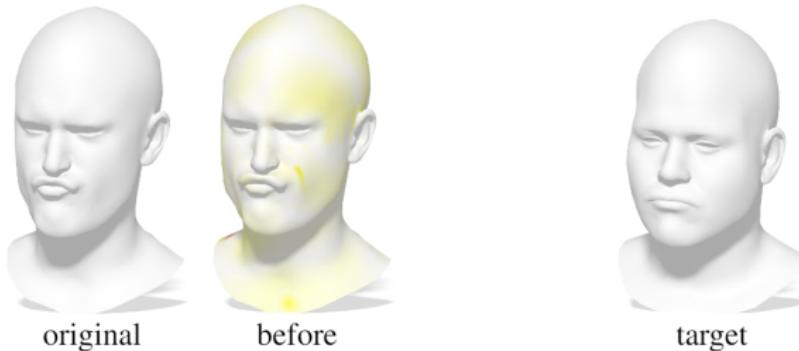
original



target

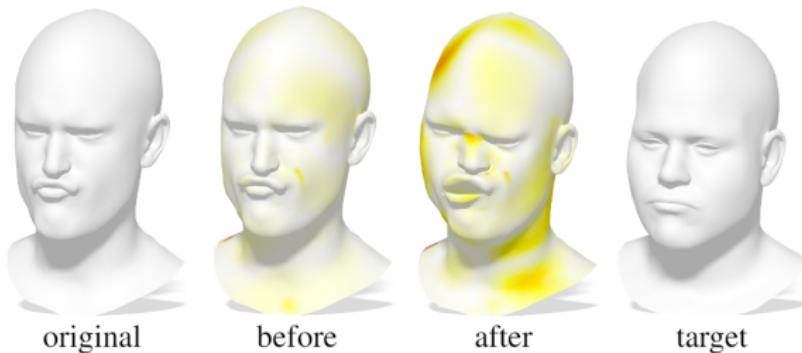
## Example: Targeted attack for adversarial training

Using adversarial examples as **training** data improves the **robustness** of the attacked learning model:



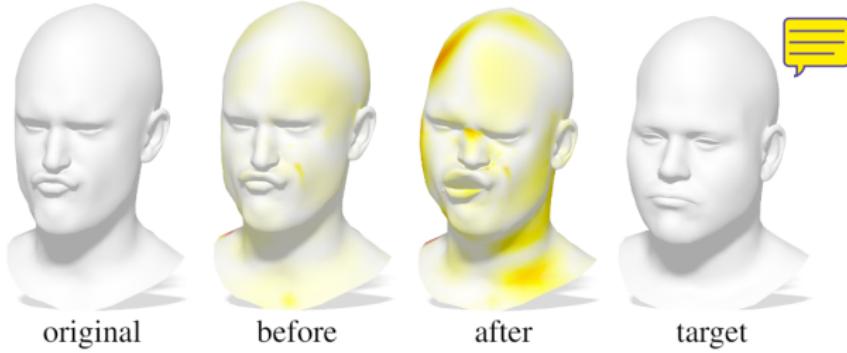
## Example: Targeted attack for adversarial training

Using adversarial examples as **training** data improves the **robustness** of the attacked learning model:



## Example: Targeted attack for adversarial training

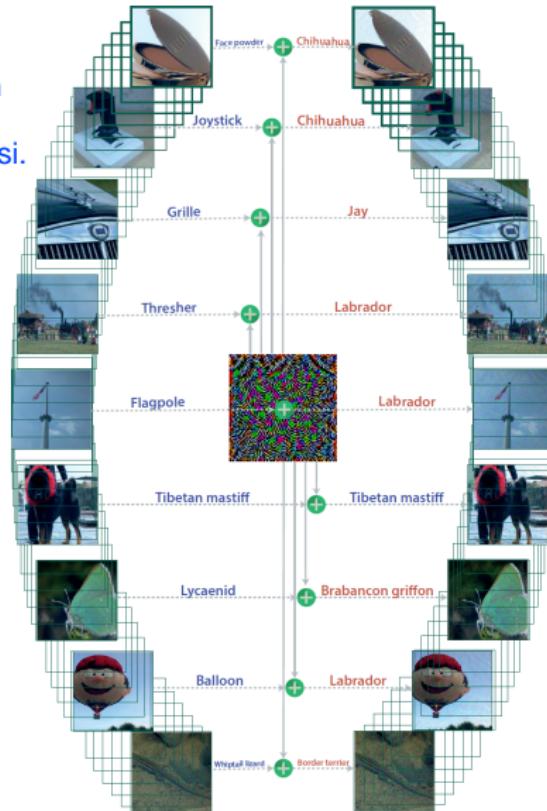
Using adversarial examples as **training** data improves the **robustness** of the attacked learning model:



But however we improve robustness, classifiers are always vulnerable!

# Universal perturbations

la perturbazione mostrata al centro di questa figura è in grado di destabilizzare i classificatori in tutti questi casi. La rete qui utilizzata è molto potente, quindi il pericolo è reale



# Non-Euclidean domains

Adversarial training can also be conducted on **geometric** domains.



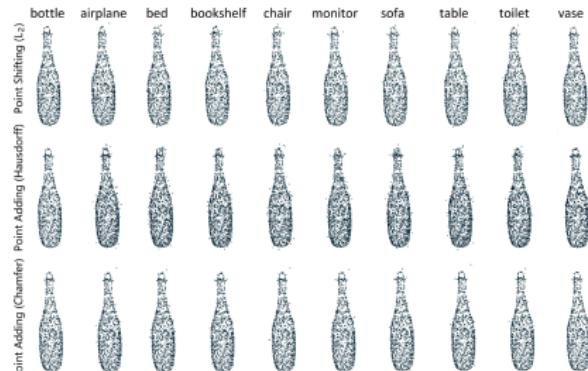
Change website content



Buy likes/  
followers



Unfollow  
untrusted users



Zügner et al, "Adversarial attacks on neural networks for graph data", 2018; Xiang et al, "Generating 3D Adversarial Point Clouds", 2018

# Non-Euclidean domains

Adversarial training can also be conducted on **geometric** domains.



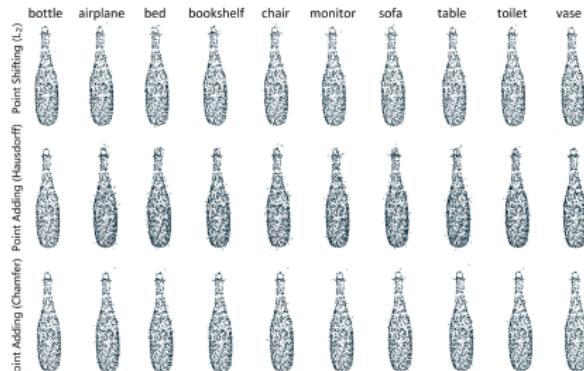
Change website content



Buy likes/  
followers



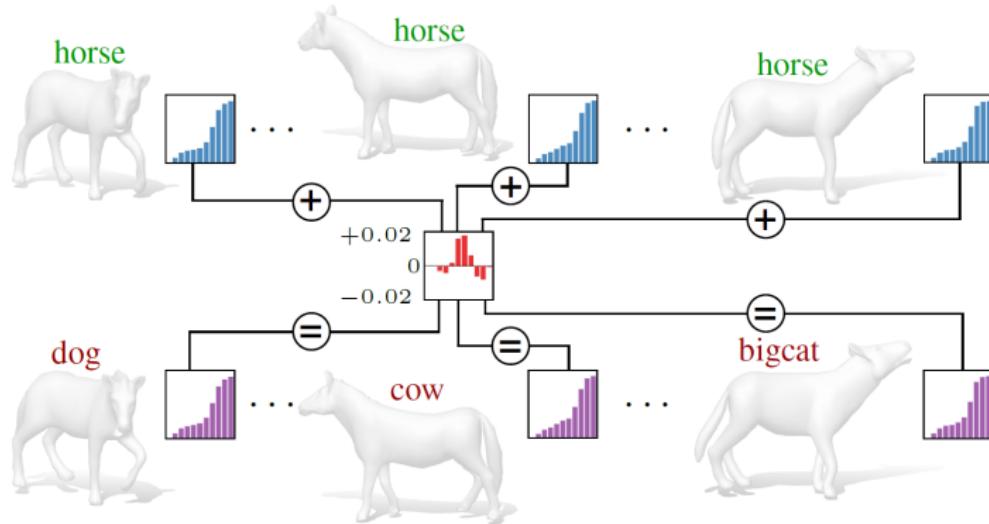
Unfollow  
untrusted users



- The notion of **perceptible** is different than with images.
- Can alter the **domain** (e.g. the graph connections) rather than just the features (e.g. the values stored at the nodes).

Zügner et al, "Adversarial attacks on neural networks for graph data", 2018; Xiang et al, "Generating 3D Adversarial Point Clouds", 2018

# Universal perturbations on 3D data



Rampini, Pestarini, Cosmo, Melzi, Rodolà, "Universal Spectral Adversarial Attacks for Deformable Shapes", 2021