Deep Learning & Applied AI

Adversarial training

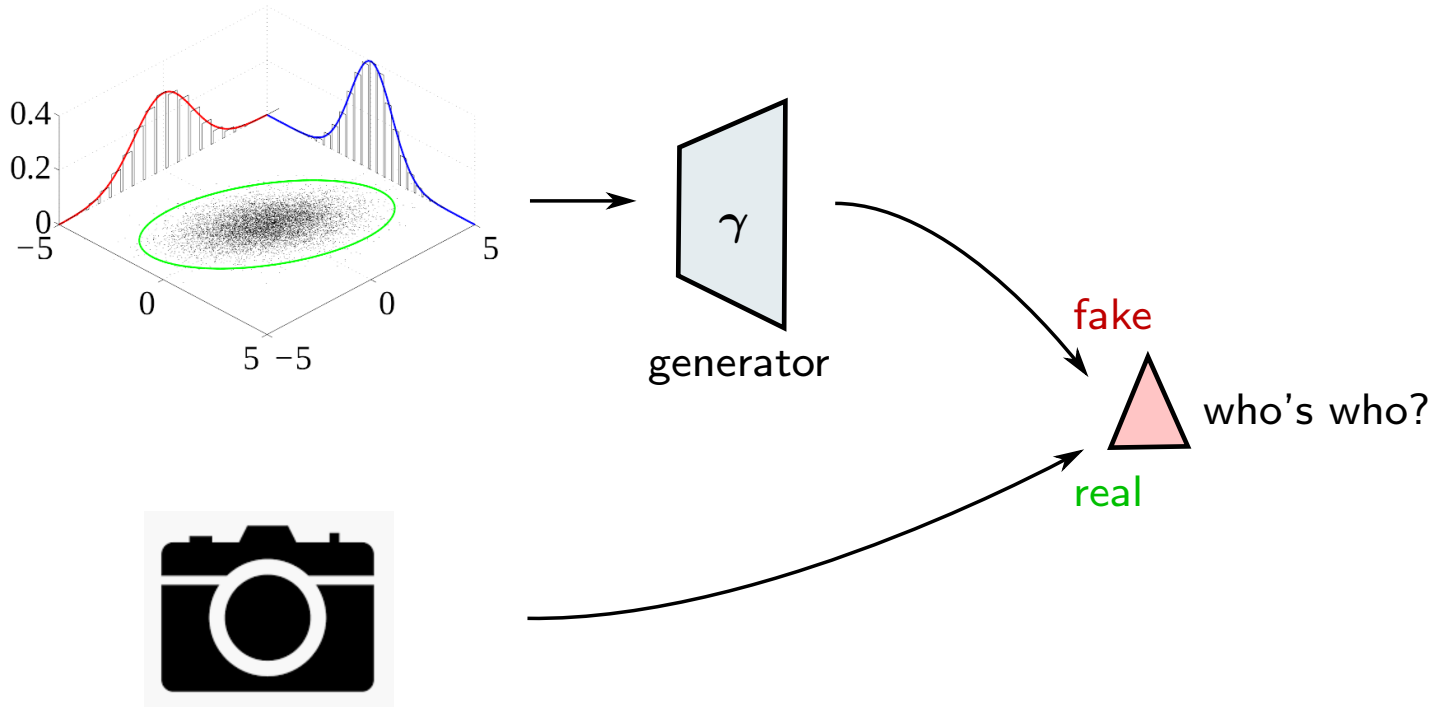
Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2019/2020 · May 18, 2020

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:

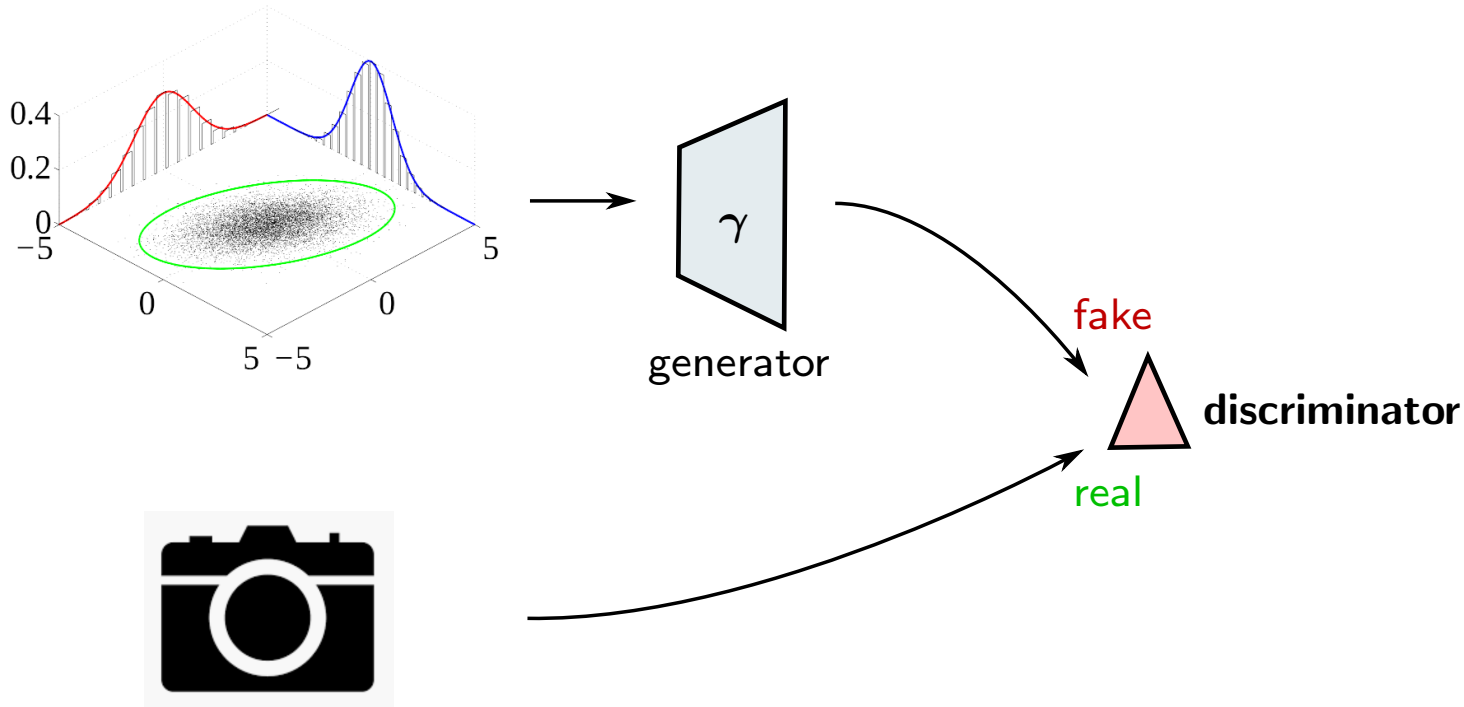


Given instances of fake and real data, a generative model is "good" when you can not distinguish between the two.

Goodfellow et al, "Generative adversarial networks", NIPS 2014

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 Δ with parameters $\boldsymbol{\delta}$, whose output is the probability that the given data is real.

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

$$\underbrace{\mathbb{E}_{\mathbf{x}} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x})}_{\text{real data}}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

$$\mathbb{E}_{\mathbf{x}} \log \underbrace{\Delta_{\boldsymbol{\delta}}(\mathbf{x})}_{\approx 1} + \mathbb{E}_{\mathbf{z}} \log(1 - \underbrace{\Delta_{\boldsymbol{\delta}}(D_{\boldsymbol{\gamma}}(\mathbf{z}))}_{\approx 0})$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Mathematically, we can phrase:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

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

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

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

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

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

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

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

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

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

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

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Let us try to maximize the discriminator score given a generator G:

$$J(G) = \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x} \sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x}))$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Let us try to maximize the discriminator score given a generator G:

$$J(G) = \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x}\sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x}\sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x}))$$

$$= \max_{\boldsymbol{\delta}} \; \int [\log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) p_d(\mathbf{x}) + \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x})) p_g(\mathbf{x})] \, d\mathbf{x}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Let us try to maximize the discriminator score given a generator G:

$$\begin{aligned} J(G) &= \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x}\sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x}\sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x})) \\ &= \max_{\boldsymbol{\delta}} \; \int [\log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) p_d(\mathbf{x}) + \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x})) p_g(\mathbf{x})] \, d\mathbf{x} \end{aligned}$$

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

$$\max_{a} \; p \log a + q \log(1 - a)$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Let us try to maximize the discriminator score given a generator G:

$$J(G) = \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x} \sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x}))$$

$$= \max_{\boldsymbol{\delta}} \int [\log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) p_d(\mathbf{x}) + \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x})) p_g(\mathbf{x})] \, d\mathbf{x}$$

For any given $\mathbf{x}$, we want to maximize $\Delta_{\boldsymbol{\delta}}(\mathbf{x}) = a$; let's rename for simplicity $p_d(\mathbf{x}) \equiv p$ 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:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Assume $\mathbf{x} \sim p_g$ for the generated data, and $\mathbf{x} \sim p_d$ for the real data, where p_g is parametrized by $\boldsymbol{\gamma}$.

Let us try to maximize the discriminator score given a generator G:

$$J(G) = \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x} \sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x} \sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x}))$$

$$= \max_{\boldsymbol{\delta}} \int [\log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) p_d(\mathbf{x}) + \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x})) p_g(\mathbf{x})] \, d\mathbf{x}$$

For any given $\mathbf{x}$, we want to maximize $\Delta_{\boldsymbol{\delta}}(\mathbf{x}) = a$; let's rename for simplicity $p_d(\mathbf{x}) \equiv p$ 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:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

$$\Delta_{\boldsymbol{\delta}}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back in the main functional:

$$J(G) = \max_{\boldsymbol{\delta}} \; \mathbb{E}_{\mathbf{x}\sim p_d} \log \Delta_{\boldsymbol{\delta}}(\mathbf{x}) \; + \; \mathbb{E}_{\mathbf{x}\sim p_g} \log(1 - \Delta_{\boldsymbol{\delta}}(\mathbf{x}))$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back in the main functional:

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back in the main functional:

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

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

$$J(G) \propto \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_d} \log \frac{p_d(\mathbf{x})}{\rho(\mathbf{x})} + \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(\mathbf{x})}{\rho(\mathbf{x})} + \text{const.}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

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

$$\Delta_{\delta}(\mathbf{x}) = \frac{p_d(\mathbf{x})}{p_d(\mathbf{x}) + p_g(\mathbf{x})}$$

Plugging it back in the main functional:

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

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

$$\begin{aligned} J(G) &\propto \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_d} \log \frac{p_d(\mathbf{x})}{\rho(\mathbf{x})} + \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_g} \log \frac{p_g(\mathbf{x})}{\rho(\mathbf{x})} + \text{const.} \\ &= \frac{1}{2}KL(p_d\|\rho) + \frac{1}{2}KL(p_g\|\rho) + \text{const.} \end{aligned}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$J(G) = \frac{1}{2} KL(p_d \| \rho) + \frac{1}{2} KL(p_g \| \rho) + \text{const.}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$\min_{p_g} \ \frac{1}{2} KL(p_d \| \rho) + \frac{1}{2} KL(p_g \| \rho) + \text{const.}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$\min_{p_g} \ \frac{1}{2} KL(p_d \| \rho) + \frac{1}{2} KL(p_g \| \rho)$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$\min_{p_g} \underbrace{\frac{1}{2}KL(p_d\|\rho) + \frac{1}{2}KL(p_g\|\rho)}_{\text{Jensen-Shannon divergence between } p_d \text{ and } p_g}$$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$\min_{p_g} \underbrace{\frac{1}{2}KL(p_d\|\rho) + \frac{1}{2}KL(p_g\|\rho)}_{JS(p_d\|p_g)}$$

Property: $p_d = p_g \;\Leftrightarrow\; JS(p_d\|p_g) = 0$

Goodfellow et al, "Generative adversarial networks", NIPS 2014

Generative adversarial networks (GAN)

Therefore, the optimal GAN generator is found by minimizing:

$$\min_{p_g} \underbrace{\frac{1}{2}KL(p_d\|\rho) + \frac{1}{2}KL(p_g\|\rho)}_{JS(p_d\|p_g)}$$

Property: $p_d = p_g \;\Leftrightarrow\; JS(p_d\|p_g) = 0$

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

Goodfellow et al, "Generative adversarial networks", NIPS 2014

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.

"speed limit 50mph"

Adversarial attacks

Szegedy et al, “Intriguing properties of neural networks”, 2013

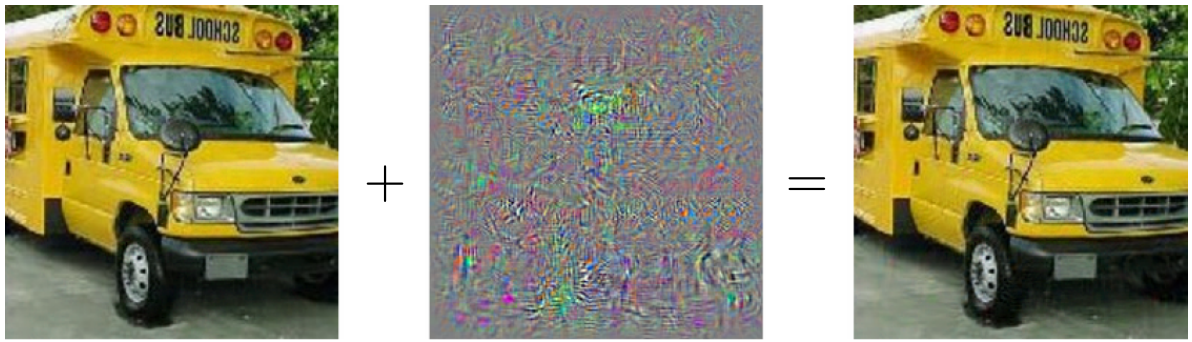
Adversarial attacks

"truck"

"ostrich"

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

Adversarial attacks



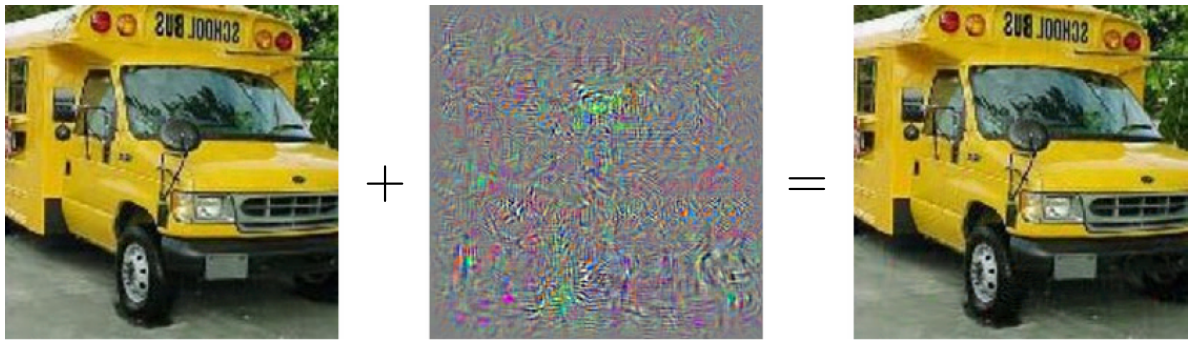
"truck" perturbation "ostrich"

The perturbation can be explicitly optimized for.

Adversarial attacks can cause a system to take unwanted actions.

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

Adversarial attacks



"truck" perturbation "ostrich"

The perturbation can be explicitly optimized for.

Adversarial attacks can cause a system to take unwanted actions.

How to construct undetectable adversarial examples?

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

Perception

An attack is undetectable if it can not be perceived as such.

This requires the definition of a metric or similarity measure s.t.:

Perception

An attack is undetectable if it can not be perceived as such.

This requires the definition of a metric or similarity measure s.t.:

- It captures the noticeability of the attack.

Perception

An attack is undetectable if it can not be perceived as such.

This requires the definition of a metric or similarity measure s.t.:

- It captures the noticeability of the attack.
- It can be minimized to construct undetectable adversarial examples.

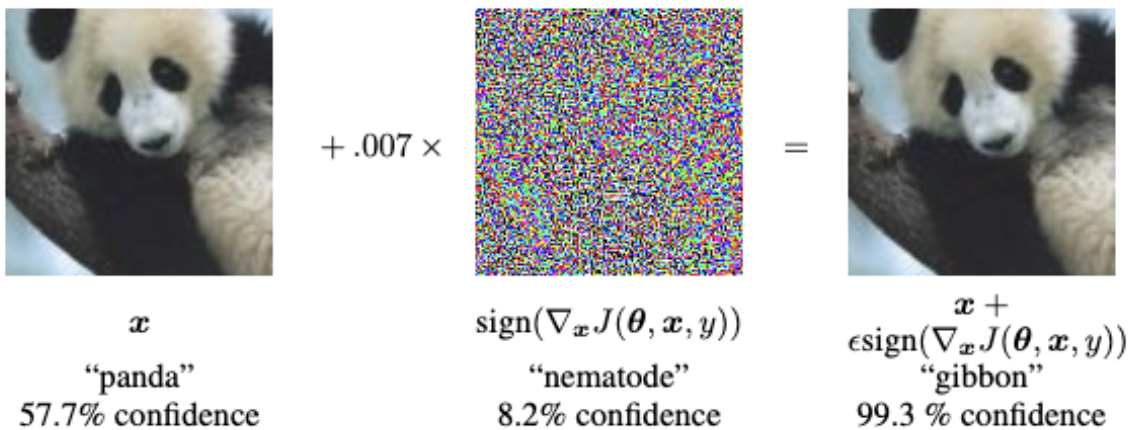
Perception

An attack is undetectable if it can not be perceived as such.

This requires the definition of a metric or similarity measure s.t.:

- It captures the noticeability of the attack.
- It can be minimized to construct undetectable adversarial examples.

The choice depends on the domain and on the task:



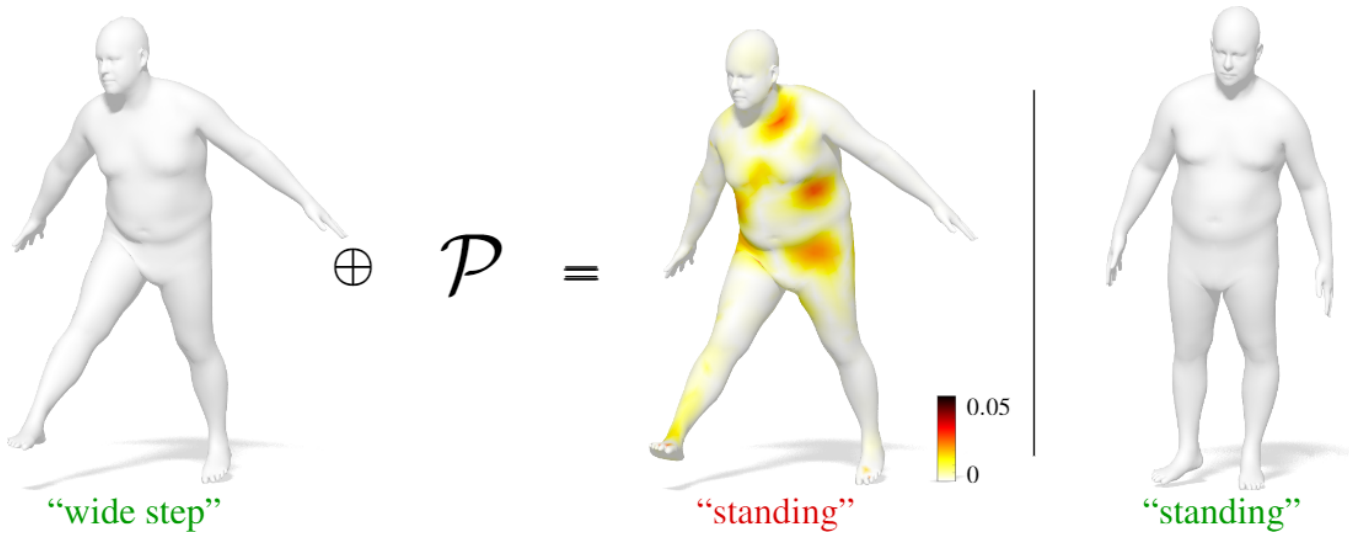
Perception

An attack is undetectable if it can not be perceived as such.

This requires the definition of a metric or similarity measure s.t.:

- It captures the noticeability of the attack.
- It can be minimized to construct undetectable adversarial examples.

The choice depends on the domain and on the task:



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.).

It is possible to train a substitute model with black-box access to the target; by attacking the substitute model (now with white-box access), we can transfer these attacks to the black-box target.

Papernot et al, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples", 2016

Targeted attacks

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

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

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

Targeted attacks

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

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

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.

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

Targeted attacks

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

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

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.

$c > 0$ is chosen as the one giving an example of minimum distance, can be found via line search algorithms.

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

Targeted attacks

A more general approach is given by:

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

where the perturbation $\boldsymbol{\delta}$ appears explicitly, and d is some distance function that depends on the specific task.

Carlini and Wagner, "Towards Evaluating the Robustness of Neural Networks", 2016

Targeted attacks

A more general approach is given by:

$$\min_{\boldsymbol{\delta} \in [0,1]^n} \; d(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta})$$
$$\text{s.t. } f(\mathbf{x} + \boldsymbol{\delta}) \leq 0$$

where the perturbation $\boldsymbol{\delta}$ appears explicitly, and d is some distance function that depends on the specific task.

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

Carlini and Wagner, "Towards Evaluating the Robustness of Neural Networks", 2016

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 some distance function that depends on the specific task.

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

Carlini and Wagner, "Towards Evaluating the Robustness of Neural Networks", 2016

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 some distance function that depends on the specific task.

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

Possible instance of the problem above:

$$\min_{\boldsymbol{\delta} \in [0,1]^n} \|\boldsymbol{\delta}\|_p + c\, \left(\max\{F(\mathbf{x} + \boldsymbol{\delta})_i \ : \ i \neq t\} - F(\mathbf{x} + \boldsymbol{\delta})_t\right)^+$$

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 $(a)^+ = \max(a, 0)$.

Carlini and Wagner, "Towards Evaluating the Robustness of Neural Networks", 2016

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 ℓ_{gt}:

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

which adds a perturbation maximizing the cost.

Kurakin et al, "Adversarial examples in the physical world", 2016

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 ℓ_{gt}:

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

which adds a perturbation maximizing the cost.

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

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

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

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

Kurakin et al, "Adversarial examples in the physical world", 2016

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 ℓ_{gt}:

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

which adds a perturbation maximizing the cost.

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

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

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

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

This method is designed to be fast, since 1 iteration ≈ 1 backprop step.

Kurakin et al, "Adversarial examples in the physical world", 2016

Example: Targeted attack for adversarial training

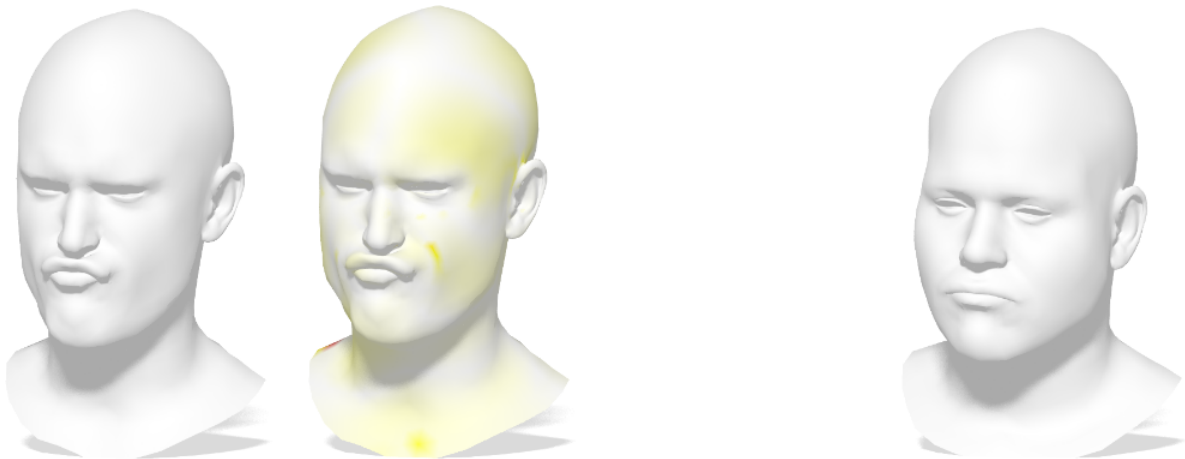
Using adversarial examples as training data improves the robustness of the attacked learning model:

original

target

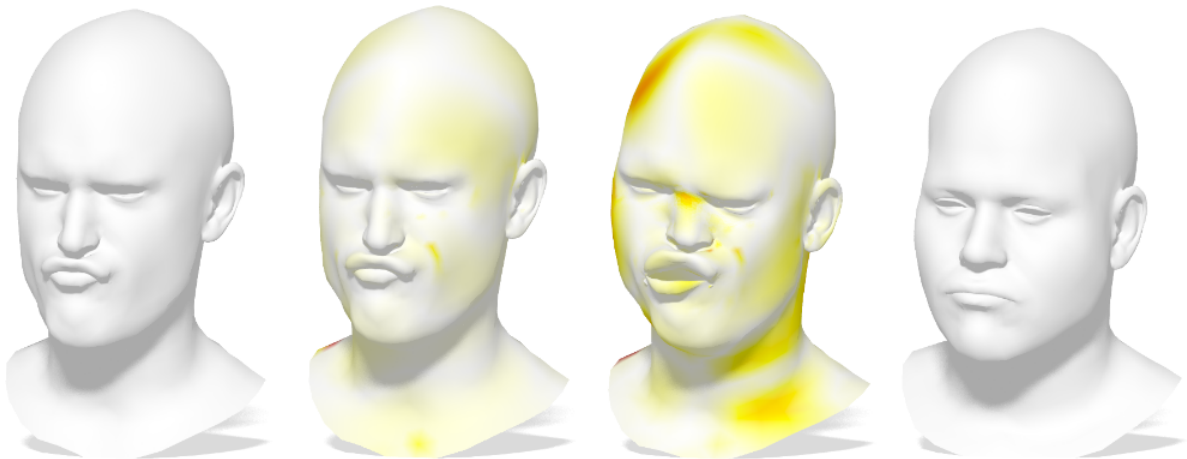
Example: Targeted attack for adversarial training

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



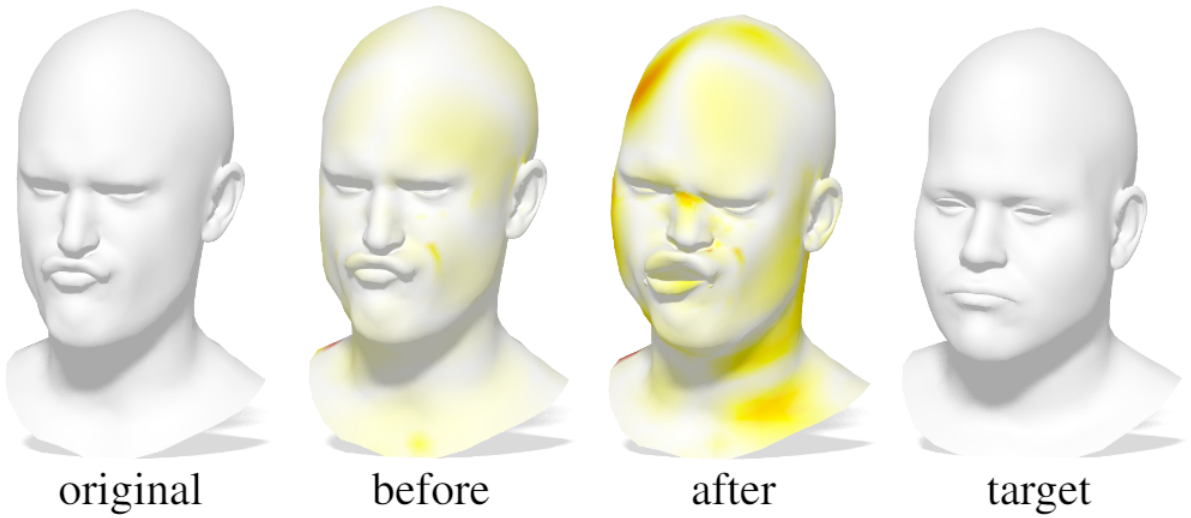
Example: Targeted attack for adversarial training

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!

Vulnerability

There exist small adversarial perturbations for arbitrary classifiers.

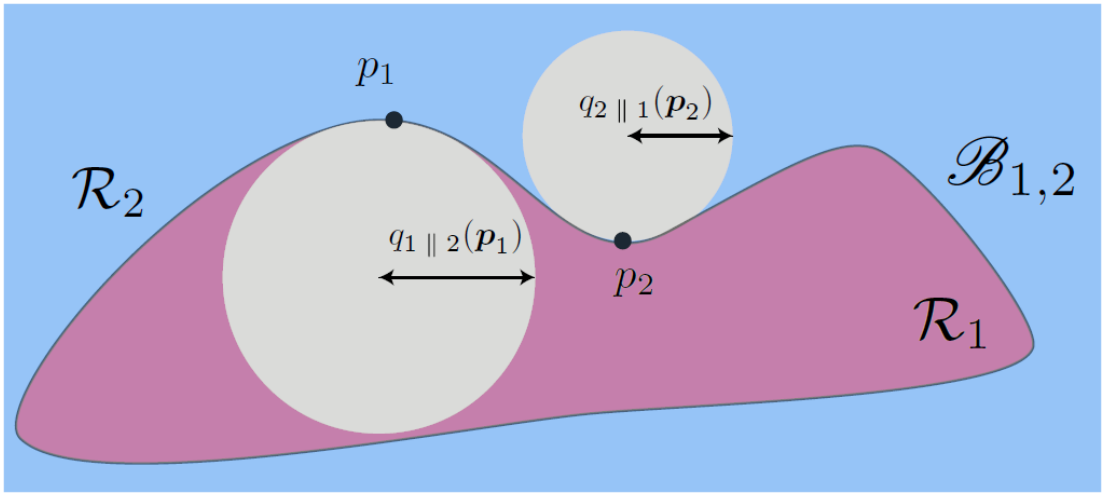
There is a maximal achievable robustness.

Fawzi et al, "Robustness of classifiers: from adversarial to random noise", 2016; Fawzi et al, "Adversarial vulnerability for any classifier", 2018

Vulnerability

There exist small adversarial perturbations for arbitrary classifiers.

There is a maximal achievable robustness.

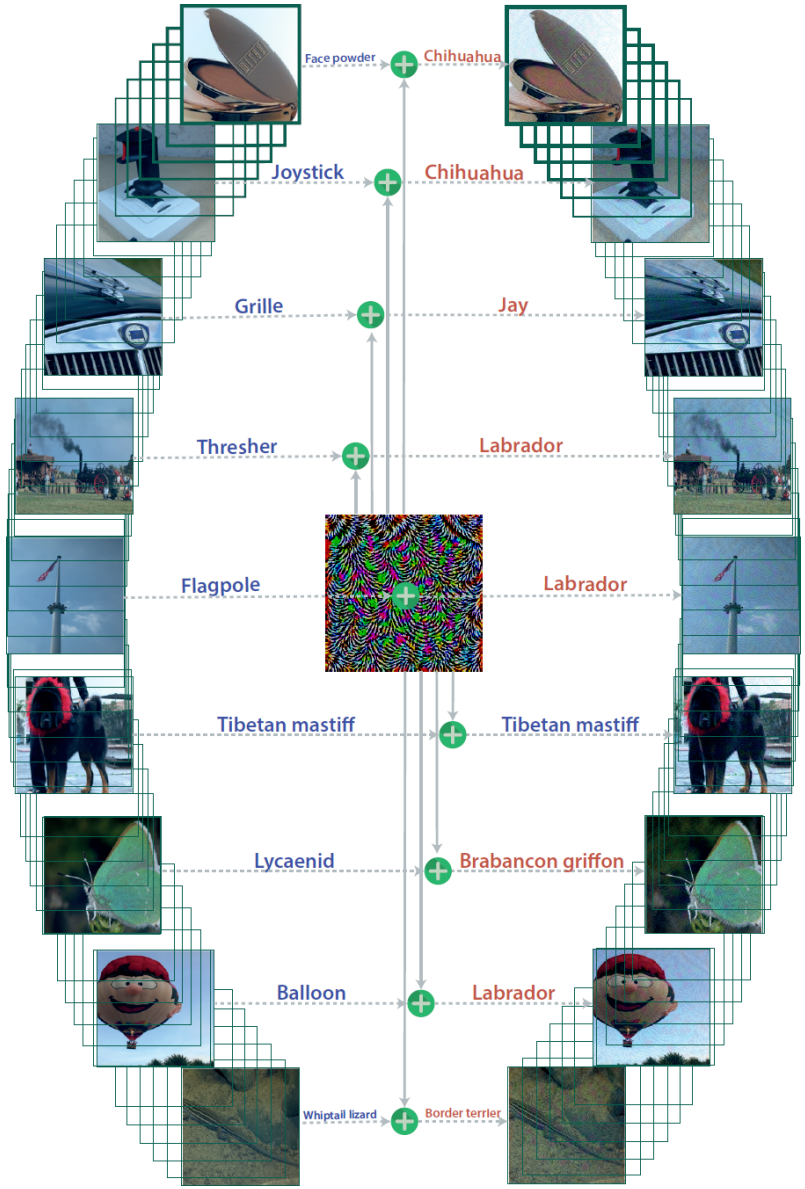


Key observation: Data points often lie very close to the decision boundary of the classifier.

Robustness can be quantified w.r.t. the curvature of the boundary.

Fawzi et al, "Robustness of classifiers: from adversarial to random noise", 2016; Fawzi et al, "Adversarial vulnerability for any classifier", 2018

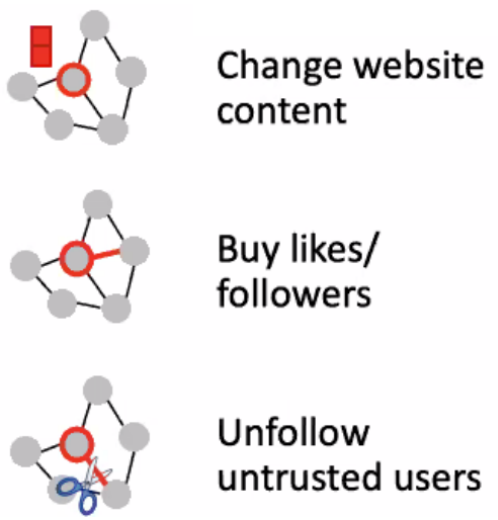
Universal perturbations

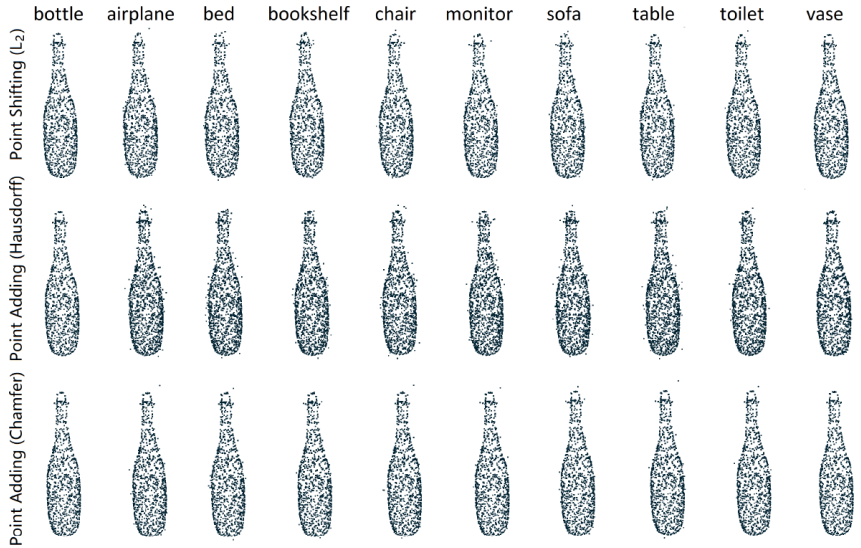


Moosavi-Dezfooli et al, "Universal adversarial perturbations", 2017

Non-Euclidean domains

Adversarial training can also be conducted on geometric domains.

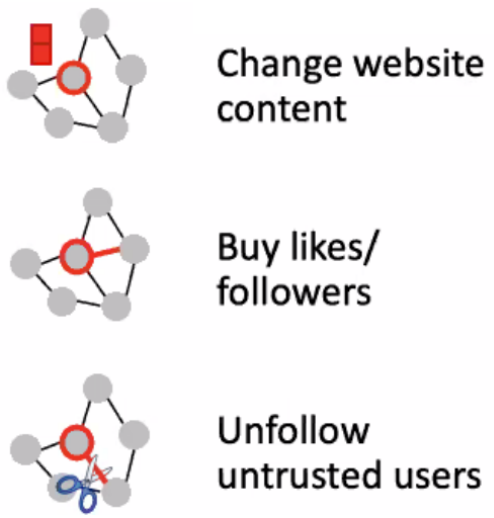


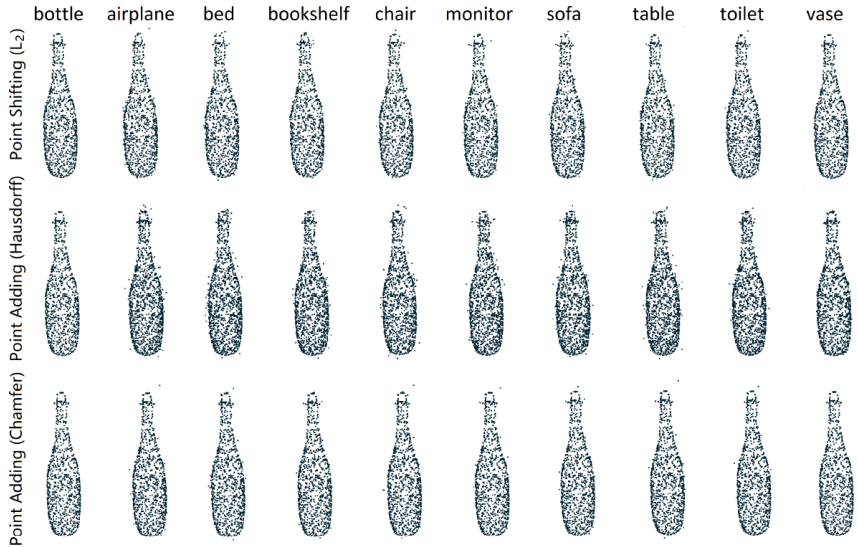


Zügner et al, "Adversarial attacks on neural networks for graph data", 2018; Xiang et al, "Generating 3D Adversarial Point Clouds", 2018

Non-Euclidean domains

Adversarial training can also be conducted on geometric domains.





- 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