Deep Learning (BEV033DLE)
Lecture 9 Adversarial examples & robust learning

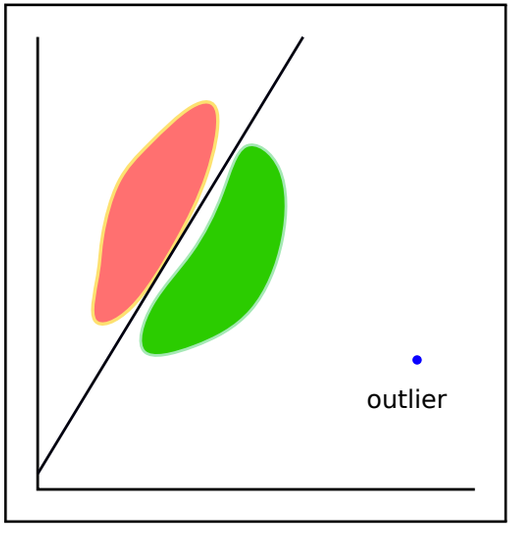
Czech Technical University in Prague

- Adversarial examples
- Adversarial attacks
- Robust learning

Adversarial examples

A purely discriminatively learned predictor has by itself no notion of the underlying data distribution.

- Consider a linear classifier shown to the right. The outlier point is classified in the same way as inlier points.
- Consider an age predictor trained on face images. What happens if it is presented an image of an amoeba?

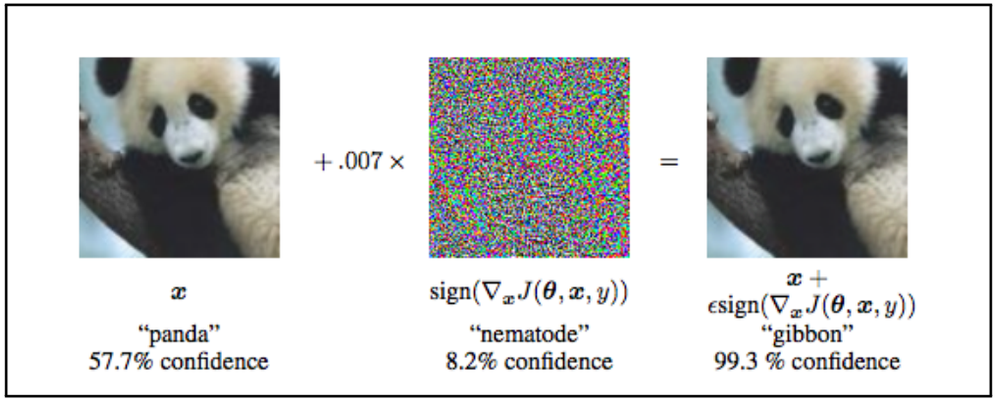


However, we expect that a deep network predictor trained to classify images with high accuracy, will predict correct classes for distorted images, provided that the distortions are visually imperceptible.

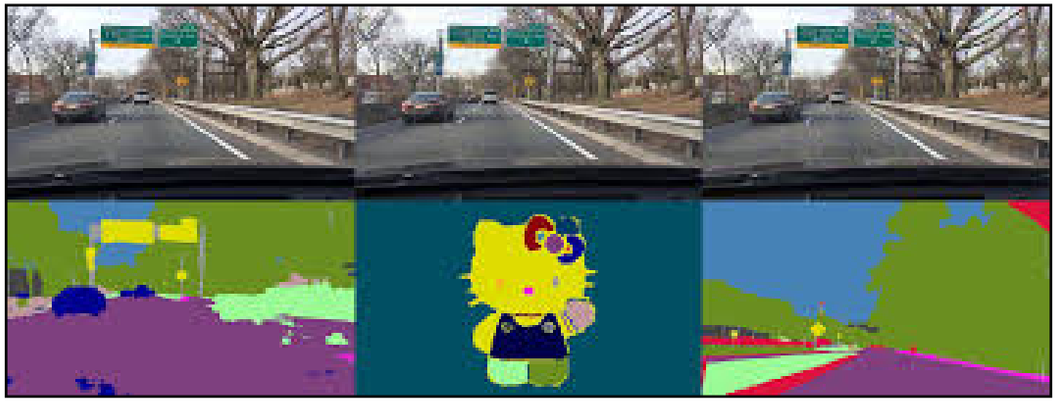
Unfortunately, this is not true!

Adversarial examples

Given a clean image x, compute the gradient of the loss w.r.t. x and add a small, imperceptible distortion in this direction



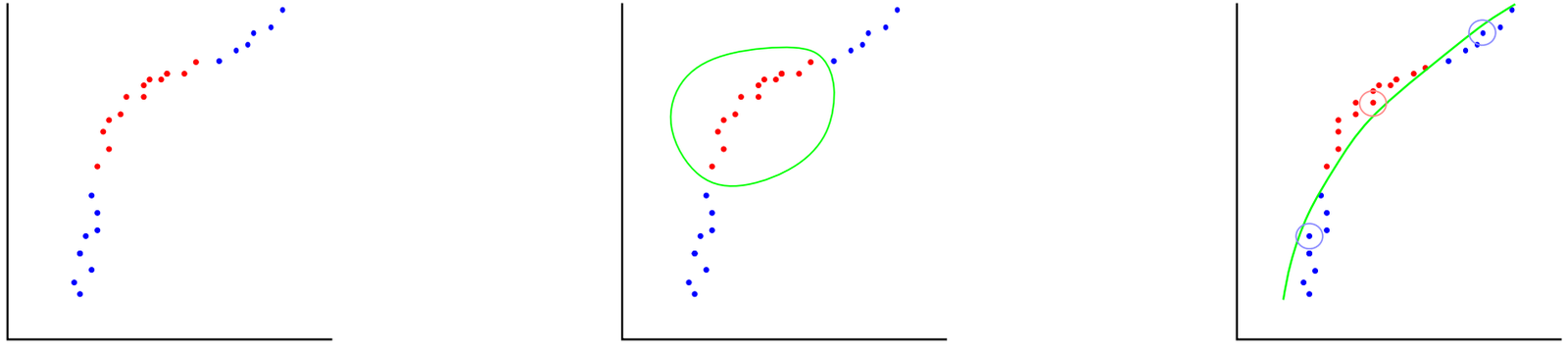
Given a clean image x and a "nonsense" target, find the smallest distortion of x such that the network will predict this target



Adding targeted but imperceptible distortions fools networks completely. Unfortunately, this holds across network architectures, training sets and tasks.

Adversarial examples

What are adversarial examples? Simplified illustration:



Left to right: training data, classifiers with different susceptibility to adversarial examples

---

Side step: Gheiros et al., ICLR 2019, CNNs trained on ImageNet are strongly biased towards recognising textures rather than shapes.

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Adversarial attacks

(Szegedy et al. 2013) Consider the context of classification networks and denote by $\ell(x, y)$ the network loss for predicting class y for the input x. E.g.

$$\ell(x, y) = -\log p(y \mid x) = -a_y(x) + \log \sum_k e^{a_k(x)},$$

where $a(x)$ denotes the activations of the last linear layer of the network.

Fast gradient sign attack: (FGSM)

Compute the gradient of the loss for the true class y_{true} and distort the input by

$$\tilde{x} = x + \varepsilon \operatorname{sign}\big(\nabla_x \ell(x, y_{true})\big)$$

with some small ε. Iterative variant of FGSM

$$\begin{aligned} x'_t &= x_{t-1} + \alpha \operatorname{sign}\big(\nabla_x \ell(x_{t-1}, y_{true})\big) \\ x_t &= P(x'_t), \end{aligned}$$

where P projects x into a specified domain, e.g. $[0, 1]^n$.

Adversarial attacks

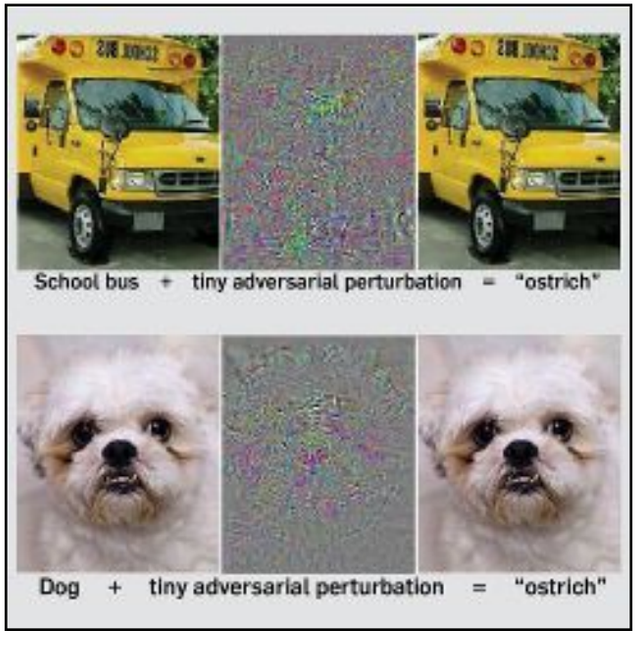
Targeted attack:

Given x and the true class y_{true}, choose a target class, e.g. $y_{targ} = \arg\min_y p(y|x)$ and set

$$\tilde{x} = x - \varepsilon \operatorname{sign}\big(\nabla_x \ell(x, y_{targ})\big)$$

Iterative variant

$$x'_t = x_{t-1} - \alpha \operatorname{sign}\big(\nabla_x \ell(x_{t-1}, y_{targ})\big)$$
$$x_t = P(x'_t)$$



Adversarial attacks

General scheme for untargeted adversarial attacks:

Fix a norm $\|x\|$, $x \in \mathbb{R}^n$ and the ball $B_\varepsilon(x) = \left\{x' \in \mathbb{R}^n \;\middle|\; \|x' - x\| \leqslant \varepsilon\right\}$

For a given example (x, y_{true}) solve

$$x_* \in \underset{x' \in B_\varepsilon(x)}{\arg\max}\, \ell(x', y_{true})$$

Example 1. Let us consider the infinity norm $\|x\| = \max_i |x_i|$ and let us furthermore assume that $\ell(x', y_{true})$ can be approximated by a linear function for $x' \in B_\varepsilon(x)$. Then the task reads

$$x_* \in \underset{x' \in B_\varepsilon(x)}{\arg\max} \left[\ell(x, y_{true}) + (x' - x)^T \nabla_x \ell(x, y_{true})\right]$$

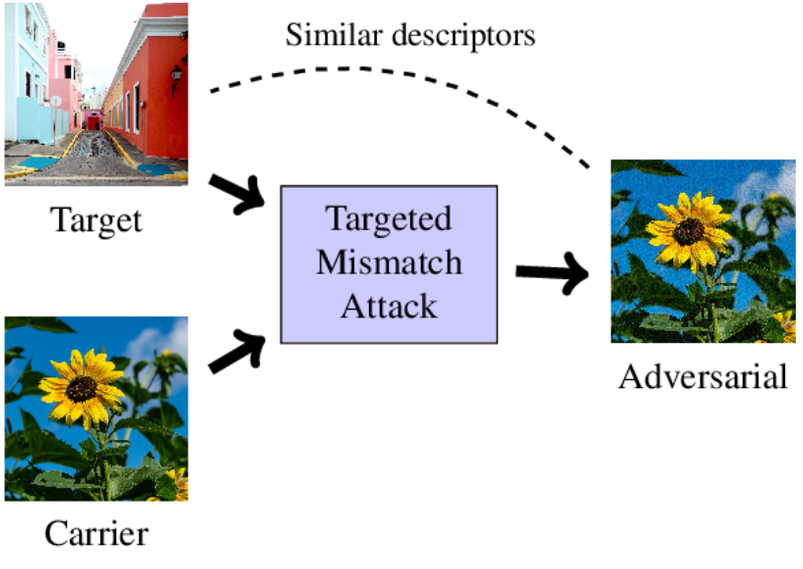
Its solution is given by $x = \varepsilon \operatorname{sign}(\nabla_x \ell(x, y_{true}))$, i.e. the FGSM attack.

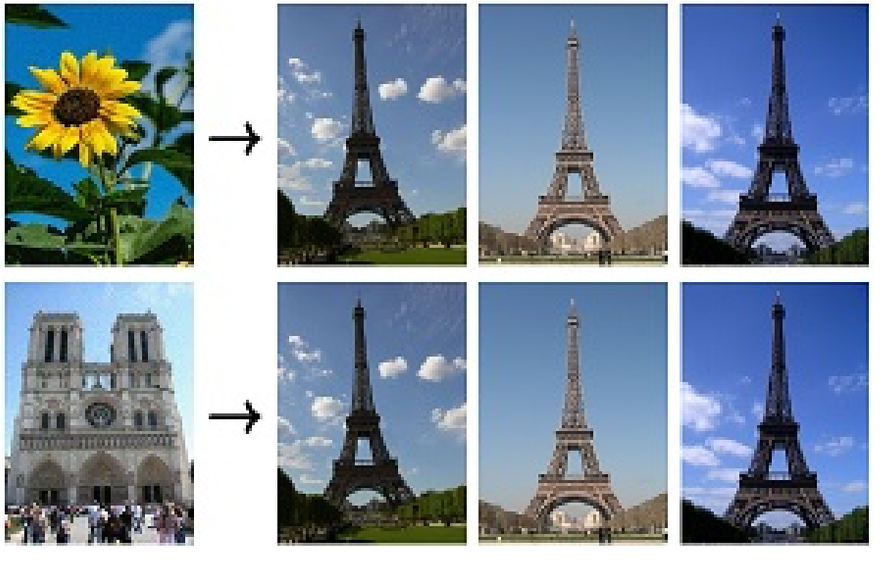
Adversarial attacks

Meanwhile there exists an array of different adversarial attacks with “dimensions”:

- targeted, untargeted
- access to architecture + weights (white box), architecture (grey box), oracle (black box)
- gradient based, score based, decision based

Adversarial attacks can take quite creative and strange forms: Query a database without revealing your query image: Tolias et al., ICCV 2019





Adversarially robust learning

(Kurakin, et al. 2017) Regularise loss by adversarial terms.

$$L(\hat{B}) = \frac{1}{(m-k)+\lambda k}\Big[\sum_{i\in B_c} \ell(x_i, y_i) + \lambda \sum_{j\in B_a} \ell(x_i^{adv}, y_i)\Big]$$

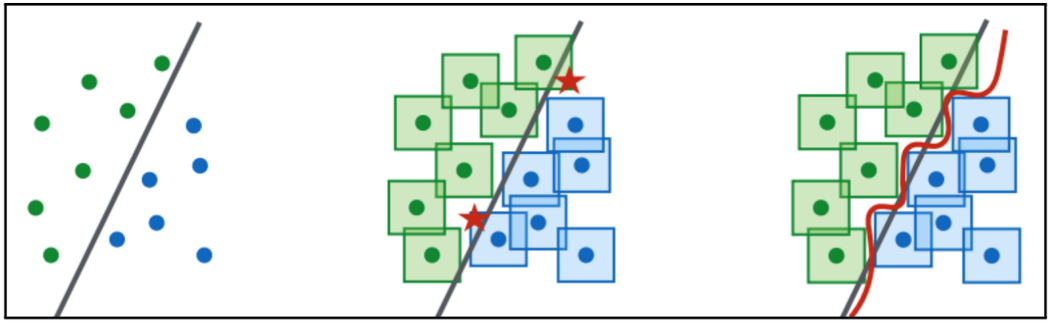
(1) read a mini-batch $B_c = \big\{(x_1, y_1), \ldots, (x_m, y_m)\big\}$,

(2) generate k adversarial examples $B_a = \big\{(x_1^{adv}, y_1), \ldots, (x_k^{adv}, y_k)\big\}$ from k randomly chosen clean examples,

(3) compose a new mini-batch $\hat{B} = B_a \cup B_c$ and do one training step

- improves robustness against one-step attacks,
- less successful w.r.t. iterative attacks,
- "label leaking effect": accuracy on adversarial examples can become higher than accuracy on clean examples.

Adversarially robust learning



(Madry et al. ICLR 2018) A more principled approach: augment ERM in an universal way.

Let B_ε denote the l_∞ ball with radius ε centered at 0. Consider the following learning task

$$R(w) = \mathbb{E}_{x,y\sim D}\left[\max_{\delta\in B_\varepsilon} \ell(w, x+\delta, y)\right] \to \min_w$$

This is a minimax task.

Analysis the inner maximisation task:

- has many equally good maxima,
- can be solved by projected gradient ascent w.r.t. δ
- maximum reached after moderate number of iterations.

Adversarially robust learning

$$R(w) = \mathbb{E}_{x,y\sim D}\left[\max_{\delta\in B_\varepsilon} \ell(w, x+\delta, y)\right] \to \min_w$$

How to minimise w.r.t. the model parameters w?

Convex analysis: What are descent directions for a function $f(w)$ defined by

$$f(w) = \max_i g_i(w)$$

at a point w_0? If g-s are convex and differentiable:

(1) denote by $I(w_0)$ the set of functions g_i that are "active" in w_0, i.e. $g_i(w_0) = f(w_0)$.

(2) Any of the negative gradients $-\nabla g_i(w_0)$, $i \in I(w_0)$ is a descent direction of $f(w)$ at w_0, provided that the vector 0 is not in their convex hull.

All together a training step reads

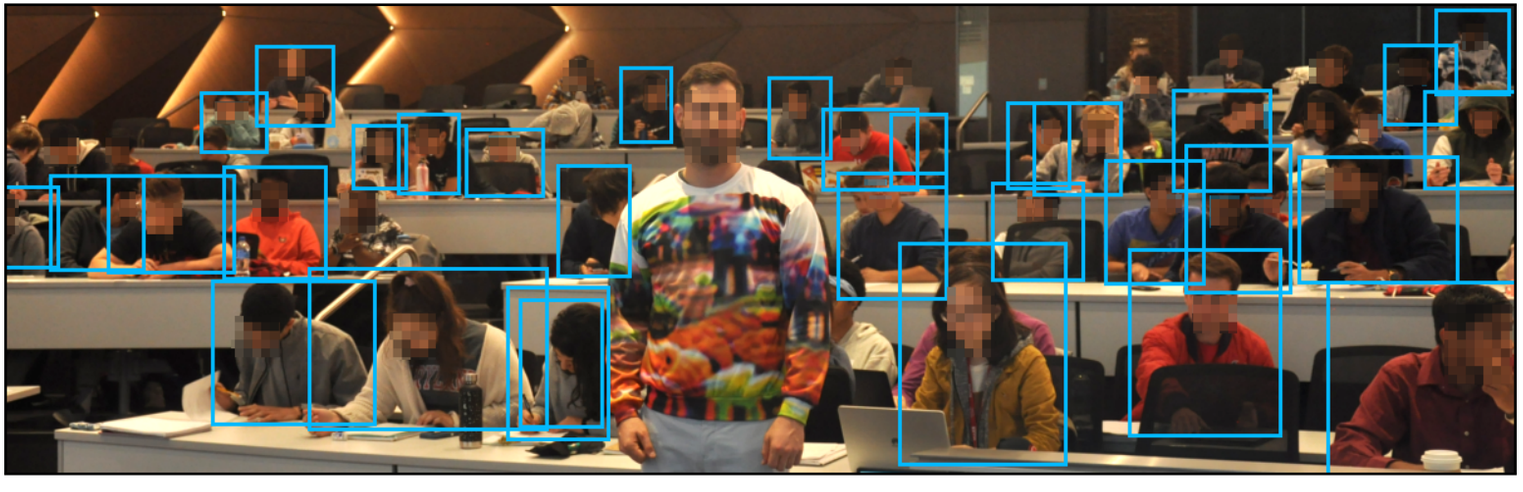
- read a mini-batch $B = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- solve the inner maximisation task for each example x_i and replace it by $x_i^{adv} = x_i + \delta_i$
- do a subgradient step for the modified mini-batch $B_a = \{(x_1^{adv}, y_1), \ldots, (x_m^{adv}, y_m)\}$

Adversarially robust learning

Adversarially robust learning is an ongoing research area. Interesting directions to follow:

- Maximum margin learning approaches e.g. Elsayed et al., NeurIPS 2018), Ding et al., ICLR 2020 try to generalise max-margin approaches from SVM to Deep networks,
- Stochastic neural networks with entropy regularisers or variants of Bayesian inference,

(Wu et al., 2019):



"This stylish pullover is a great way to stay warm this winter, whether in the office or on-the-go. It features a stay-dry microfleece lining, a modern fit, and adversarial patterns the evade most common object detectors"