
# Evasion Attacks / Adversarial Examples

Battista Biggio

Department of Electrical and Electronic Engineering
University of Cagliari, Italy

# Attacks against Machine Learning

| Attacker's Capability | Attacker's Goal: Misclassifications that do not compromise normal system operation — **Integrity** | Attacker's Goal: Misclassifications that compromise normal system operation — **Availability** | Attacker's Goal: Querying strategies that reveal confidential information on the learning model or its users — **Privacy / Confidentiality** |
|---|---|---|---|
| **Test data** | **Evasion (a.k.a. adversarial examples)** | Sponge Attacks | Model extraction / stealing<br>Model inversion (hill climbing)<br>Membership inference |
| **Training data** | Backdoor/targeted poisoning (to allow subsequent intrusions) – e.g., backdoors or neural trojans | Indiscriminate (DoS) poisoning (to maximize test error)<br><br>Sponge Poisoning | - |

**Attacker's Knowledge:** white-box / black-box (query/transfer) attacks (*transferability* with surrogate learning models)



Biggio and Roli, *Wild Patterns*, Patt. Rec. 2018, Best paper award and PR medal 2021

# Evasion Attacks against Machine Learning at Test Time

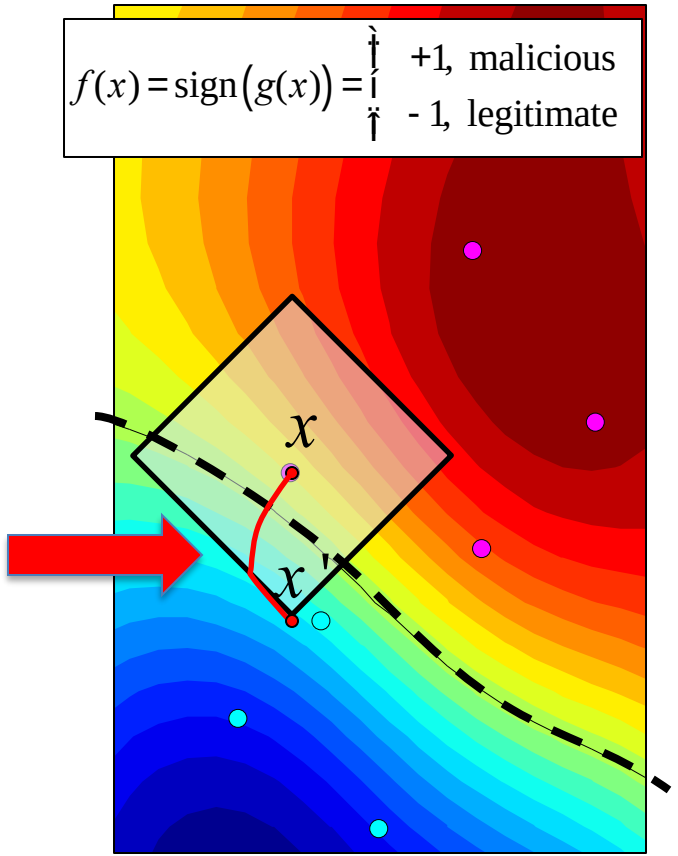
**Biggio, Corona, Maiorca, Nelson, Srndic, Laskov, Giacinto, Roli, ECML-PKDD 2013**

- **Goal:** maximum-confidence *evasion*
- **Knowledge:** *perfect (white-box attack)*
- **Attack strategy:**

$$\min_{x'} g(x')$$

$$\text{s.t.} \|x - x'\|_p \leq d_{\max}$$

- Non-linear, constrained optimization
  - **Projected gradient descent**: approximate solution for *smooth* functions
- Gradients of g(x) can be analytically computed in many cases
  - SVMs, Neural networks



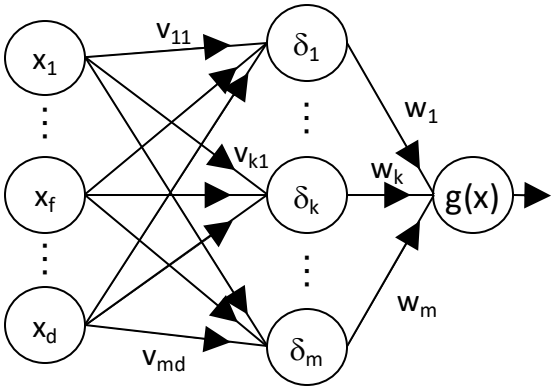
# Computing Descent Directions

## Support vector machines

$$g(x) = \sum_i a_i y_i k(x, x_i) + b, \quad \nabla g(x) = \sum_i a_i y_i \nabla k(x, x_i)$$

**RBF kernel gradient:** $\nabla k(x, x_i) = -2\gamma \exp\left\{-\gamma \| x - x_i \|^2\right\}(x - x_i)$

## Neural networks

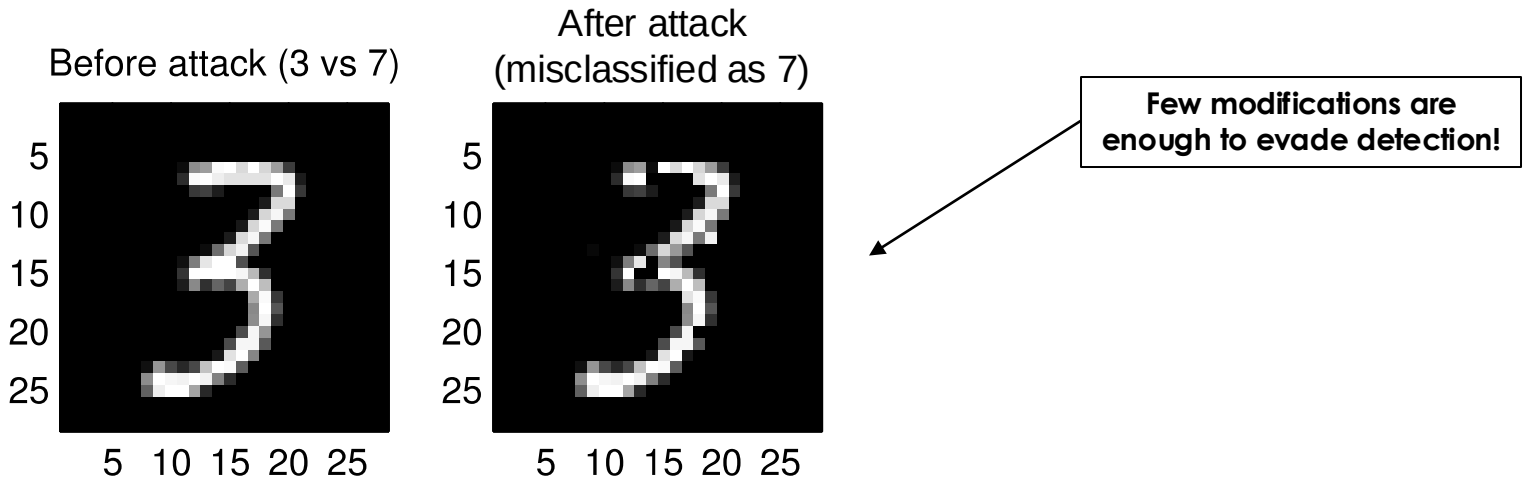



$$g(x) = \left[1 + \exp\left(-\sum_{k=1}^{m} w_k \delta_k(x)\right)\right]^{-1}$$

$$\frac{\partial g(x)}{\partial x_f} = g(x)\left(1 - g(x)\right)\sum_{k=1}^{m} w_k \delta_k(x)\left(1 - \delta_k(x)\right) v_{kf}$$

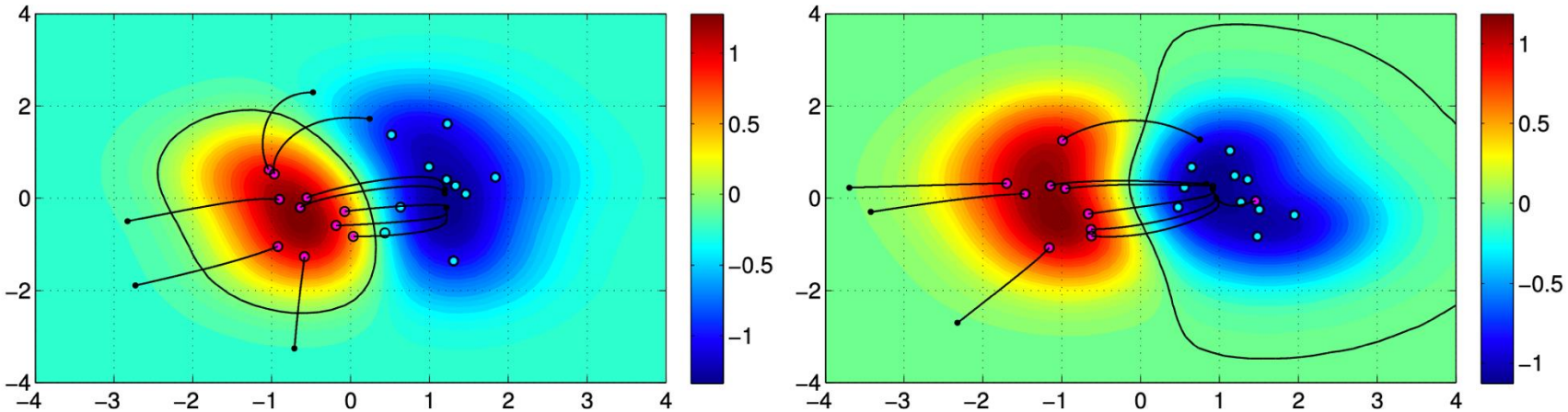
# An Example on Handwritten Digits

- Nonlinear SVM (RBF kernel) to discriminate between '3' and '7'
- **Features**: gray-level pixel values (28 x 28 image = 784 features)



# Problem: Do We Always Evade with Gradient Descent?

- No! Look at the rightmost plot:
  - Many red samples do not cross the boundary …
  - … even if they are able to get sufficiently far from the red class



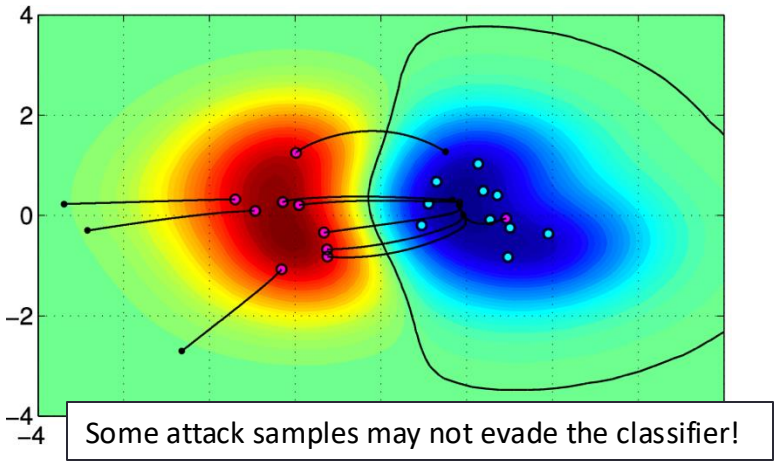



Biggio et al., Evasion Attacks Against Machine Learning at Test Time? ECML 2013

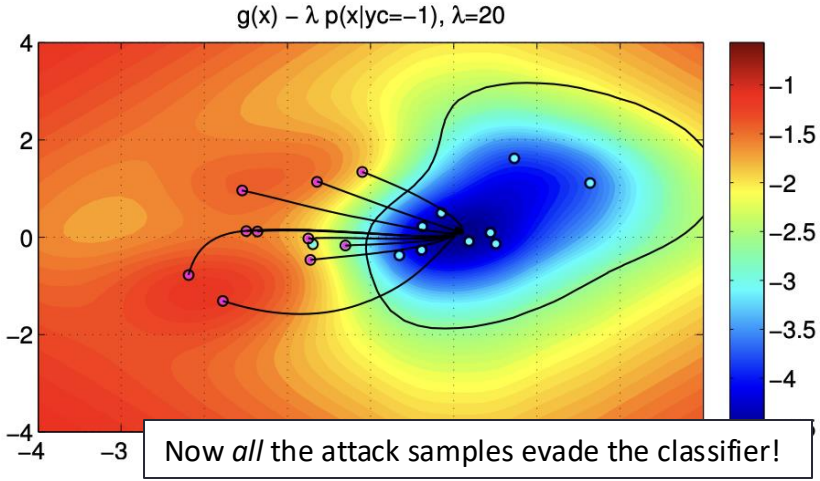
# Solution 1: Kernel Density Estimation (KDE)

- Add KDE of the target class $y_t$ to the objective function
  - to attract the attack samples towards the target class
  - Trade-off parameter $\lambda$: *evasion rate* vs *perturbation size*

$$\min_{x'} g(x') - \lambda p(x'|y_t)$$
$$\text{s.t.} \|x - x'\|_p \leq d_{\max}$$



Some attack samples may not evade the classifier!




Now *all* the attack samples evade the classifier!

# Gradient Descent with KDE (GD-KDE)

**Input:** $\mathbf{x}^0$, the initial attack point; $t$, the step size; $\lambda$, the trade-off parameter; $\epsilon > 0$ a small constant.
**Output:** $\mathbf{x}^*$, the final attack point.

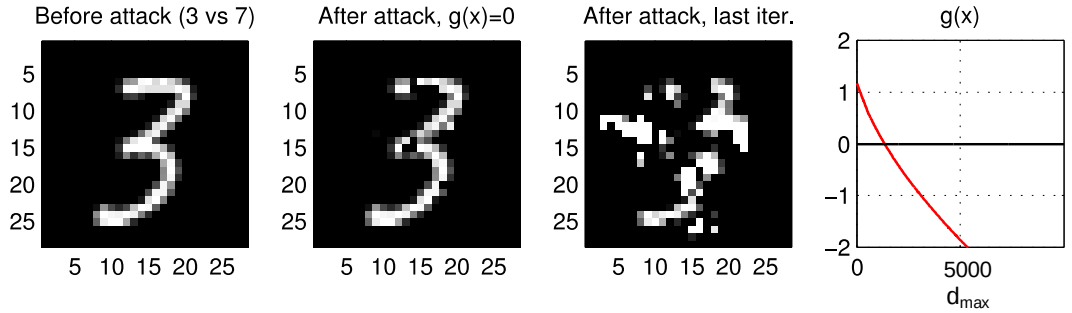
$m \leftarrow 0$.
**repeat**
  $m \leftarrow m + 1$
  Set $\nabla F(\mathbf{x}^{m-1})$ to a unit vector aligned with $\nabla g(\mathbf{x}^{m-1}) - \lambda \nabla p(\mathbf{x}^{m-1} | y^c = -1)$.
  $\mathbf{x}^m \leftarrow \mathbf{x}^{m-1} - t \nabla F(\mathbf{x}^{m-1})$
  **if** $d(\mathbf{x}^m, \mathbf{x}^0) > d_{\max}$ **then**
    Project $\mathbf{x}^m$ onto the boundary of the feasible region.
  **end if**
**until** $F(\mathbf{x}^m) - F(\mathbf{x}^{m-1}) < \epsilon$
**return:** $\mathbf{x}^* = \mathbf{x}^m$

**KDE gradient (RBF kernel):**

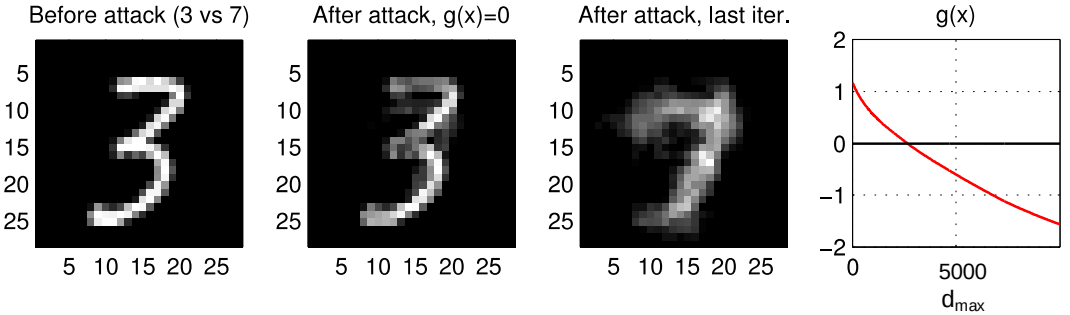
$$\nabla p(x \mid y^c = -1) = -\frac{2}{nh} \sum_{i \mid y_i^c = -1} \exp\left(-\frac{\| x - x_i \|^2}{h}\right)(x - x_i)$$

# Example on Handwritten Digits

**Without mimicry**
**λ = 0**

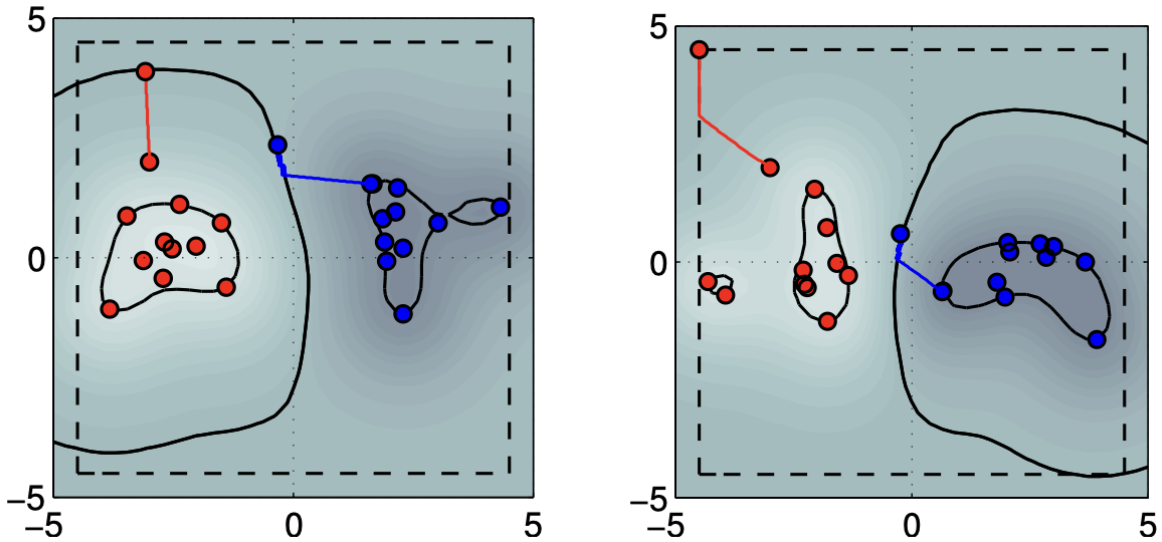



**With mimicry**
**λ = 10**



# Solution 2: Adversarial Initialization

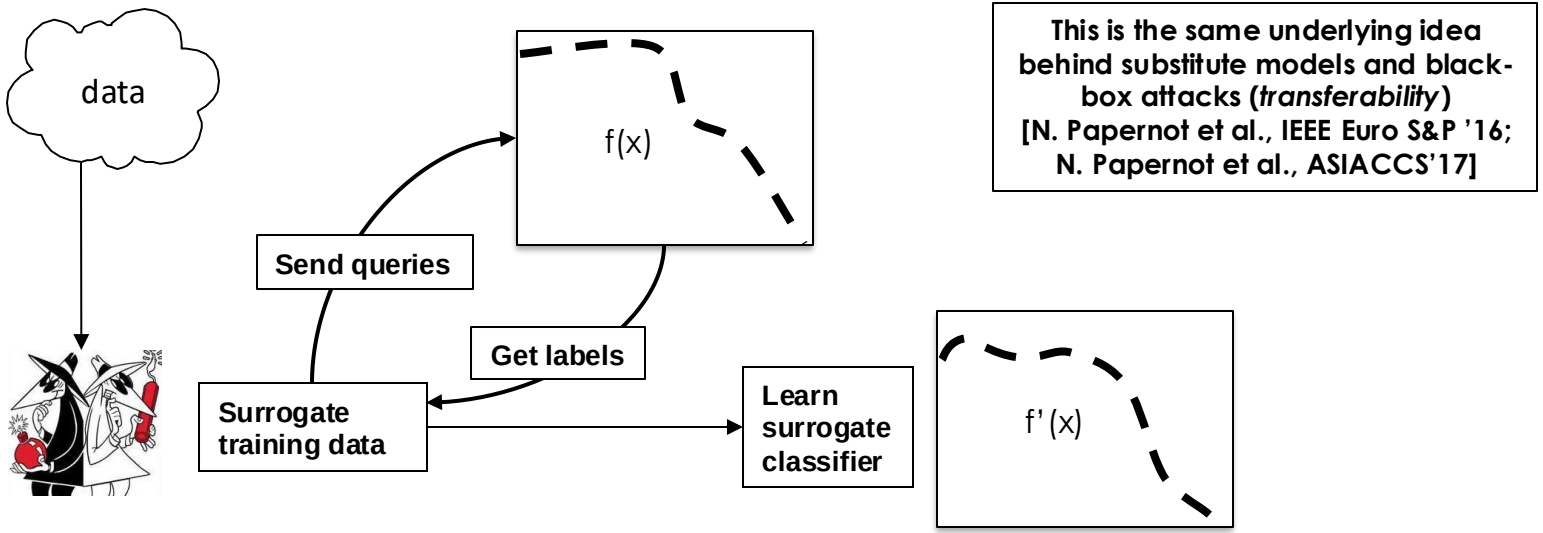
- We do not actually need density estimation
  - **Smarter idea:** to initialize the attack from a point in the target class!



# From White-box to Black-box Attacks

# Bounding the Adversary's Knowledge

- Only feature representation and (possibly) learning algorithm are known
- Surrogate data sampled from the same distribution as the classifier's training data
- Classifier's feedback to label surrogate data



# Experiments on PDF Malware Detection

- **PDF:** hierarchy of interconnected objects (keyword/value pairs)


```
13 0 obj
<< /Kids [ 1 0 R 11 0 R ]
/Type /Page
... >> end obj
17 0 obj
<< /Type /Encoding
/Differences [ 0 /C0032 ] >>
endobj
```

| **Features:** *keyword count* | |
|---|---|
| /Type | 2 |
| /Page | 1 |
| /Encoding | 1 |
| ... | |

- **Adversary's capability**
  - adding up to $d_{max}$ objects to the PDF
  - removing objects may compromise the PDF file (and embedded malware code)!

$$\min_{x'} g(x') - \; / \; p(x' \mid y = -1)$$

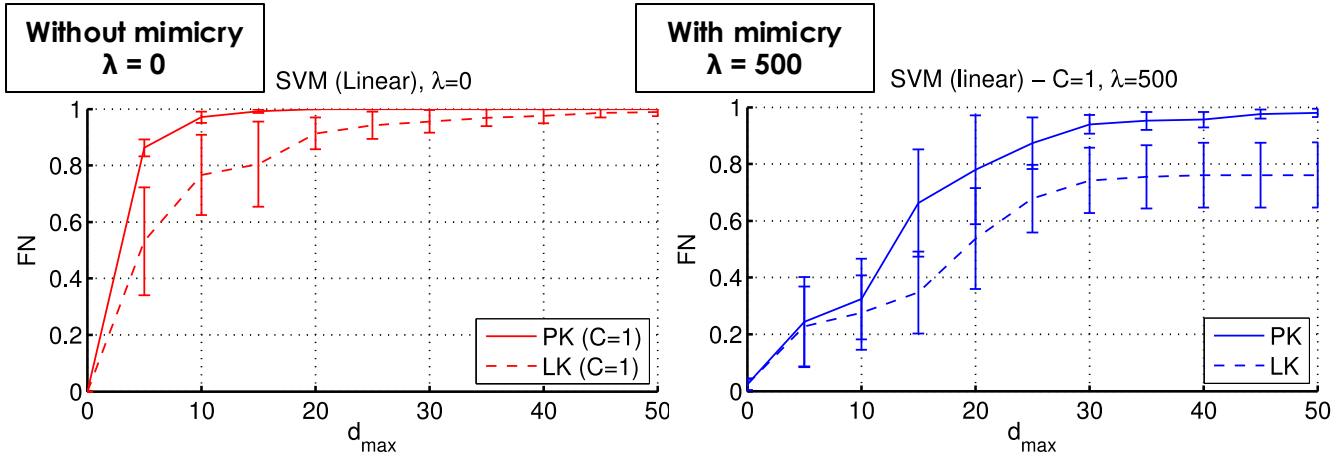
$$\text{s.t. } d(x, x') \leq d_{\max}$$

$$x \pounds x'$$

# Experiments on PDF Malware Detection
## Linear SVM

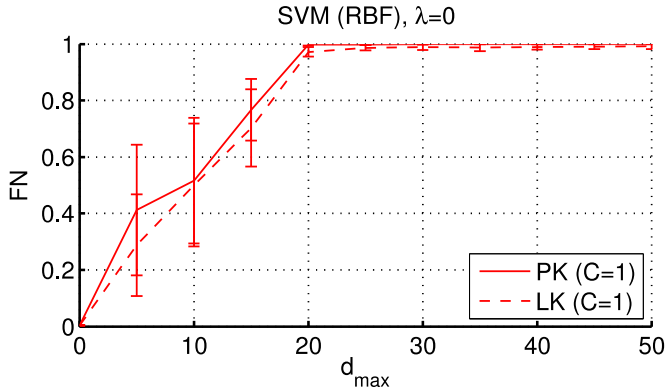
- **Dataset:** 500 malware samples (*Contagio*), 500 benign (Internet)
  - 5-fold cross-validation
  - Targeted (surrogate) classifier trained on 500 (100) samples

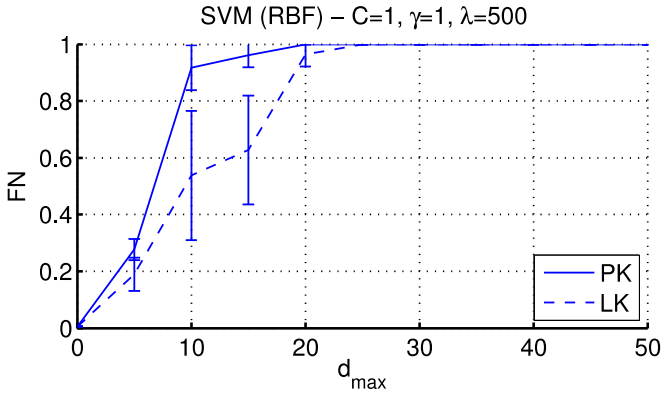
- **Evasion rate** (FN) at FP=1% vs max. number of added keywords
  - Perfect knowledge (PK); Limited knowledge (LK)

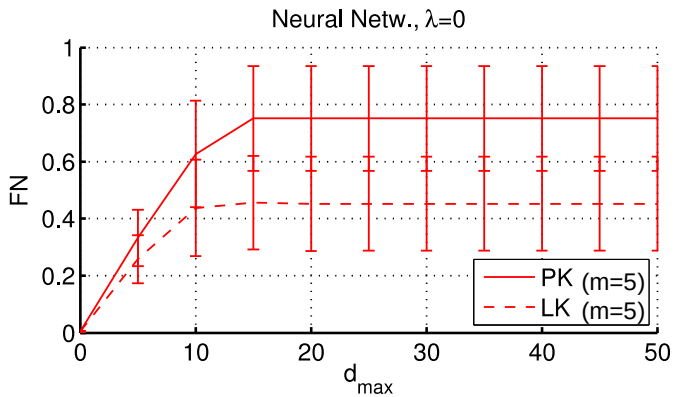


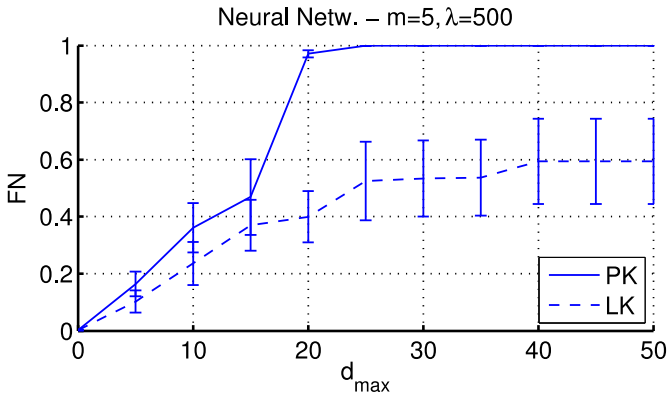
# Experiments on PDF Malware Detection

## SVM with RBF kernel, Neural Network



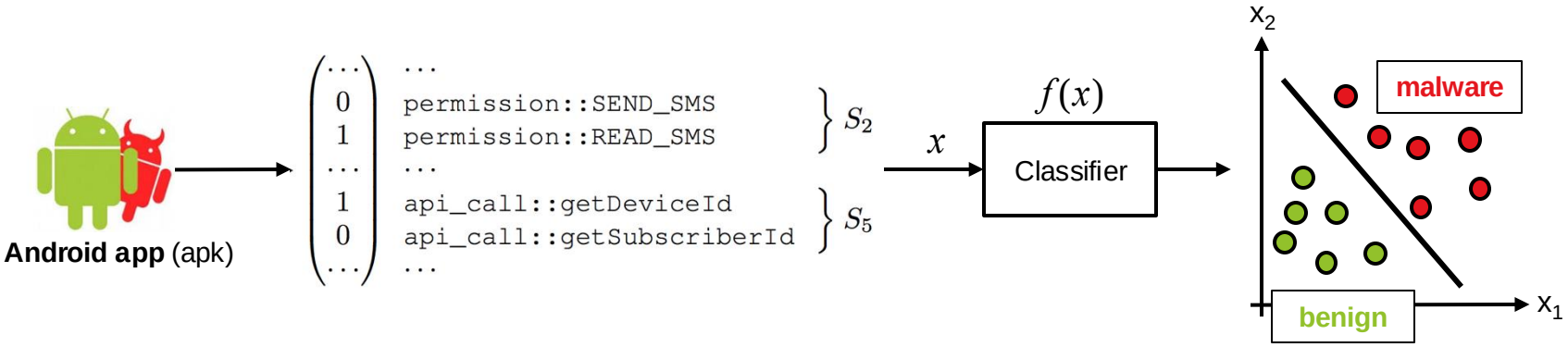



# Experiments on Android Malware Detection

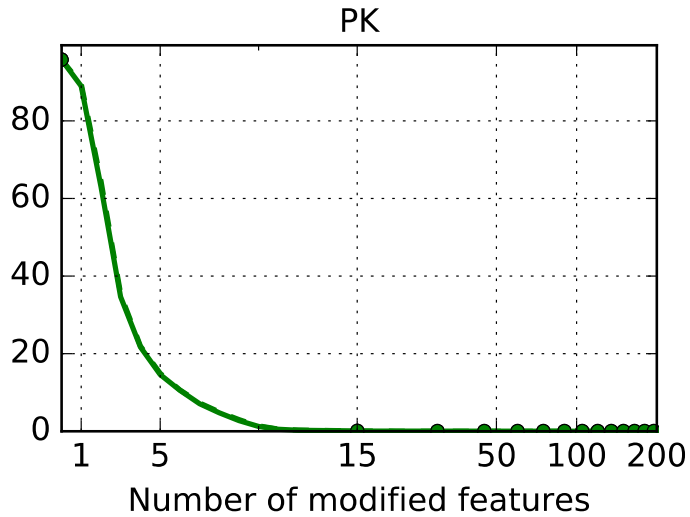
- **Drebin:** Arp et al., NDSS 2014
  - Android malware detection directly on the mobile phone
  - Linear SVM trained on features extracted from static code analysis

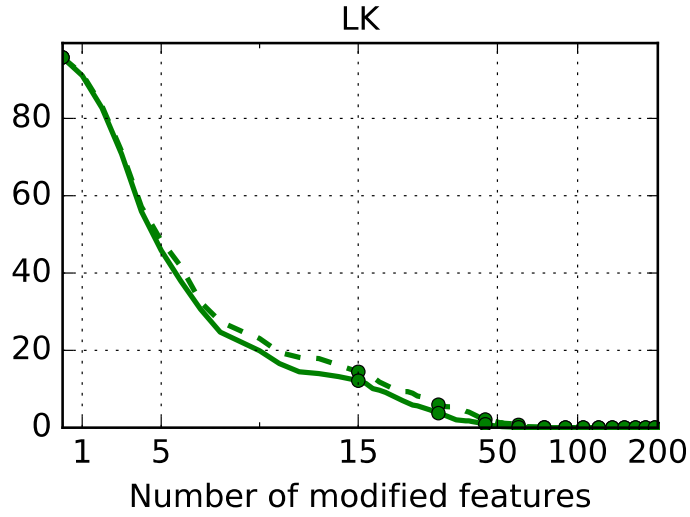
| Feature sets | | |
|---|---|---|
| manifest | $S_1$ | Hardware components |
| | $S_2$ | Requested permissions |
| | $S_3$ | Application components |
| | $S_4$ | Filtered intents |
| dexcode | $S_5$ | Restricted API calls |
| | $S_6$ | Used permission |
| | $S_7$ | Suspicious API calls |
| | $S_8$ | Network addresses |



# Experiments on Android Malware Detection

- **Dataset (Drebin):** 5,600 malware and 121,000 benign apps (TR: 30K, TS: 60K)
- **Detection rate** at FP=1% vs max. number of manipulated features (averaged on 10 runs)
  - Perfect knowledge (PK) white-box attack; Limited knowledge (LK) black-box attack

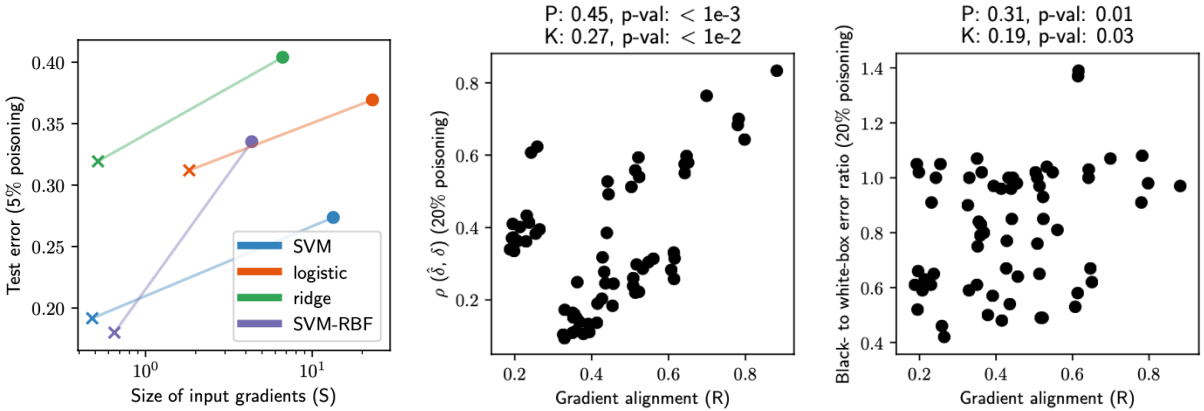


# Why Do Adversarial Attacks Transfer? (USENIX Sec. 2019)

- Transferability is the ability of an attack developed against a surrogate model to succeed also against a different target model

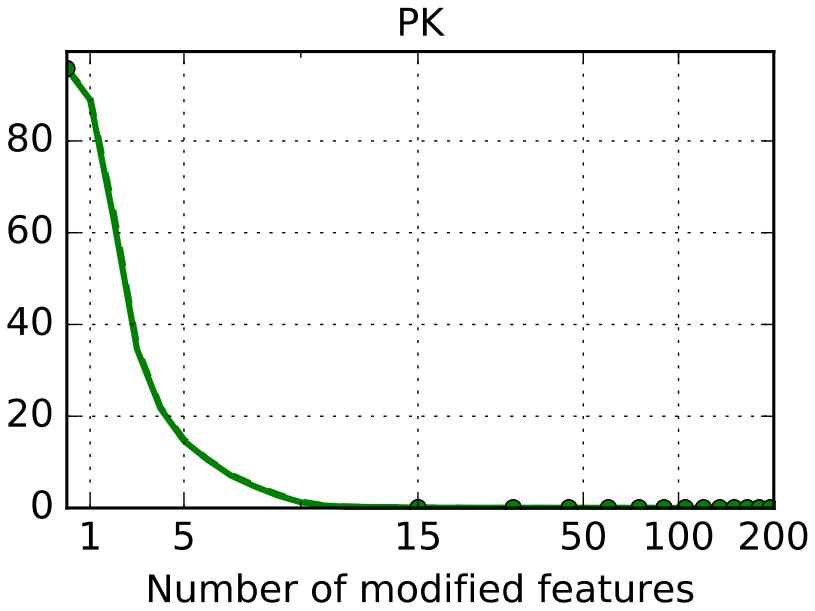
- In our paper, we show that *transferability* depends on
  - the **vulnerability of the target model**, and
  - the **alignment of** (poisoning/evasion) **gradients** between the target and the surrogate model



# Take-home Messages

- Linear and non-linear *supervised* classifiers are vulnerable to well-crafted evasion attacks

- Performance evaluation should be always performed as a function of the adversary's knowledge and capability via **Security Evaluation Curves**



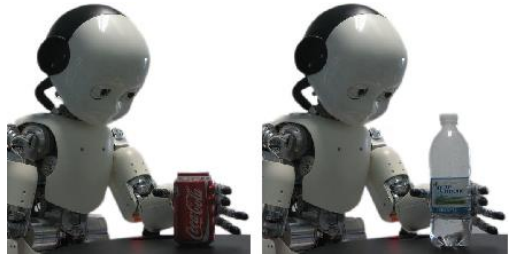
# Hands-on Demo

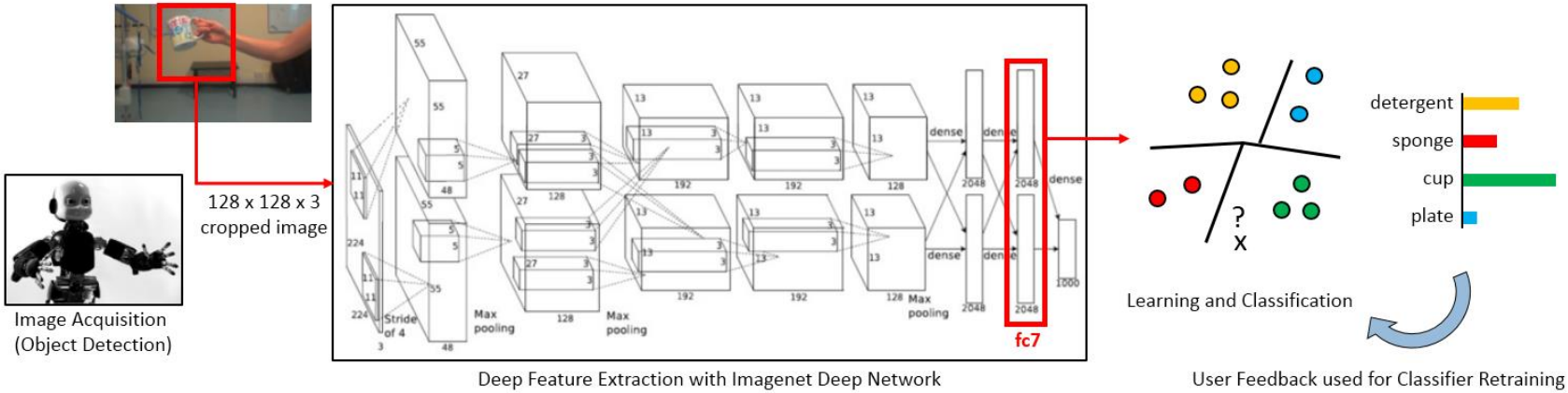

https://github.com/unica-mlsec/mlsec/blob/main/notebooks/advx-challenge.ipynb

# Evasion of Multiclass Classifiers

# Is Deep Learning Safe for Robot Vision?

- Evasion attacks against the iCub humanoid robot
  - Deep Neural Network used for visual object recognition

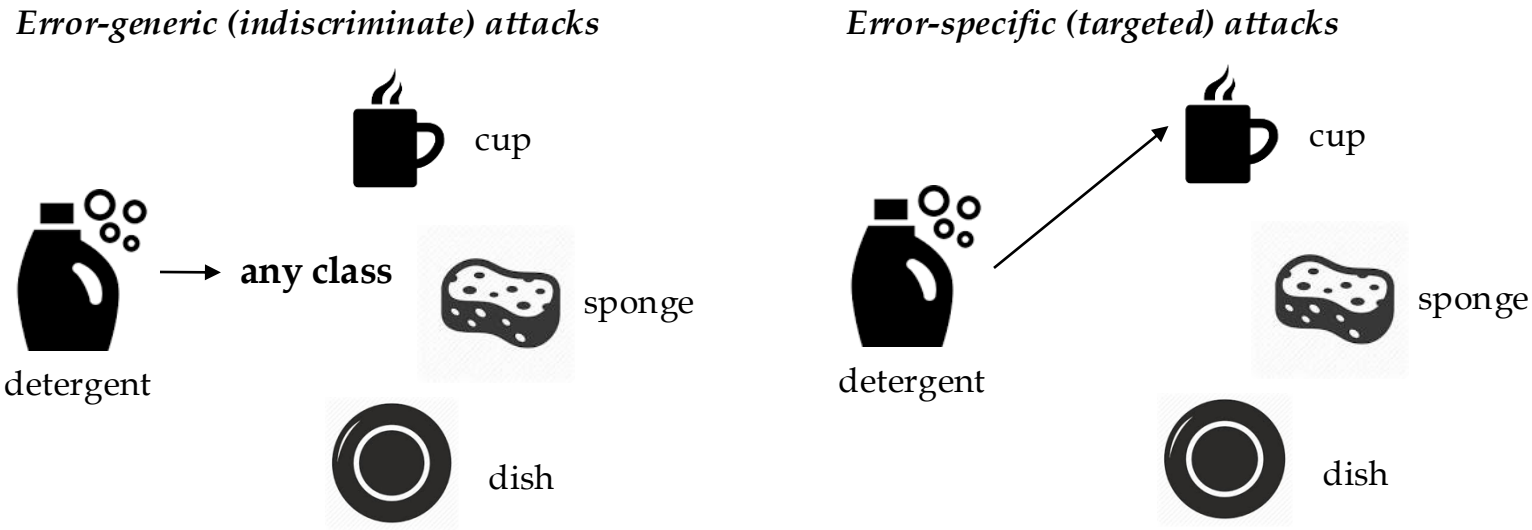




# iCubWorld28 Data Set: Example Images

# From Binary to Multiclass Evasion

- In multiclass problems, classification errors occur in different classes.
- Thus, the attacker may aim:
  1. to have a sample misclassified as any class different from the true class **(error-generic attacks)**
  2. to have a sample misclassified as a specific class **(error-specific attacks)**



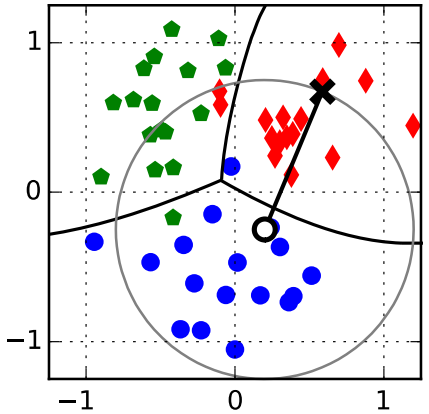
# Error-generic (Indiscriminate) Evasion

- **Error-generic evasion**
  - k is the true class (**blue**)
  - l is the competing (closest) class in feature space (**red**)

$$\Omega(\boldsymbol{x}) = f_k(\boldsymbol{x}) - \max_{l \neq k} f_l(\boldsymbol{x})$$

- The attack minimizes the objective to have the sample misclassified as the *closest* class (could be any!)

$$\begin{aligned} \min_{\boldsymbol{x}'} \quad & \Omega(\boldsymbol{x}')\,, \\ \text{s.t.} \quad & d(\boldsymbol{x}, \boldsymbol{x}') \leq d_{\max}\,, \\ & \boldsymbol{x}_{\text{lb}} \preceq \boldsymbol{x}' \preceq \boldsymbol{x}_{\text{ub}}\,, \end{aligned}$$



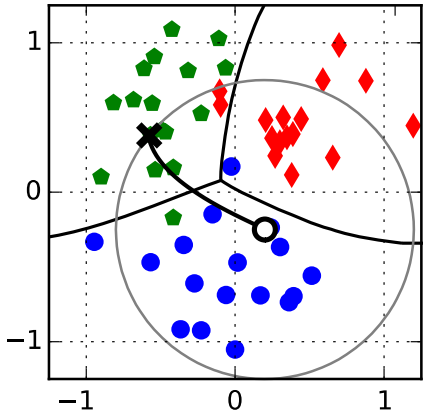
# Error-specific (Targeted) Evasion

- **Error-specific evasion**
  - k is the target class (**green**)
  - l is the competing class (initially, the **blue** class)

$$\Omega(\boldsymbol{x}) = f_k(\boldsymbol{x}) - \max_{l \neq k} f_l(\boldsymbol{x})$$

- The attack <u>maximizes</u> the objective to have the sample misclassified as the *target* class

$$\begin{aligned} \max_{\boldsymbol{x}'} \quad & \Omega(\boldsymbol{x}')\,, \\ \text{s.t.} \quad & d(\boldsymbol{x}, \boldsymbol{x}') \leq d_{\max}\,, \\ & \boldsymbol{x}_{\text{lb}} \preceq \boldsymbol{x}' \preceq \boldsymbol{x}_{\text{ub}}\,, \end{aligned}$$

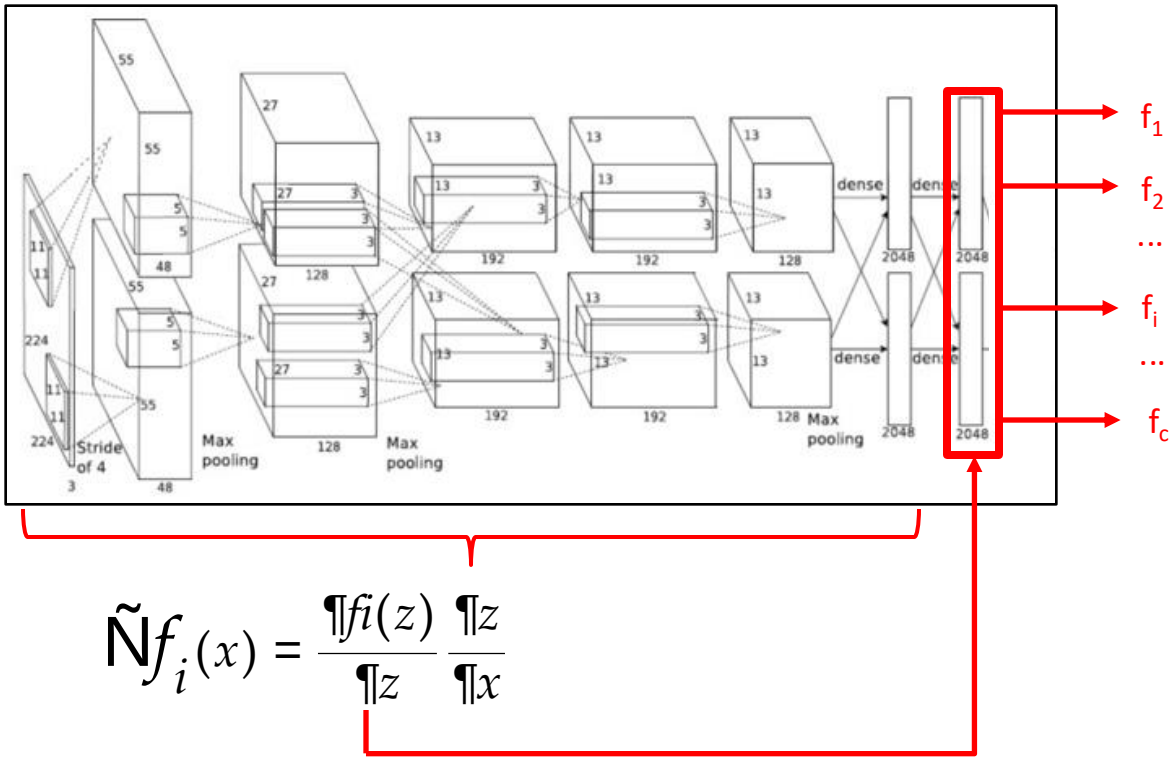


# Adversarial Examples – Gradient Computation

**Gradient-based attacks**

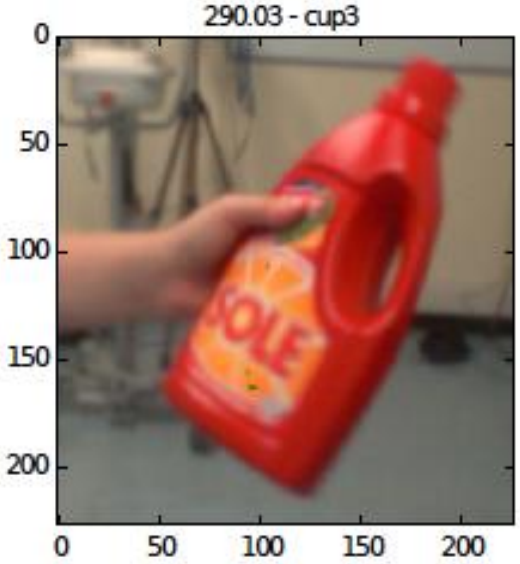
The gradient is computed with the chain rule

- the gradient of $f_i(\boldsymbol{z})$ can be computed if the chosen classifier is differentiable, and then
- be backpropagated through the DNN with automatic differentiation



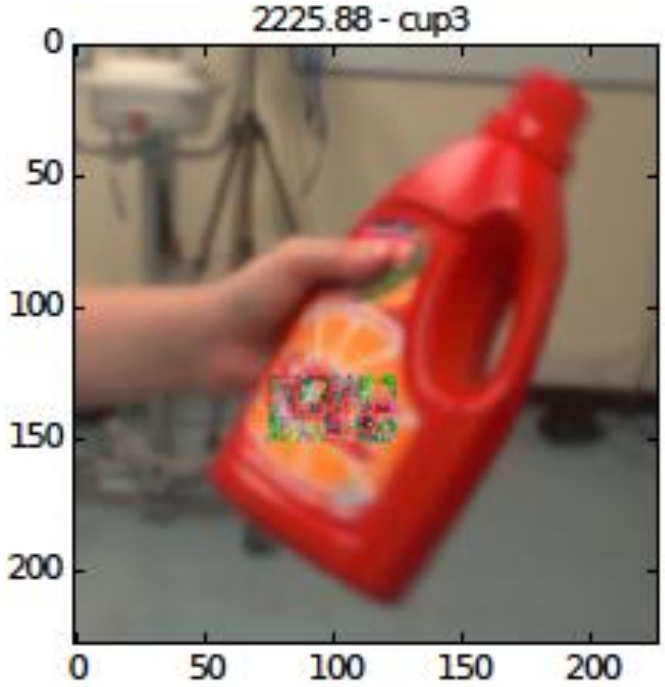

$$\tilde{\nabla} f_i(x) = \frac{\partial f_i(z)}{\partial z} \frac{\partial z}{\partial x}$$

# Example of Adversarial Images against iCub




- An adversarial example from class *laundry-detergent*, modified by the proposed algorithm to be misclassified as *cup*

# The *Sticker* Attack against iCub




*Adversarial example* generated by manipulating only a specific region, to simulate a sticker that could be applied to the real-world object.

This image is classified as *cup*.



[Melis, Biggio et al., Is Deep Learning Safe for Robot Vision? ICCVW ViPAR 2017]

# Loss Functions for Targeted/Indiscriminate Attacks

- Using the same loss functions used to train ML models, denoted with $L$
  - and given an input sample $\boldsymbol{x}$ and its true class label $y$

- We can formalize/generalize adversarial attacks as:
  - $\max_{\boldsymbol{\delta}} L(\boldsymbol{x}+\boldsymbol{\delta}, y, \boldsymbol{\theta}) = \min_{\boldsymbol{\delta}} -L(\boldsymbol{x}+\boldsymbol{\delta}, y, \boldsymbol{\theta})$, for indiscriminate attacks
  - $\min_{\boldsymbol{\delta}} L(\boldsymbol{x}+\boldsymbol{\delta}, y_t, \boldsymbol{\theta})$, for targeted attacks, with $y_t \neq y$

# The Effect of Different Norms

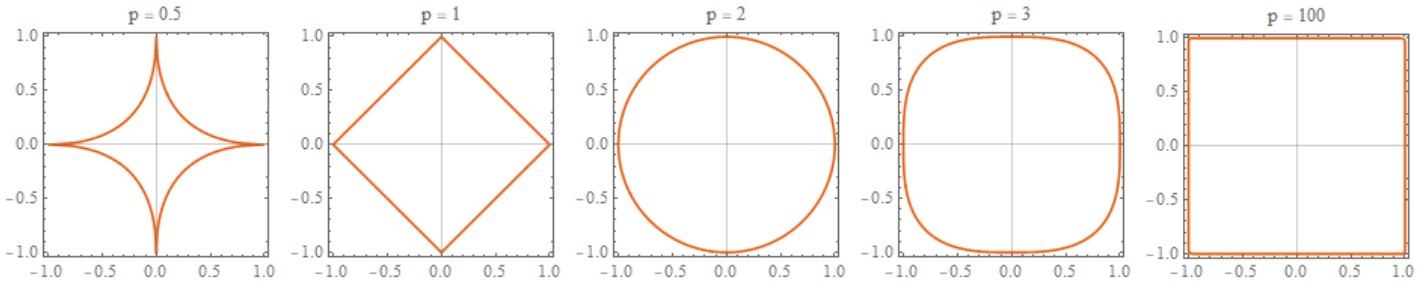
# Constraints are Regularizers

Typically, (convex) $\ell_p$ norms are used as constraints/regularizers

$\ell_p$ norms with $p \geq 1$, $\ell_p(\boldsymbol{x}) = \left(\sum_j |x_j|^p\right)^{1/p}$

Most popular examples

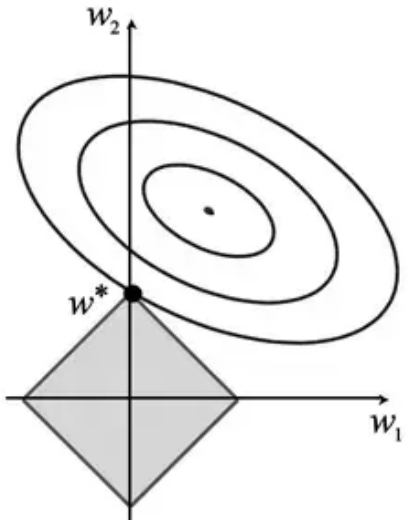
- $\ell_0$ is not convex, and amounts to counting non-zero elements in $x$
- $\ell_1 = |x_1| + |x_2| + \cdots + |x_d|$
- $\ell_2 = x_1^2 + x_2^2 + \ldots + x_d^2$
- $\ell_\infty = \max_j |x_j|$



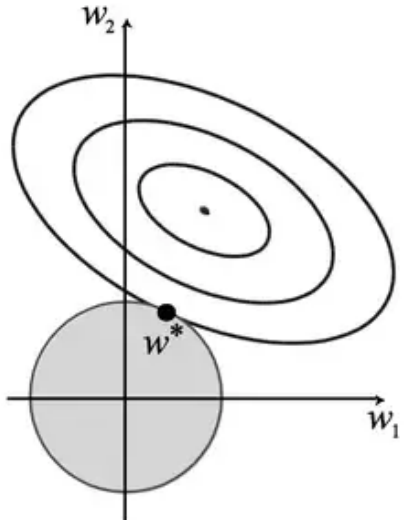
# Sparsity

- $\ell_0$ and $\ell_1$ regularization enforce *sparsity*, i.e., many values in $\boldsymbol{x}$ will be set to zero
  - Why? The optimum is often found at one of the vertices!

- Sparsity helps automatically perform feature selection
- Features assigned $w_j = 0$ can be disregarded



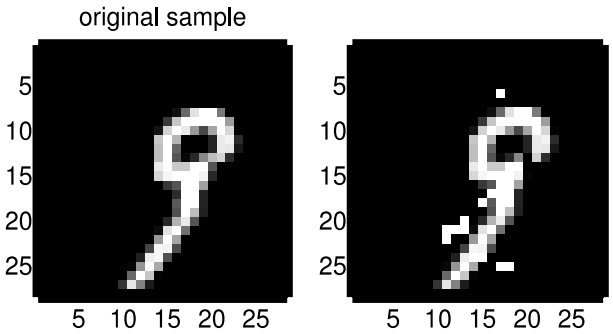

$\ell_1$



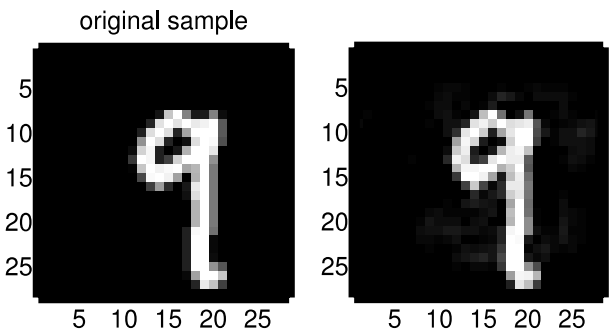

$\ell_2$

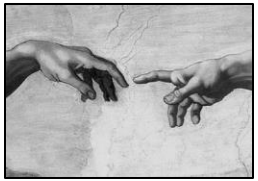
# Sparse vs Dense Attacks

**Sparse Evasion Attacks (L1-norm constrained)**




**Dense Evasion Attacks (L2-norm constrained)**





# 2014: Deep Learning Meets Adversarial Machine Learning

# The Discovery of Adversarial Examples

## Intriguing properties of neural networks


**Christian Szegedy** — Google Inc.
**Wojciech Zaremba** — New York University
**Ilya Sutskever** — Google Inc.
**Joan Bruna** — New York University
**Dumitru Erhan** — Google Inc.
**Ian Goodfellow** — University of Montreal
**Rob Fergus** — New York University, Facebook Inc.

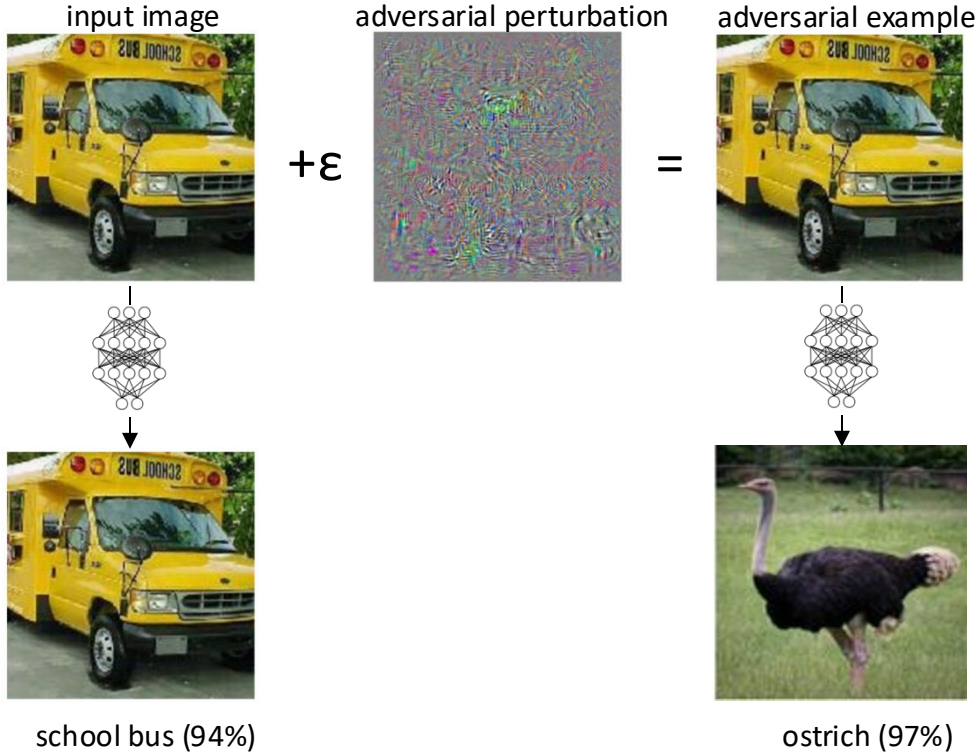

... we find that deep neural networks learn **input-output mappings** that are fairly **discontinuous** to a significant extent. We can cause the network to misclassify an image by applying a certain **hardly perceptible perturbation**, which is found by maximizing the network's prediction error ...



Szegedy, Goodfellow et al., Intriguing Properties of NNs, ICLR 2014

# Adversarial Examples against Deep Neural Networks

- Szegedy et al. (2014) *independently developed gradient-based attacks* against DNNs
- They were investigating **model interpretability**, trying *to understand at which point a DNN prediction changes*
- They found that the **minimum perturbations required to trick DNNs were really small**, even imperceptible to humans



# Creation of Adversarial Examples

- Minimize $\|r\|_2$ subject to:
  1. $f(x+r) = l$
  2. $x + r \in [0,1]^m$

The adversarial image $x + r$ is visually hard to distinguish from $x$
Informally speaking, the solution $x + r$ is the closest image to $x$ classified as $l$ by f

The solution is approximated using using a box-constrained limited-memory BFGS

**School Bus (x)**

**Adversarial Noise (*r*)**

**Ostrich**
**Struthio Camelus**



Szegedy, Goodfellow et al., Intriguing Properties of NNs, ICLR 2014

# Minimum-norm vs Maximum-confidence Attacks

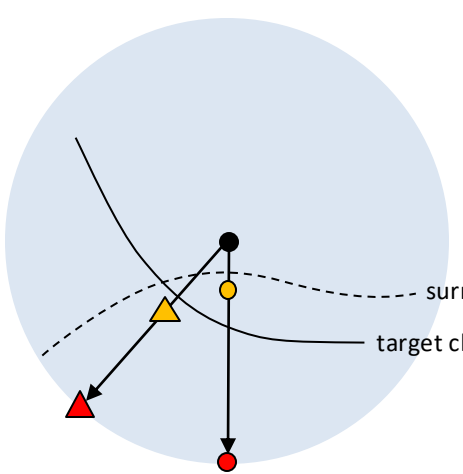
- Szegedy et al., ICLR 2014 aim to measure the minimum distance to evasion
  - Better suited to the analysis of adversarial robustness in the white-box case

- Biggio et al., ECML 2013 maximizes misclassification confidence within a given budget
  - The intuition was to craft attacks that are more difficult to detect, and to evade classifiers with higher probability also when knowledge of the boundary is not perfect (*transfer attacks*)

**Adversary's goal.** As suggested by Laskov and Kloft [17], the adversary's goal should be defined in terms of a utility (loss) function that the adversary seeks to maximize (minimize). In the evasion setting, the attacker's goal is to manipulate a single (without loss of generality, positive) sample that should be misclassified. Strictly speaking, it would suffice to find a sample $\mathbf{x}$ such that $g(\mathbf{x}) < -\epsilon$ for any $\epsilon > 0$; *i.e.*, the attack sample only just crosses the decision boundary.[2] Such attacks, however, are easily thwarted by slightly adjusting the decision threshold. A better strategy for an attacker would thus be to create a sample that is misclassified with high confidence; *i.e.*, a sample minimizing the value of the classifier's discriminant function, $g(\mathbf{x})$, subject to some feasibility constraints.

Biggio, Roli et al., ECML PKDD 2013

# Minimum-norm vs Maximum-confidence Attacks



- initial / source example
- minimum-distance *black-box* adversarial example
- minimum-distance *white-box* adversarial example
- maximum-confidence *black-box* adversarial example
- maximum-confidence *white-box* adversarial example

surrogate classifier $\hat{f}(\boldsymbol{x})$ used to craft *black-box* adversarial examples

target classifier $f(\boldsymbol{x})$ used to craft *white-box* adversarial examples



Demontis et al., Why do adversarial attacks transfer? USENIX 2019

# Many Black Swans After 2014

- Several defenses have been proposed against adversarial examples, and more powerful attacks have been developed to show that they are ineffective.
  - Remember the arms race?

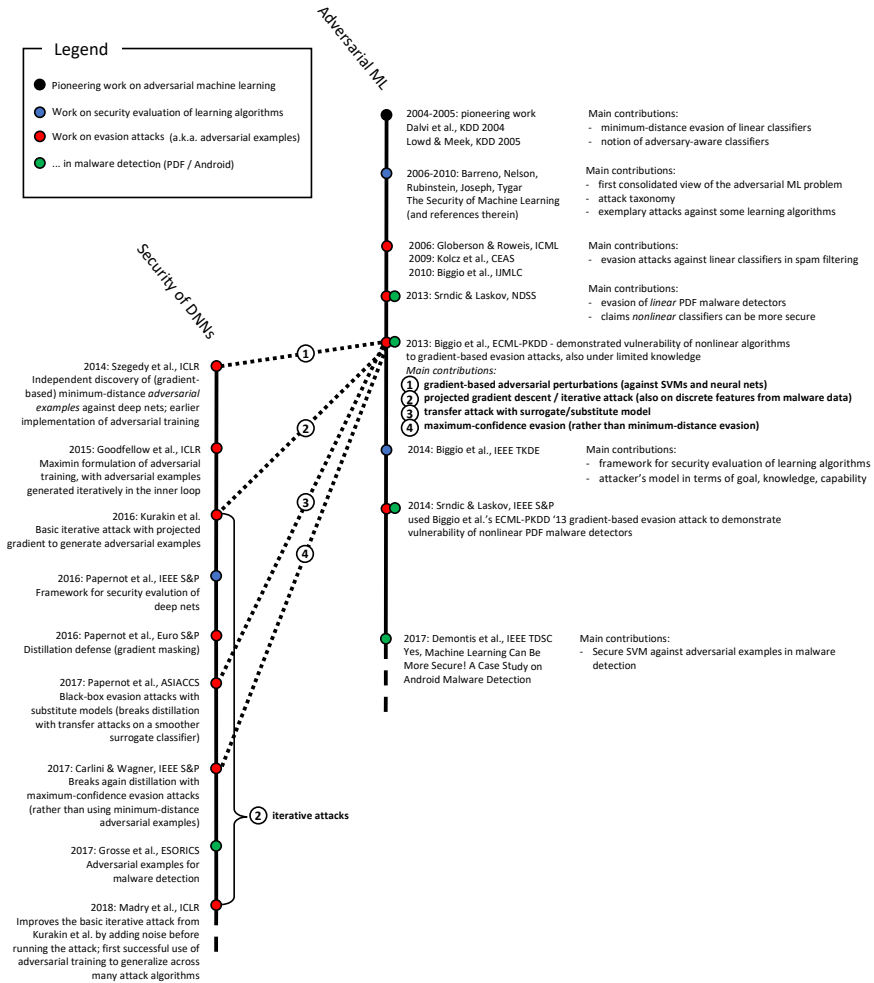
- Most of these attacks are modifications to the optimization problems reported for evasion attacks / adversarial examples, using different gradient-based solution algorithms, initializations and stopping conditions.



Search https://arxiv.org with keywords "adversarial examples"

# Timeline of Learning Security

Biggio and Roli, **Wild Patterns**: *Ten Years After The Rise of Adversarial Machine Learning*, Pattern Recognition, 2018



# Attack Algorithms: A Unifying View

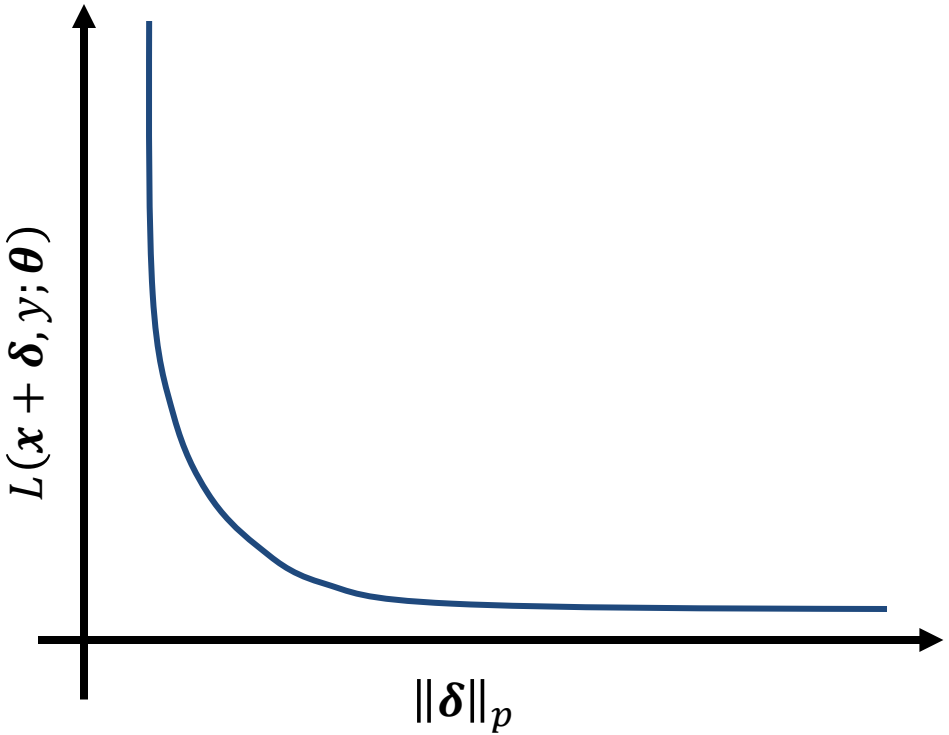
# General Categorization

$$\min_{\boldsymbol{\delta}} [L(\boldsymbol{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}), \|\boldsymbol{\delta}\|_p]$$

Minimize loss to cause misclassification

Minimize perturbation size (measured with Lp norm)

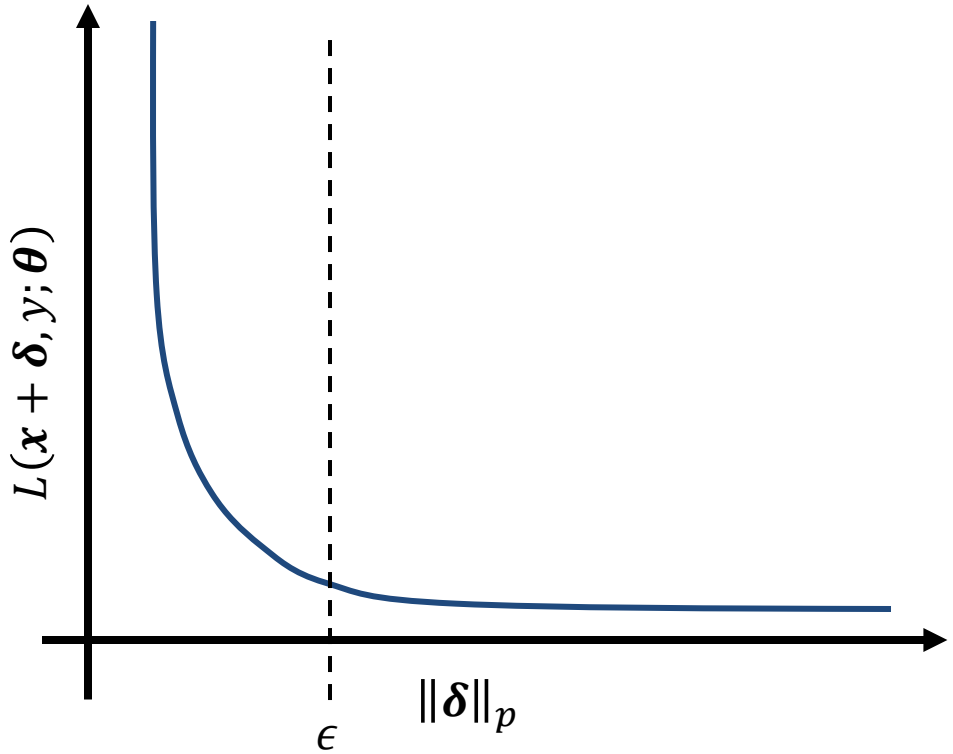
# Pareto Frontier




**Trade-off between misclassification confidence and perturbation size**

*Pareto-optimal* solutions with different trade-offs are found along the blue curve (Pareto frontier)

# Hard-constraint: Maximum-confidence Attacks

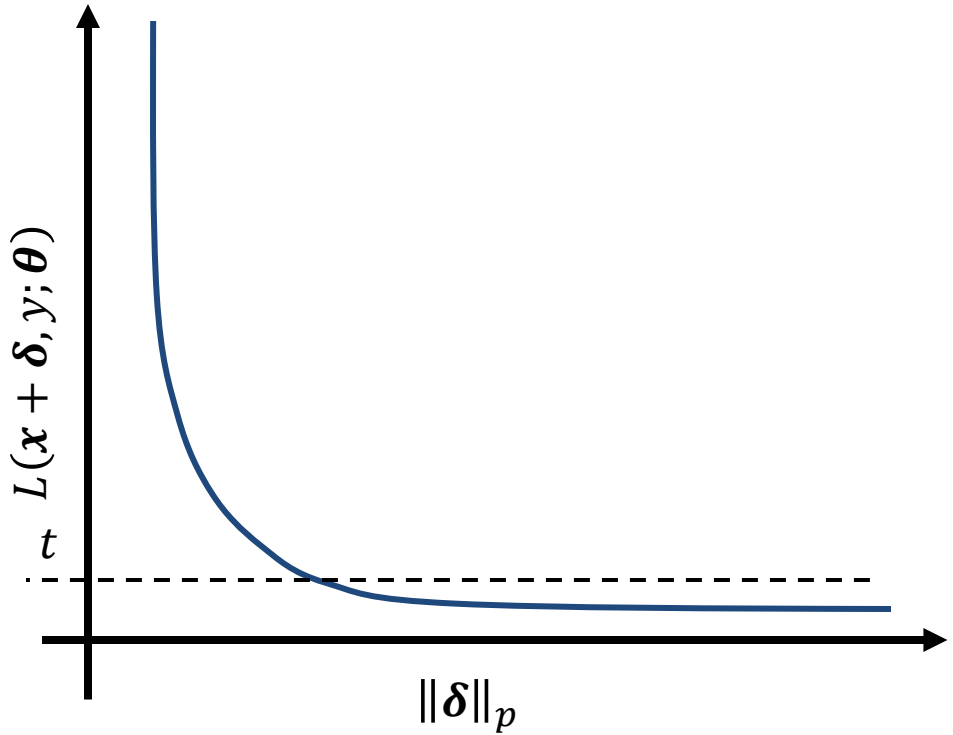



Minimize loss to cause misclassifiation (FGSM, PGD)

The perturbation is checked as hard constraint, bound on maximum manipulation

$$\min \; L(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta}),$$
$$\text{s.t.} \quad \|\boldsymbol{\delta}\|_p \leq \epsilon$$

# Hard-constraint: Minimum-norm Attacks



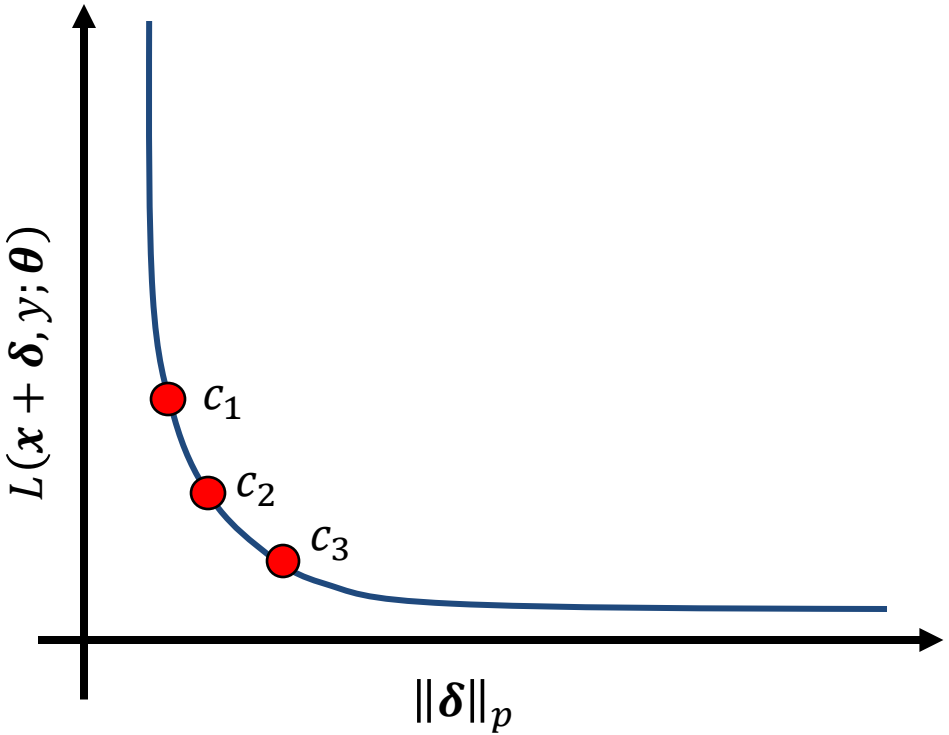

Minimize perturbation w.r.t. Lp norm

Score is used only as a constraint, not optimized

Hard to solve directly – normally a soft-constraint is used instead

$$\min \|\boldsymbol{\delta}\|_p$$
$$\text{s.t. } L(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta}) < t$$

# Soft-constraint: Trade-off Solution




All constraints are imposed as quantities modulated by coefficients, behaving as regularizers

Modulating the multipliers shifts the solution towards trade-off between score and distance

$$\min L(\boldsymbol{x}+\boldsymbol{\delta}, y; \boldsymbol{\theta}) + c\|\boldsymbol{\delta}\|_p$$

# Generalized Gradient-Descent Attack Algorithm

**Input** : $\boldsymbol{x}$, the initial point; $y$, the true class of the initial point; $n$, the number of iterations; $\alpha$, the learning rate; $f$, the target model; $\Delta$, the considered region.

**Output** : $\boldsymbol{x}^\star$, the solution found by the algorithm

$\boldsymbol{x}_0 \leftarrow \texttt{initialize}(\boldsymbol{x})$ ▷ Initialize starting point

$\hat{\boldsymbol{\theta}} \leftarrow \texttt{approximation}(\boldsymbol{\theta})$ ▷ Approximate model parameters

$\boldsymbol{\delta}_0 \leftarrow \mathbf{0}$ ▷ Initial $\delta$

**for** $i \in [1, n]$ **do**

  $\boldsymbol{\delta}' \leftarrow \boldsymbol{\delta}_i - \alpha \nabla_{\boldsymbol{x}_i} L(\boldsymbol{x}_0 + \boldsymbol{\delta}_i, y; \hat{\boldsymbol{\theta}})$ ▷ Compute optimizer step

  $\boldsymbol{\delta}_{i+1} \leftarrow \texttt{apply-constraints}(\boldsymbol{x}_0, \boldsymbol{\delta}', \Delta)$ ▷ Apply constraints (if needed)

$\boldsymbol{\delta}^\star \leftarrow \texttt{best}(\boldsymbol{\delta}_0, ..., \boldsymbol{\delta}_n)$ ▷ Choose best perturbation

**return** $\boldsymbol{\delta}^\star$

# Maximum-confidence Attacks

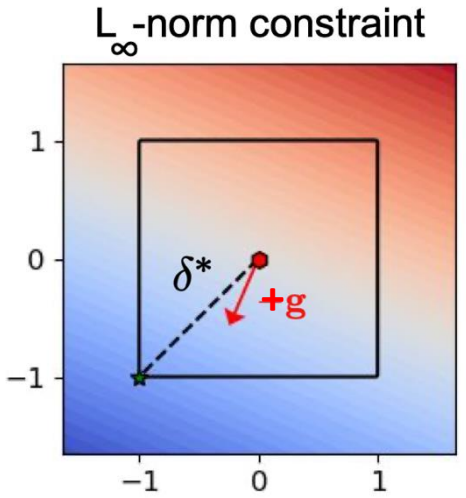
# Fast Gradient Sign Method (2015)

- Perturbed image obtained as: $\boldsymbol{x}^\star = \boldsymbol{x} + \epsilon \, \text{sign}(\nabla L(\boldsymbol{x}, y; \boldsymbol{\theta}))$
- Why?
  - $\max_{||\boldsymbol{\delta}||_\infty \leq \epsilon} L(\boldsymbol{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}) \approx$ (linear approximation)

$$\max_{||\boldsymbol{\delta}||_\infty \leq \epsilon} L(\boldsymbol{x}, y; \boldsymbol{\theta}) + \boldsymbol{\delta}^{\mathrm{T}} \nabla L(\boldsymbol{x}, y; \boldsymbol{\theta}) =$$

$$L(\boldsymbol{x}, y; \boldsymbol{\theta}) + \max_{||\boldsymbol{\delta}||_\infty \leq \epsilon} \boldsymbol{\delta}^{\mathrm{T}} \boldsymbol{g}$$

- The solution is to set $\boldsymbol{\delta}^\star = \epsilon \, \text{sign}(\boldsymbol{g})$
- Loss function at $\boldsymbol{\delta}^\star$ (optimum): $L(\boldsymbol{x}, y; \boldsymbol{\theta}) + \epsilon \, ||\boldsymbol{g}||_1$
  - (cf. *dual norm* and *steepest gradient descent)*



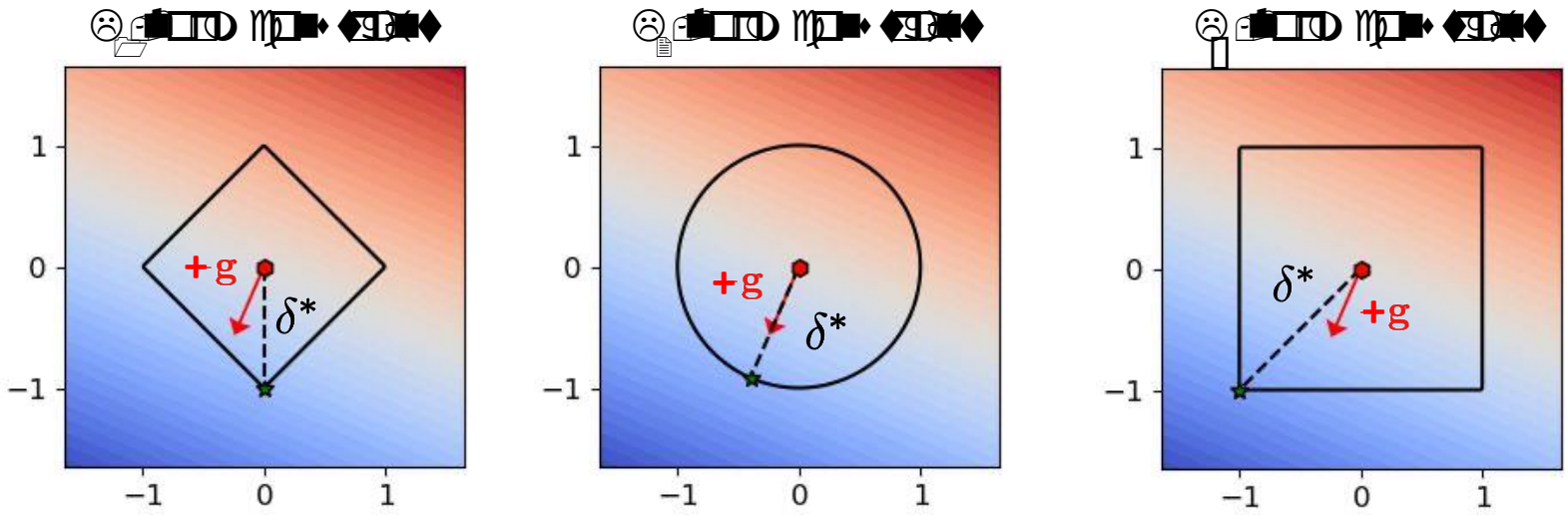



Goodfellow et al., Explaining and Harnessing Adversarial Examples ICLR 2015

# Why Are Attacks Effective / Imperceptible against DNNs?

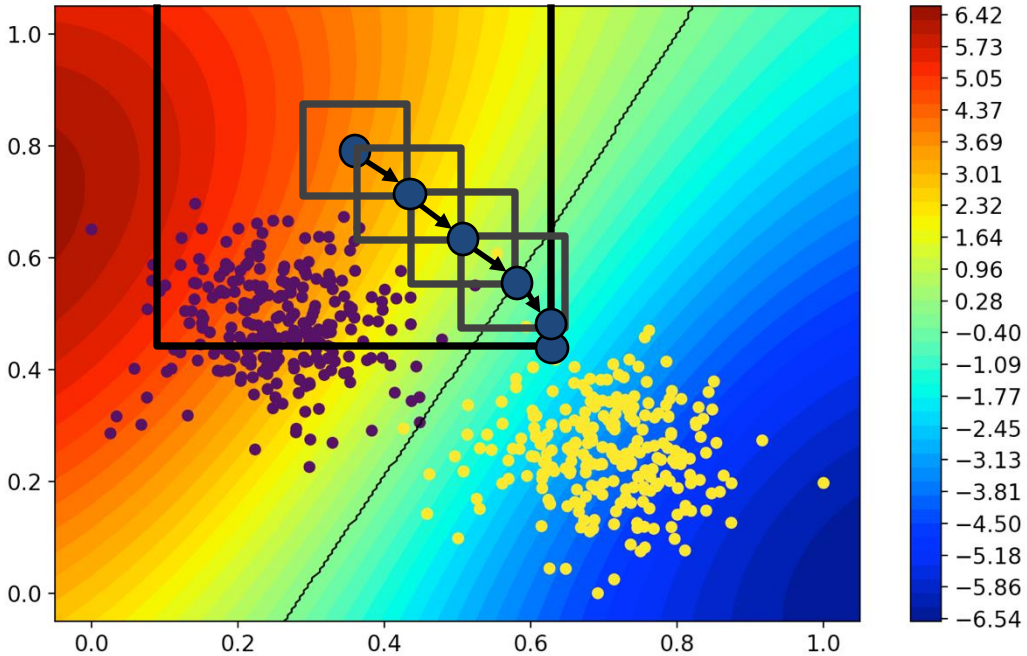
- Let's pretend that a deep network behaves in a linear way…
- Under this linearity assumption, the output of the net to the adversarial input $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta}$ is $\mathbf{w}^{\mathrm{T}} \tilde{\boldsymbol{x}} = \mathbf{w}^{\mathrm{T}} \boldsymbol{x} + \mathbf{w}^{\mathrm{T}} \boldsymbol{\delta}$
- The key concept is that **if the dimensionality of the input $\boldsymbol{x}$** is very *high* and the vector of the adversarial perturbation $\boldsymbol{\delta}$ is aligned with the "classifier" (its weight vector $\mathbf{w}$), then **many infinitesimal changes to the input** add up to **one large change to the output**.

# Fast Gradient Method (FGM): Extension to Other Norms



# Projected Gradient Descent (2018)

- Also known as *Basic Iterative Method* (BIM)
- PGD is just the iterative version of FGM
- *Number of iterations* and *step size* need to be fixed a priori

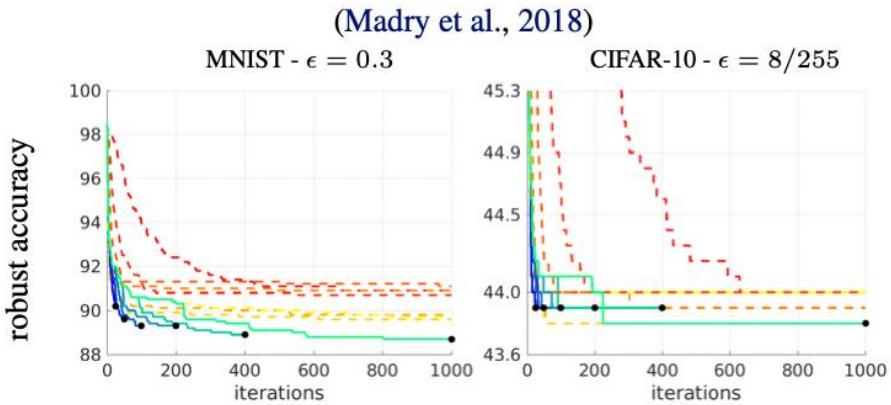





Madry et al. Towards deep learning models resistant to adversarial attacks, ICLR 2018
Kurakin et al., Adversarial Examples in the Physical World, ICLR-W 2017

# AutoPGD (2020)

- It dynamically halves the step size while optimizing the attack, if no improvement in the loss function is observed




**Algorithm 1** APGD

**Input:** $f$, $S$, $x^{(0)}$, $\eta$, $N_{\text{iter}}$, $W = \{w_0, \dots, w_n\}$
**Output:** $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_S\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
**for** $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_S\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_S\left(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\right)$
  **if** $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  **end if**
  **if** $k \in W$ **then**
    **if** Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\max}$
    **end if**
  **end if**
**end for**

# AutoPGD (2020)

- AutoPGD-CE uses the cross-entropy loss

$$\mathrm{CE}(x, y) = -\log p_y = -z_y + \log\left(\sum_{j=1}^{K} e^{z_j}\right)$$

- AutoPGD-DLR uses a novel *scale-invariant* loss: **Difference of Logits Ratio (DLR)**

$$\mathrm{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}$$

# Minimum-norm Attacks

# Adversarial Examples against DNNs (2014)

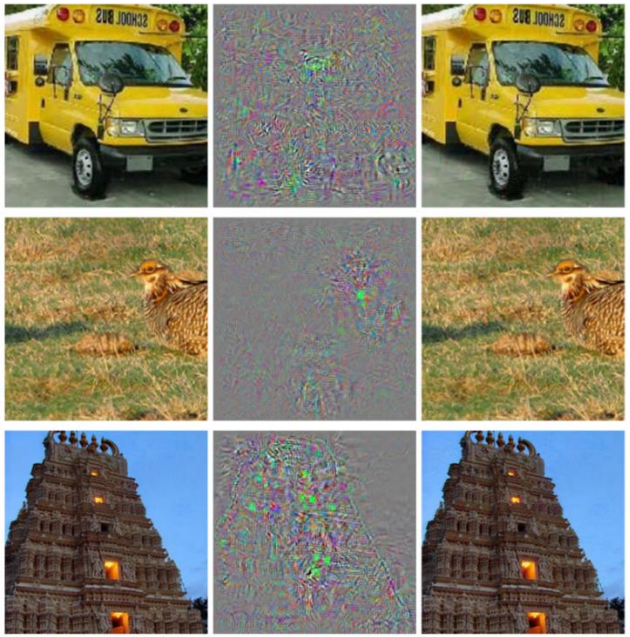
- Szegedy et al. formalized the problem as

$$\min_{\boldsymbol{\delta}} \ ||\boldsymbol{\delta}||_2$$
$$\text{s.t.} \quad f(\boldsymbol{x}+\boldsymbol{\delta}) = y_t \qquad (y_t \neq y)$$
$$\mathbf{x}+\boldsymbol{\delta} \in [0,1]^{\mathrm{d}}$$

- Relaxation to use L-BFGS-B:
  *L-BFGS-B*: https://en.wikipedia.org/wiki/Limited-memory_BFGS

$$\min_{\boldsymbol{x}+\boldsymbol{\delta}\in[0,1]^{\mathrm{d}}} \ c \cdot ||\boldsymbol{\delta}||_2 + L(\boldsymbol{x}+\boldsymbol{\delta}, y_t, \boldsymbol{\theta})$$

  - where they find the minimum c>0 that achieves misclassification (via line search)
  - $L(\boldsymbol{x}+\boldsymbol{\delta}, y_t; \boldsymbol{\theta})$ is the cross-entropy loss

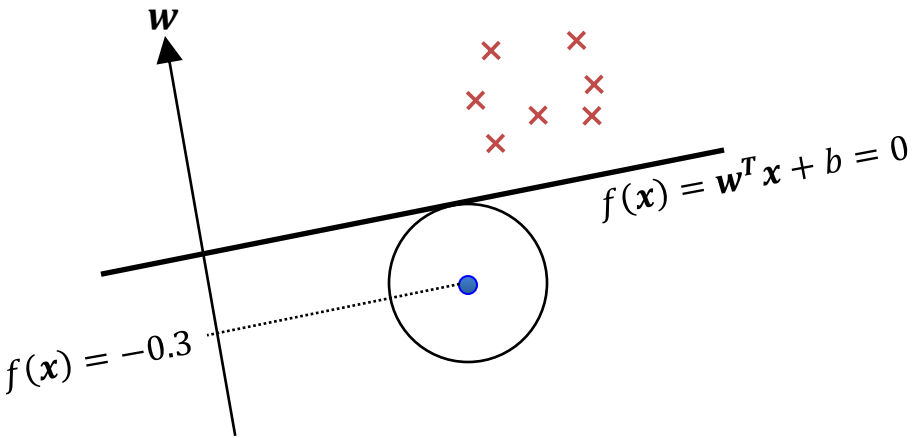




Szegedy et al., Intriguing Properties of Neural Networks, ICLR 2014

# DeepFool (2016)

- **Idea**: closed-form solution assuming a linear classifier, then iterate
  - For $L_2$, binary classifier: $\boldsymbol{x}^\star = \boldsymbol{x} - \frac{f(\boldsymbol{x})}{||\boldsymbol{w}||}\boldsymbol{w}$




Distance of $\boldsymbol{x}$ to the hyperplane: $\frac{|f(\boldsymbol{x})|}{||\boldsymbol{w}||}$

**Algorithm 1** DeepFool for binary classifiers

**input:** Image $\boldsymbol{x}$, classifier $f$.
**output:** Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
**while** $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
  $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2}\nabla f(\boldsymbol{x}_i)$,
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
  $i \leftarrow i + 1$.
**end while**
**return** $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.


Moosavi-Dezfooli et al., DeepFool: ..., CVPR 2016

# DeepFool (2016)

- In the multi-class case, the (signed) $L_p$ distance to the boundary $B$ is estimated as:

$$d(\boldsymbol{x}, B) = \frac{f_k(\boldsymbol{x}) - f_y(\boldsymbol{x})}{\left\| \nabla f_k(\boldsymbol{x}) - \nabla f_y(\boldsymbol{x}) \right\|_q}$$

- As in the binary case, the algorithm iterates to refine the initial guess, until a misclassification is achieved

**Algorithm 2** DeepFool: multi-class case

**input:** Image $\boldsymbol{x}$, classifier $f$.
**output:** Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
**while** $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  **for** $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  **end for**
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
**end while**
**return** $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$



Moosavi-Dezfooli et al., DeepFool: ..., CVPR 2016

# Jacobian Saliency-Map Attack (2016)

- JSMA uses saliency maps (i.e., *input gradients*) to compute perturbations

- Given $t$ as the target class, and $i$ as an input feature
  - it only retains the feature values that would change the output if *added (1st equation)* or *removed (2nd eq.)*

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left| \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} \right| \text{ otherwise} \end{cases}$$

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \\ \left| \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} \right| \left(\sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i}\right) \text{ otherwise} \end{cases}$$

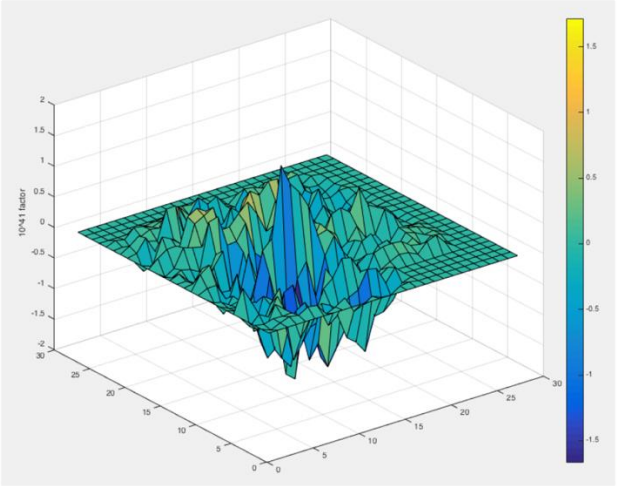


Fig. 7: Saliency map of a 784-dimensional input to the LeNet architecture (cf. validation section). The 784 input dimensions are arranged to correspond to the 28x28 image pixel alignment. Large absolute values correspond to features with a significant impact on the output when perturbed.

# Carlini-Wagner Attack (2017)

- Initially proposed to bypass one defensive mechanism (known as *distillation*)
  - **Problem**: cross-entropy loss exhibits vanishing gradients - attacks do not work correctly!
  - **Solution**: to define the so-called *logit* loss $L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \max_{k \neq y} f_k(x) - f_y(x)$
    - *Note*: if $L < 0$, an adversarial example is found

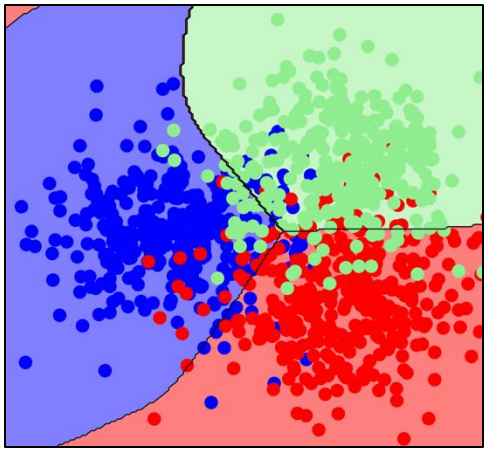
- **Goal:** to find minimum-norm adversarial examples, using relaxation:

$$\min_{\boldsymbol{\delta}} \ ||\boldsymbol{\delta}||_2 + c \cdot \max(L(\boldsymbol{x}, y, \boldsymbol{\theta}), -\kappa)$$
$$\text{s.t.} \quad \mathbf{x} + \boldsymbol{\delta} \in [0,1]^{\mathrm{d}}$$

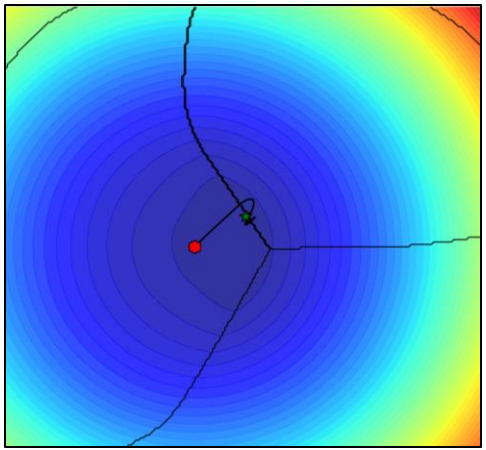
  - $c > 0$ is again chosen via line search
  - $\kappa \geq 0$ can be set to achieve misclassification with a non-zero confidence in the target class
  - The box constraint is also replaced via a change of variables (no constraints at all)

- **Solver:** The *Adam* algorithm is used to solve the unconstrained optimization, even though tuning $c$ is computationally costly (requires re-running the attack many times)
  - *Adam*: https://d2l.ai/chapter_optimization/adam.html

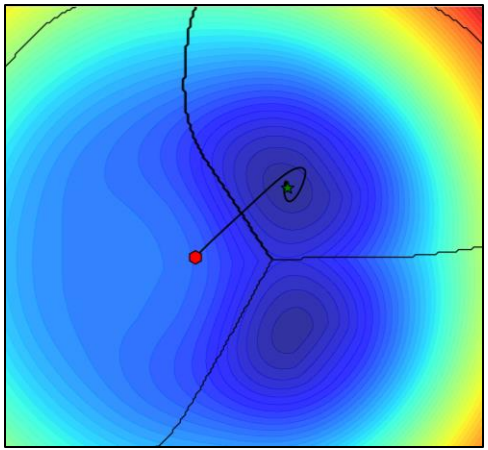
# Carlini-Wagner Attack (2017)



$\kappa = 0$



$\kappa = 2$

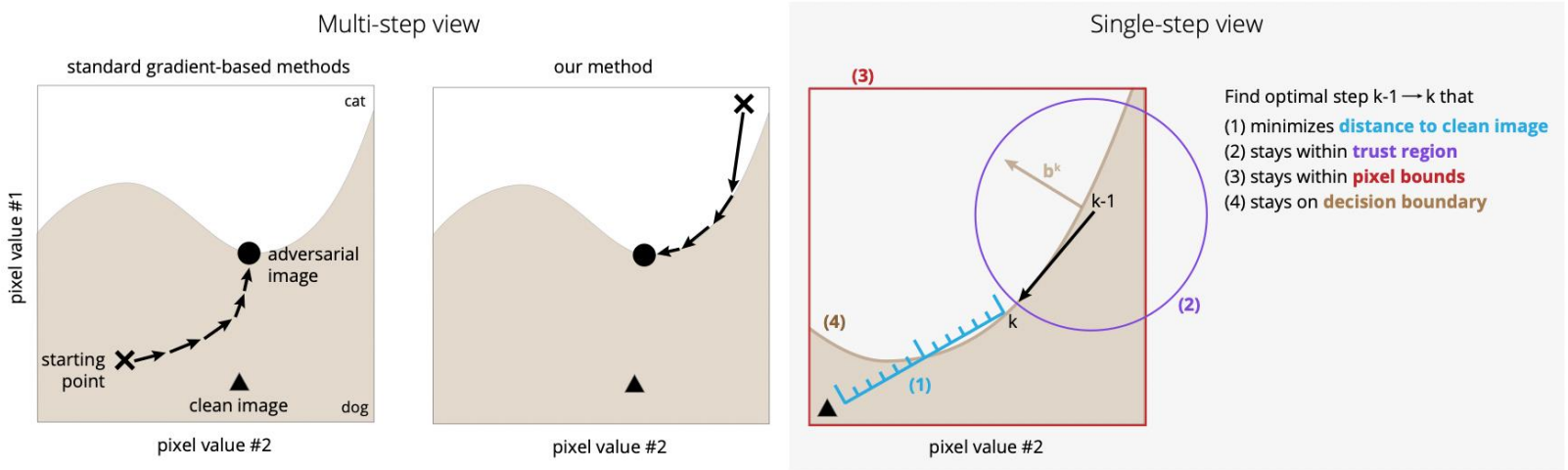




Carlini and Wagner, Towards Evaluating the Robustness of Neural Networks, S&P 2017

# Brendel-Bethge Attack (2019)

- **Idea:**
  - to initialize the attack from an adversarial point
  - to find boundary between $\boldsymbol{x}$ and the init point (via line search)
  - to try following the boundary to minimize the perturbation size






Brendel et al., Accurate, Reliable and Fast robustness Evaluation, NeurIPS 2019

# Fast Adaptive Boundary (FAB) Attack (2020)

- FAB uses essentially the same approximation of DeepFool to estimate distance to boundary
- However
  - it improves the projection, also accounting for the presence of the box constraint
  - and uses momentum to accelerate convergence

**Algorithm 1** FAB-attack

**Input :** $x_{\text{orig}}$ original point, $c$ original class, $N_{\text{restarts}}, N_{\text{iter}}, \alpha_{\max}, \beta, \eta, \epsilon, p$

**Output :** $x_{\text{out}}$ adversarial example

$u \leftarrow +\infty$

**for** $j = 1, \dots, N_{restarts}$ **do**

  **if** $j = 1$ **then** $x^{(0)} \leftarrow x_{\text{orig}}$;

  **else** $x^{(0)} \leftarrow$ randomly sampled s.th. $\left\|x^{(0)} - x_{\text{orig}}\right\|_p = {}^{\min\{u,\epsilon\}}/_2$;

  **for** $i = 0, \dots, N_{iter} - 1$ **do**

    $s \leftarrow \arg\min_{l \neq c} \frac{|f_l(x^{(i)}) - f_c(x^{(i)})|}{\left\|\nabla f_l(x^{(i)}) - \nabla f_c(x^{(i)})\right\|_q}$

    $\delta^{(i)} \leftarrow \text{proj}_p(x^{(i)}, \pi_s, C)$

    $\delta^{(i)}_{\text{orig}} \leftarrow \text{proj}_p(x_{\text{orig}}, \pi_s, C)$

    compute $\alpha$ as in Equation (9)

    $x^{(i+1)} \leftarrow \text{proj}_C\Big((1-\alpha)\left(x^{(i)} + \eta\delta^{(i)}\right) + \alpha(x_{\text{orig}} + \eta\delta^{(i)}_{\text{orig}})\Big)$

    **if** $x^{(i+1)}$ *is not classified as* $c$ **then**

      **if** $\left\|x^{(i+1)} - x_{orig}\right\|_p < u$ **then**

        $x_{\text{out}} \leftarrow x^{(i+1)}$

        $u \leftarrow \left\|x^{(i+1)} - x_{\text{orig}}\right\|_p$

      **end**

      $x^{(i+1)} \leftarrow (1-\beta)x_{\text{orig}} + \beta x^{(i+1)}$

    **end**

  **end**

**end**

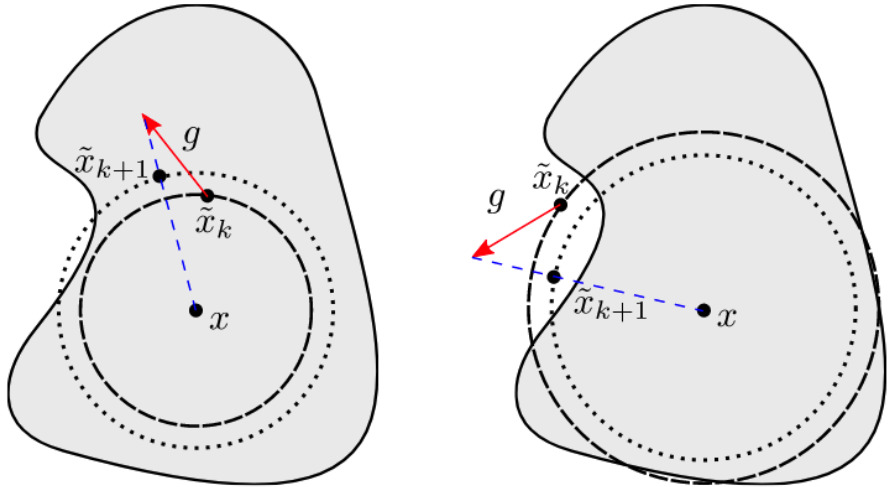
perform 3 steps of final search on $x_{\text{out}}$ as in (13)



Croce & Hein, Minimally distorted Adversarial Examples with …, ICML 2020

# Decoupled Direction and Norm (DDN) Attack (2019)

- DDN works in two steps
  - it performs a PGD-step
  - it adjusts the maximum perturbation size
- It uses *cosine annealing* as a decay strategy for the step size
- Specialized to $L_2$, but very fast and effective



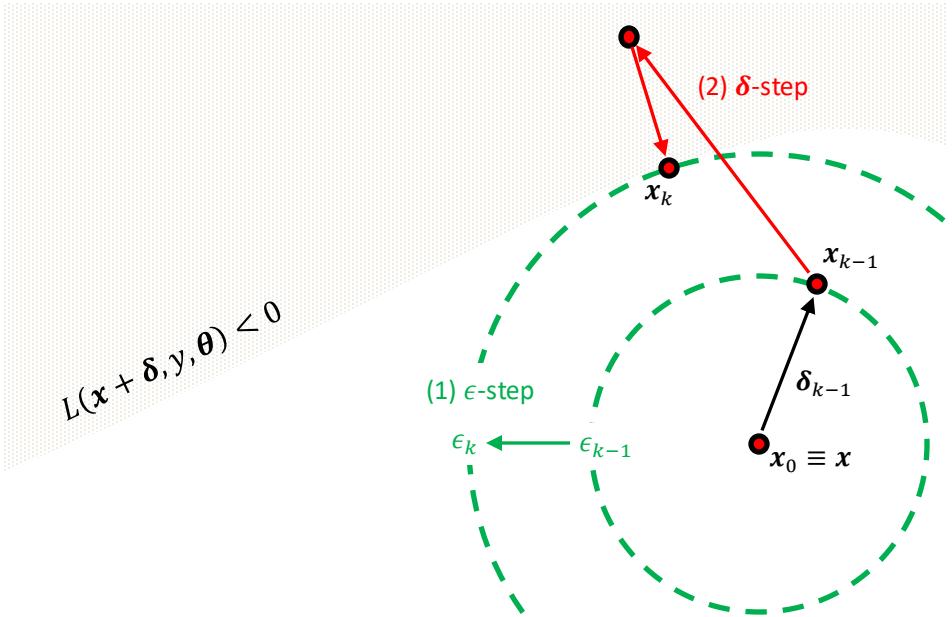

(a) $\tilde{x}_k$ not adversarial (b) $\tilde{x}_k$ adversarial



Rony et al., Decoupling Direction and Norm …, CVPR 2019

# Fast Minimum-Norm (FMN) Attacks (2021)

Biggio et al., 2013
Szegedy et al., 2014
Goodfellow et al., 2015 (FGSM)
Papernot et al., 2015 (JSMA)
Carlini & Wagner, 2017 (CW)
Madry et al., 2017 (PGD)
...
*Croce et al., FAB, AutoPGD ...*
*Rony et al., DDN, ALMA, ...*
***Pintor et al., 2021 (FMN)***




**FMN**

Fast convergence to good local optima

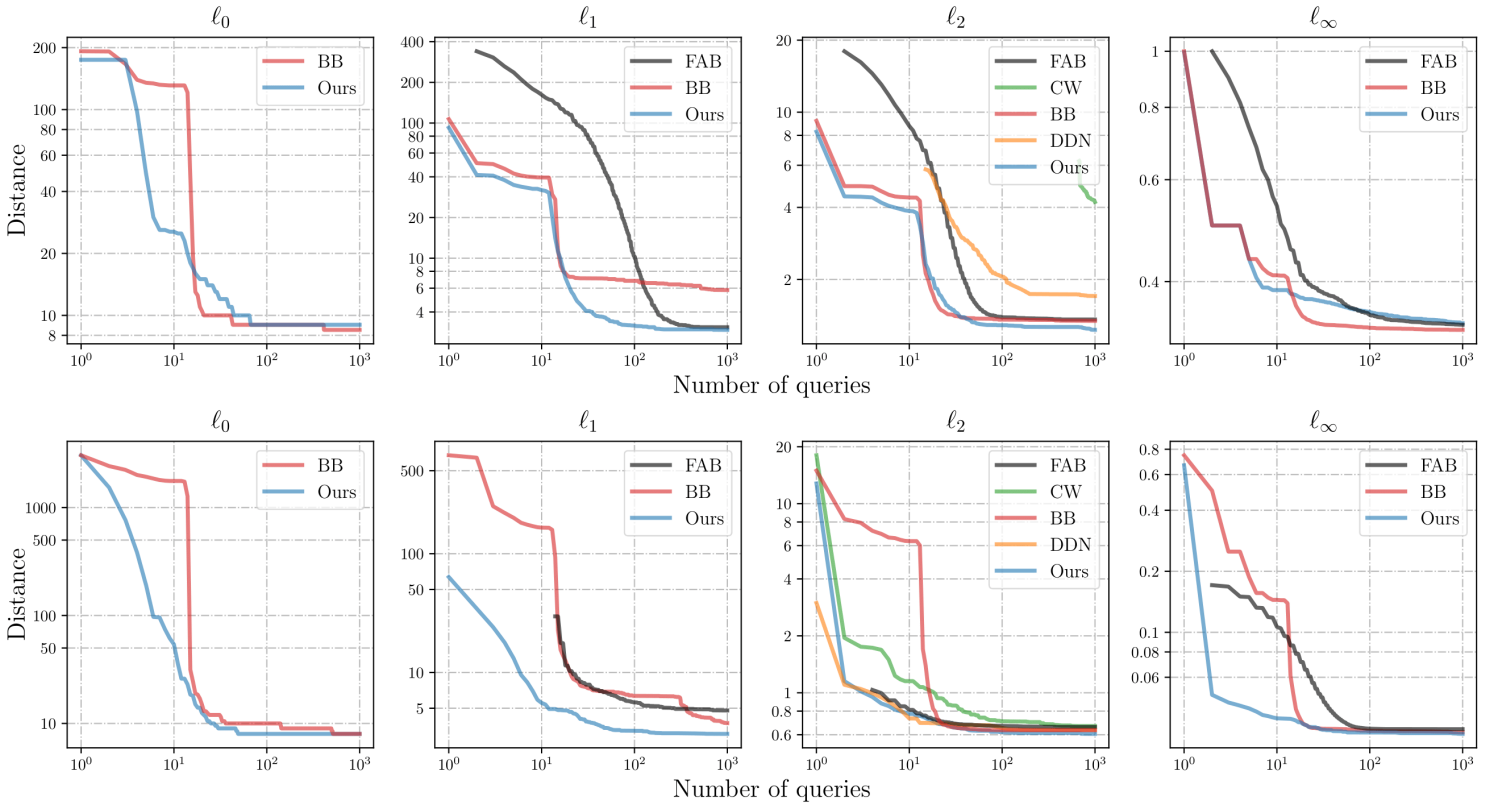
Works in different norms ($\ell_0, \ell_1, \ell_2, \ell_\infty$)

Easy tuning /robust to hyperparameter choice

# Experimental Results: Query-distortion Curves

MNIST challenge

CIFAR challenge

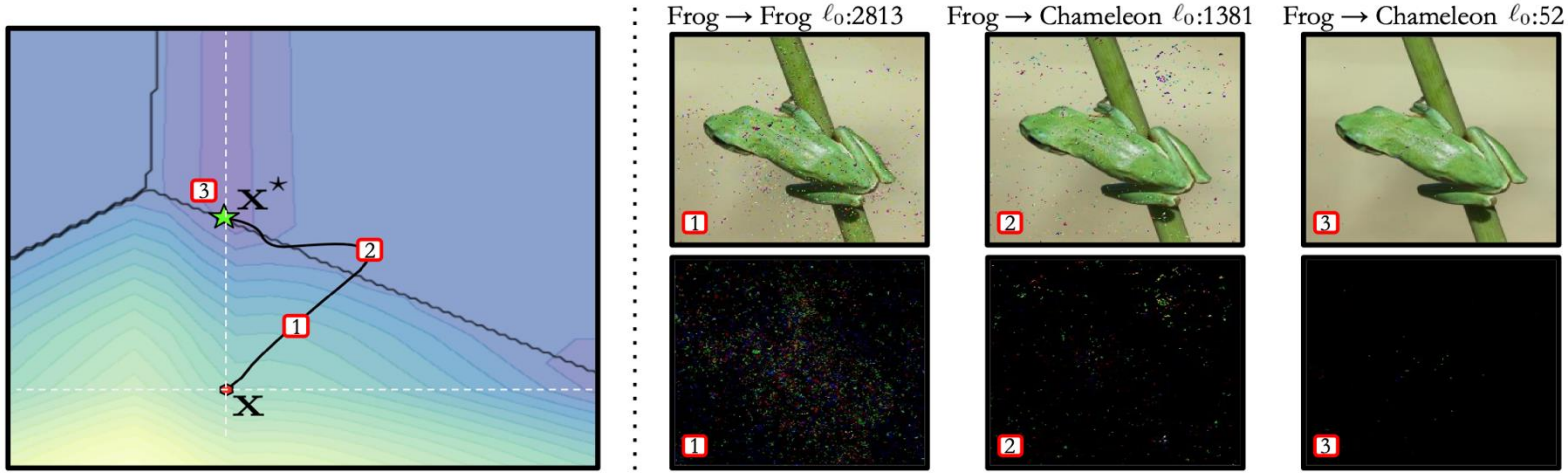





Pintor, Brendel, Roli, Biggio, *Fast minimum-norm adversarial attacks…*, NeurIPS 2021

# Optimization of Sparse Attacks

- **sigma-zero:** novel state-of-the-art attack in L0 (sparse perturbations)
  - Smooth L0-norm approximation and dynamic thresholding mechanism to retain high sparsity
  - https://arxiv.org/abs/2402.01879



# Benchmarks

# Automation with AutoAttack (AA)

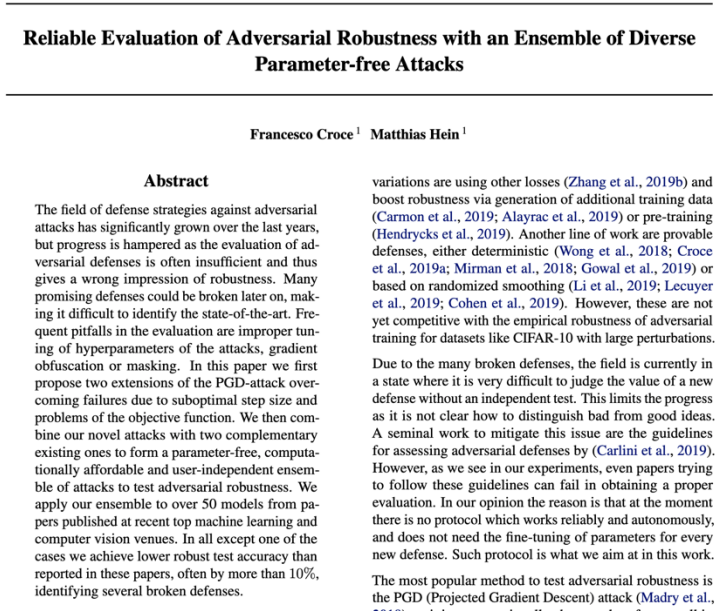
Automatic evaluation of a model by ensembling attacks:

- AutoPGD-*CE*
- AutoPGD-*DLR*
- FAB
- Square Attack (*black-box*)

(AA also plays with initialization, repetitions, etc.)

Latest development: *Adaptive AutoAttack*

- Yao et al., NeurIPS '21 (https://arxiv.org/abs/2102.11860)



**Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks**

**Francesco Croce** [1] **Matthias Hein** [1]

**Abstract**

The field of defense strategies against adversarial attacks has significantly grown over the last years, but progress is hampered as the evaluation of adversarial defenses is often insufficient and thus gives a wrong impression of robustness. Many promising defenses could be broken later on, making it difficult to identify the state-of-the-art. Frequent pitfalls in the evaluation are improper tuning of hyperparameters of the attacks, gradient obfuscation or masking. In this paper we first propose two extensions of the PGD-attack overcoming failures due to suboptimal step size and problems of the objective function. We then combine our novel attacks with two complementary existing ones to form a parameter-free, computationally affordable and user-independent ensemble of attacks to test adversarial robustness. We apply our ensemble to over 50 models from papers published at recent top machine learning and computer vision venues. In all except one of the cases we achieve lower robust test accuracy than reported in these papers, often by more than 10%, identifying several broken defenses.

variations are using other losses (Zhang et al., 2019b) and boost robustness via generation of additional training data (Carmon et al., 2019; Alayrac et al., 2019) or pre-training (Hendrycks et al., 2019). Another line of work are provable defenses, either deterministic (Wong et al., 2018; Croce et al., 2019a; Mirman et al., 2018; Gowal et al., 2019) or based on randomized smoothing (Li et al., 2019; Lecuyer et al., 2019; Cohen et al., 2019). However, these are not yet competitive with the empirical robustness of adversarial training for datasets like CIFAR-10 with large perturbations.

Due to the many broken defenses, the field is currently in a state where it is very difficult to judge the value of a new defense without an independent test. This limits the progress as it is not clear how to distinguish bad from good ideas. A seminal work to mitigate this issue are the guidelines for assessing adversarial defenses by (Carlini et al., 2019). However, as we see in our experiments, even papers trying to follow these guidelines can fail in obtaining a proper evaluation. In our opinion the reason is that at the moment there is no protocol which works reliably and autonomously, and does not need the fine-tuning of parameters for every new defense. Such protocol is what we aim at in this work.

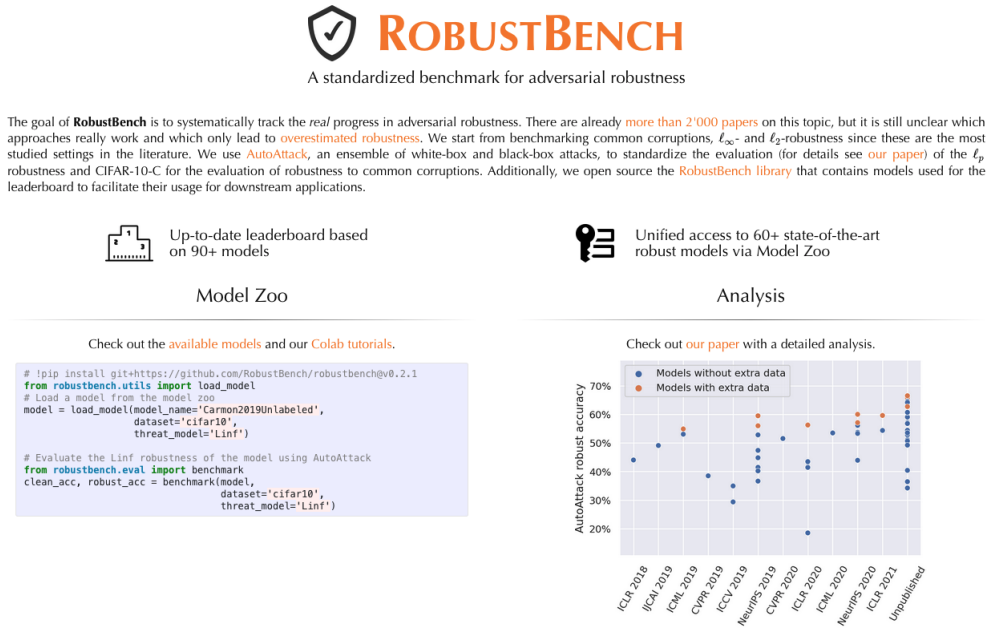
The most popular method to test adversarial robustness is the PGD (Projected Gradient Descent) attack (Madry et al.,

# Current Benchmark for Vision Models: *RobustBench*

Public benchmark for defenses
https://robustbench.github.io/

It uses AutoAttack, defenses are listed in leaderboard by results

All models are available and can be tested offline



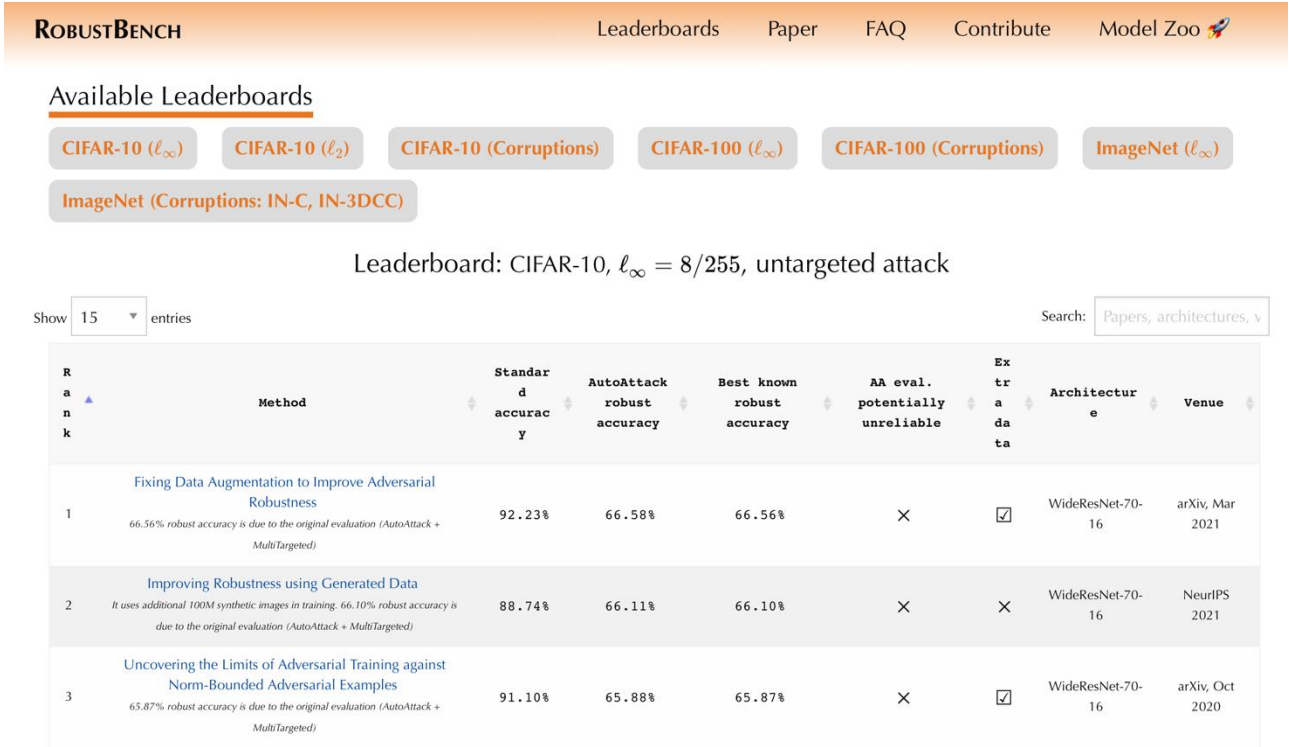
# Current Benchmark for Vision Models: *RobustBench*

*Pros*

- First standardized benchmark
- Easy to use (AutoAttack provides a good combination of attacks)

*Cons*

- Evalutes RA at fixed perturbation budget
- Computationally demanding (ensembling four attacks)

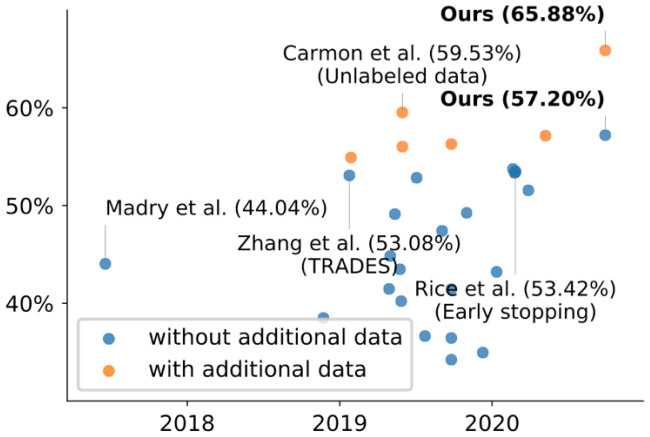



| Rank | Method | Standard accuracy | AutoAttack robust accuracy | Best known robust accuracy | AA eval. potentially unreliable | Extra data | Architecture | Venue |
|---|---|---|---|---|---|---|---|---|
| 1 | Fixing Data Augmentation to Improve Adversarial Robustness<br>*66.56% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)* | 92.23% | 66.58% | 66.56% | ✗ | ☑ | WideResNet-70-16 | arXiv, Mar 2021 |
| 2 | Improving Robustness using Generated Data<br>*It uses additional 100M synthetic images in training. 66.10% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)* | 88.74% | 66.11% | 66.10% | ✗ | ✗ | WideResNet-70-16 | NeurIPS 2021 |
| 3 | Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples<br>*65.87% robust accuracy is due to the original evaluation (AutoAttack + MultiTargeted)* | 91.10% | 65.88% | 65.87% | ✗ | ☑ | WideResNet-70-16 | arXiv, Oct 2020 |

# Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples

- Sven Gowal (Deepmind) et al., 2021
  https://arxiv.org/abs/2010.03593

*We discover that it is possible to train robust models that go well beyond state-of-the-art results by combining larger models, Swish/SiLU activations and model weight averaging.*




**Extracted from RobustBench:**
https://robustbench.github.io/

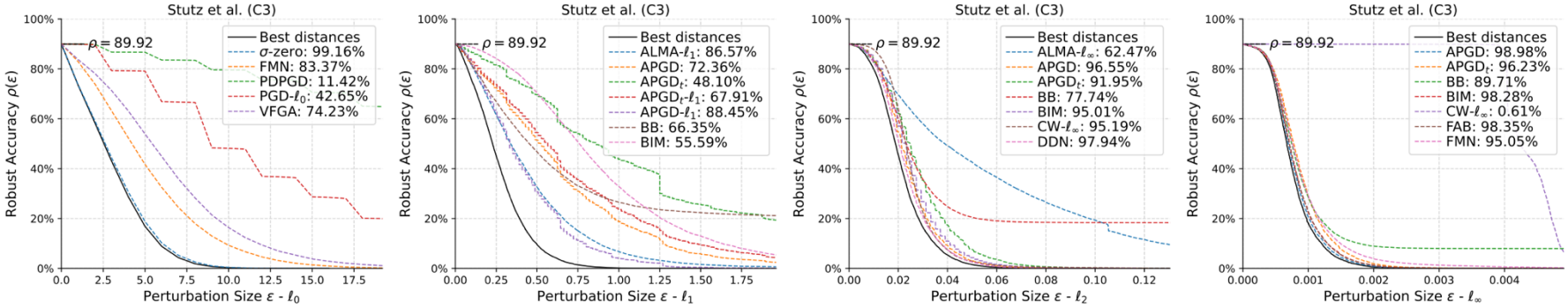
# Attack Implementations

- Most popular Python libraries implementing attacks
  - we will use *secml* in this course




# Benchmarking Gradient-based Attacks

- **AttackBench**: Benchmarked *all* gradient-based attack implementations (more than 100 attack implementations, ~1,000 different configurations tested) in terms of optimality/success and efficiency/complexity
  - https://attackbench.github.io
  - https://arxiv.org/abs/2404.19460

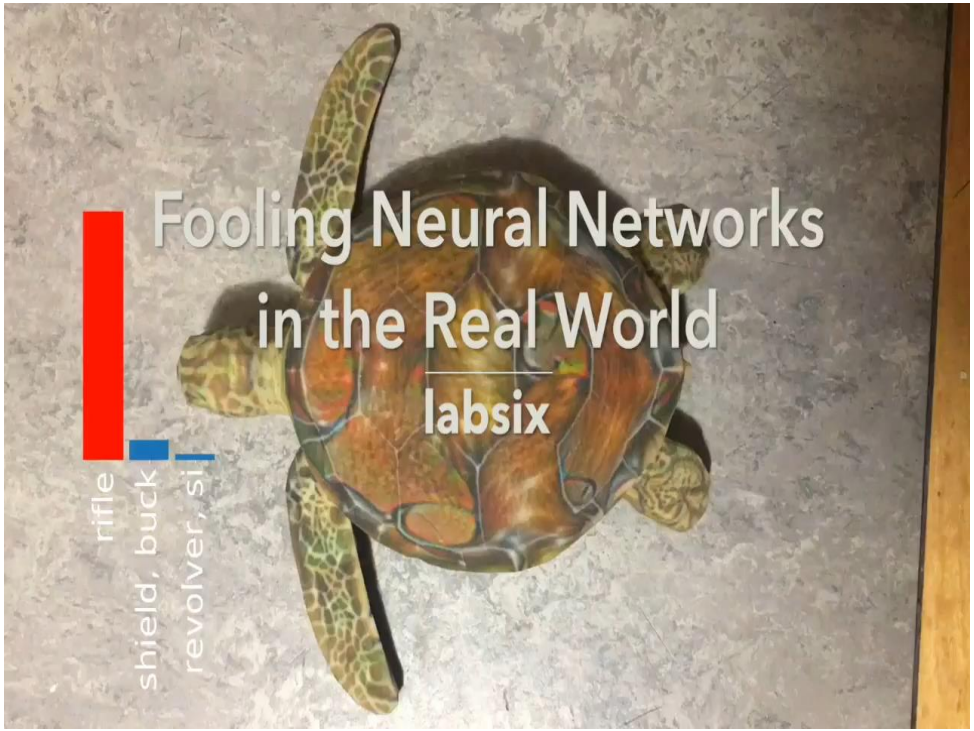


# Physical Attacks: *EoT* and Adversarial Patches

# Adversarial Examples in the Physical World

- Do adversarial images fool deep networks **even when** they **operate** in the **physical world**, for example, **images** are **taken from** a **cell-phone camera**?
  - Alexey Kurakin et al. (2016, 2017) explored the possibility of creating adversarial images for machine learning systems which operate in the physical world. They used images taken from a cell-phone camera as an input to an Inception v3 image classification neural network
  - They showed that in such a set-up, a significant fraction of adversarial images crafted using the original network are misclassified even when fed to the classifier through the camera

# Adversarial Examples in the Physical World






Athalye et al., Synthesizing robust adversarial examples. ICML, 2018

# Adversarial Examples in the Physical World

## Robust Physical-World Attacks on Machine Learning Models

Visit https://iotsecurity.eecs.umich.edu/#roadsigns for an FAQ

Ivan Evtimov[1], Kevin Eykholt[2], Earlence Fernandes[1], Tadayoshi Kohno[1],
Bo Li[4], Atul Prakash[2], Amir Rahmati[3], and Dawn Song*[4]

[1]University of Washington
[2]University of Michigan Ann Arbor
[3]Stony Brook University
[4]University of California, Berkeley

# How Are They Optimized?

- The constraint for the attacker here is to craft a perturbation which is **robust** to changes in *illumination*, *pose*, *distance to the camera*, etc.
  - How is this achieved?

- The previous examples all make use of the notion of **EoT:** Expectation over Transformations, originally proposed by Athalye et al., ICML 2018 https://arxiv.org/pdf/1707.07397.pdf
  - **Main idea:** to optimize the perturbation to be invariant to different image transformations

$$\begin{aligned} \arg\max_{x'} \quad & \mathbb{E}_{t\sim T}[\log P(y_t|t(x'))] \\ \text{subject to} \quad & \mathbb{E}_{t\sim T}[d(t(x'), t(x))] < \epsilon \\ & x \in [0,1]^d \end{aligned}$$

# Adversarial Patch

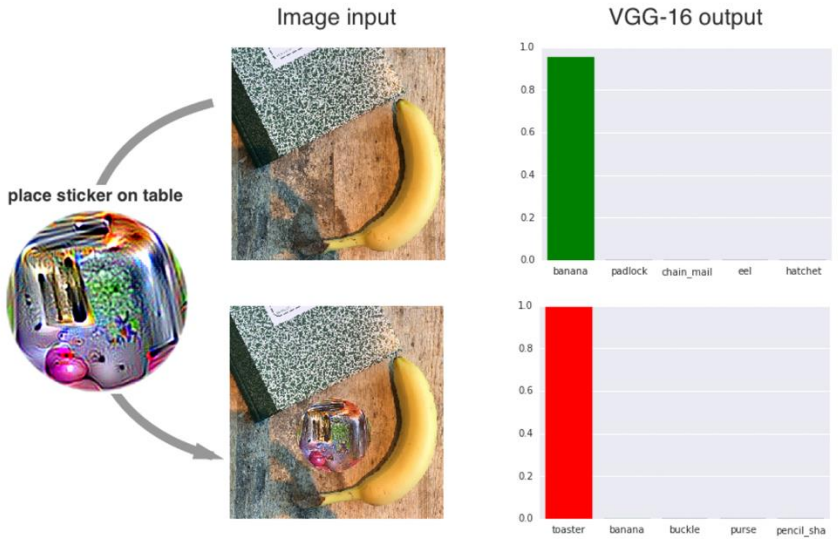
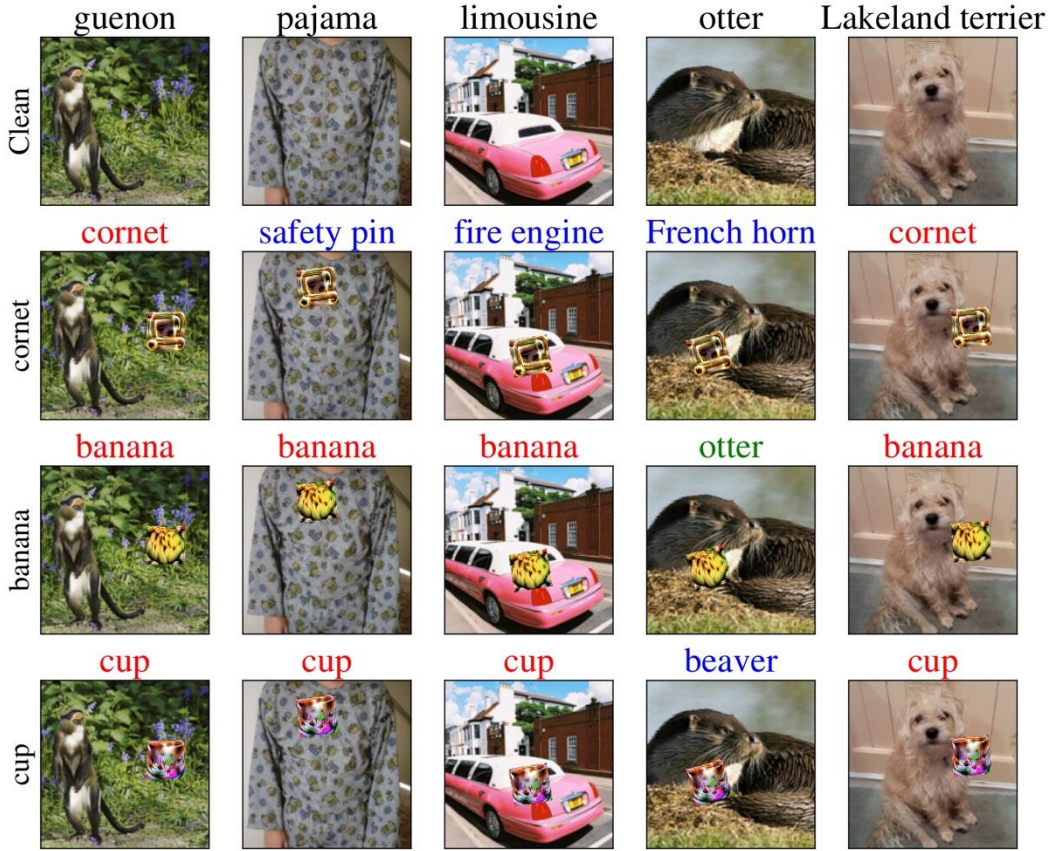



Figure 1: A real-world attack on VGG16, using a physical patch generated by the white-box ensemble method described in Section 3. When a photo of a tabletop with a banana and a notebook (top photograph) is passed through VGG16, the network reports class 'banana' with 97% confidence (top plot). If we physically place a sticker targeted to the class "toaster" on the table (bottom photograph), the photograph is classified as a toaster with 99% confidence (bottom plot). See the following video for a full demonstration: `https://youtu.be/e9IAu4lT9w8`

# ImageNet Patch

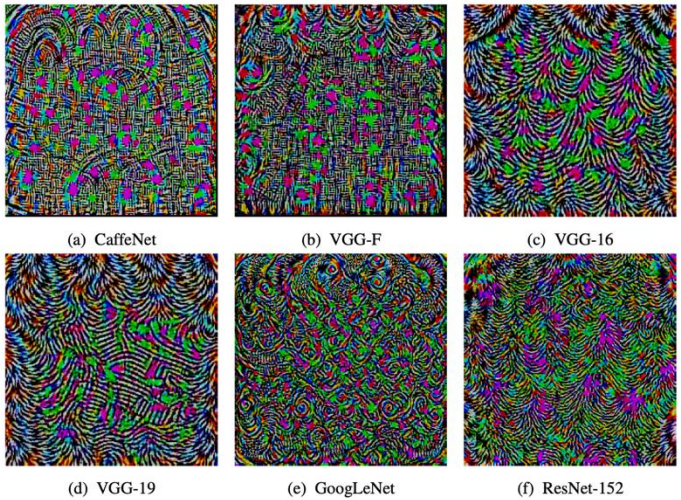



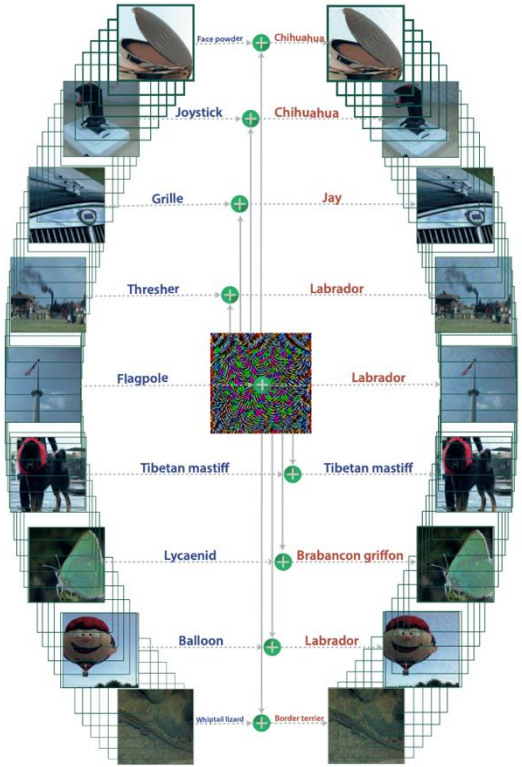



M. Pintor et al., ImageNet-Patch: A Dataset for ..., Patt. Rec. 2022

# Universal Adversarial Perturbations (UAP)

- **Same principle of EoT**: optimizing the perturbation over different images
  - However, this attack just considers images from different classes (and not different views of the same object)






S. Moosavi-Dezfooli et al., https://arxiv.org/abs/1610.08401

# Black-box (Gradient-free) Evasion Attacks

# Motivations

Model might be non-differentiable (e.g. Random Forest classifiers)

Target is unavailable, like "Machine Learning as a Service" (MLaaS), available only through APIs

**No gradients can be computed in these scenarios, Black-box attacks are needed!**

$$\nabla_x L(x, y; \theta)$$

# (1) Black-box *Transfer* Attacks

**Attack surrogate classifier**

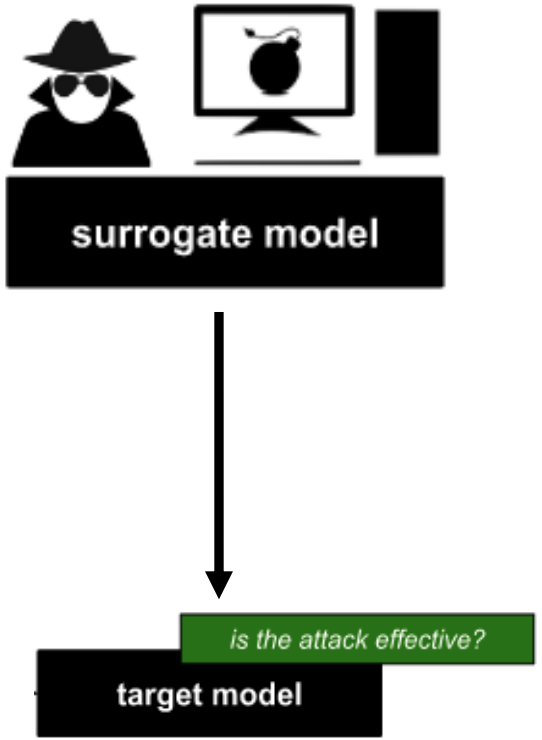
Consider a close approximation of the real target model, by either training a new one on same or similar data, or use a pretrained model

**Compute gradient attacks**

The attacker chose a differentiable model, to maximize the easiness of computing attacks

**Transfer the results**

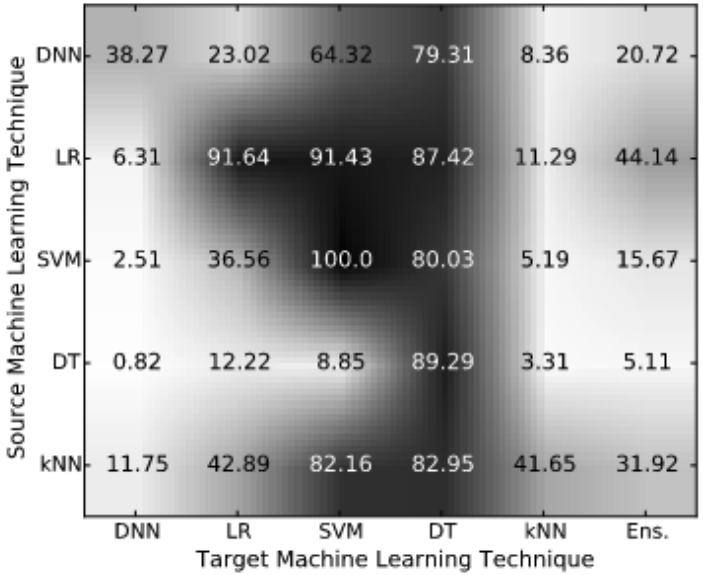
Try to evade the real target using the adversarial examples computed on the surrogate



# Surrogate models

**Competition between models**
Training different surrogates and compute how they transfer "all vs all"

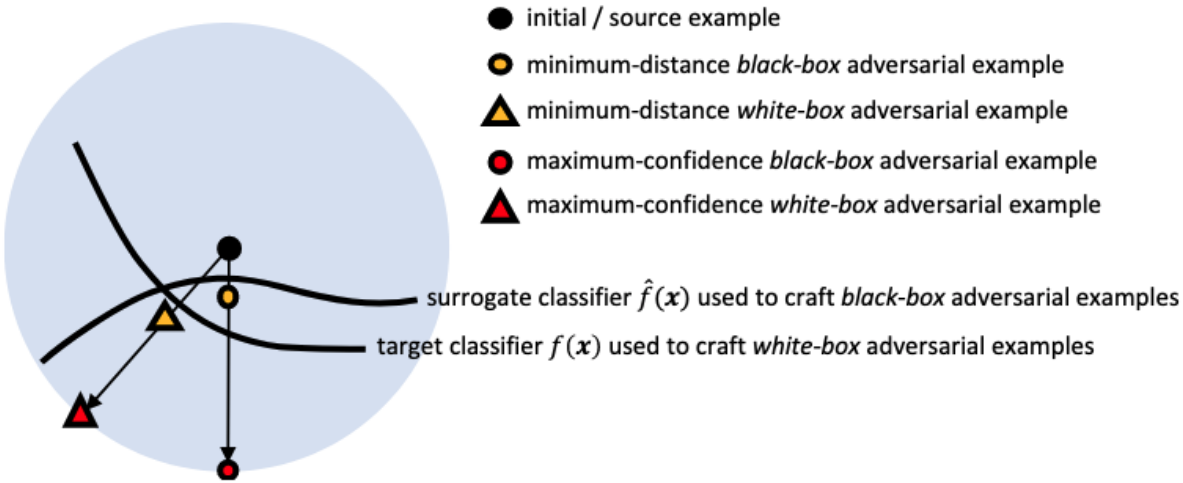
**Different techniques have different results**
The heatmap shows that different models behave differently when tested with attacks optimized on other models





Papernot et al., Practical Black-box Attacks against Machine Learning, AsiaCCS 2017

# Do Transfer Attacks Work?

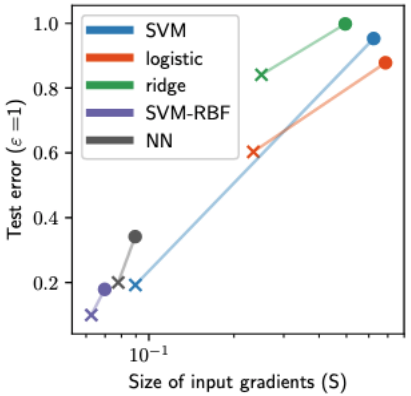


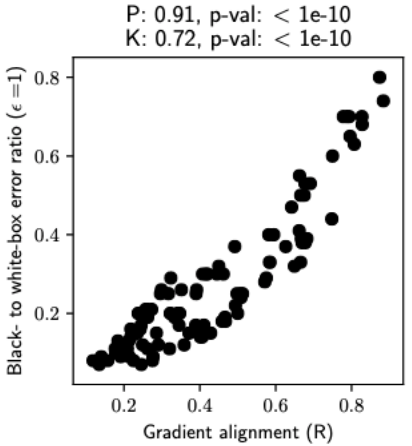

Maximum confidence attacks might better transfer, but more perturbation is needed

Minimum distance attacks are likely to stop working because decision boundary is different

# Quantifying Transferability through Metrics

$$T = \ell(y, \mathbf{x} + \hat{\boldsymbol{\delta}}, \mathbf{w}) \approxeq \ell(y, \mathbf{x}, \mathbf{w}) + \hat{\boldsymbol{\delta}}^{\top} \nabla_{\mathbf{x}} \ell(y, \mathbf{x}, \mathbf{w})$$




**Main Insight:** by looking at the size of gradients, the gradient alignment, and the variability of the loss function, it is possible to understand if a model will suffer from transfer attacks

# Recap

## Benefits

No white-box access to target model

Depending on the domain, data and similar pretrained model are already available

## Issue: to build a good surrogate model

Training the surrogate might lead to training errors, and the attack might not transfer since the approximation is not good enough

Data might be unavailable as well

Might require the attacker to interact with the target to extract labels (to be used at training time)

# (2) Black-box *Query* Attacks

**No need for training a model from scratch**

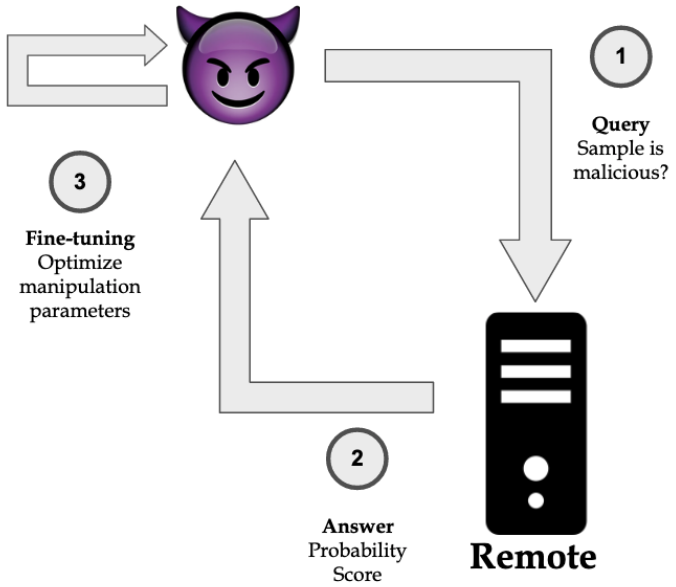
The attacker can query the target (that might be on a remote server), no need for looking for data, pretrained models, etc.

**Optimize attack based on feedback**

The attack is optimized based on the classifier's output/prediction

- *Hard-label* vs *soft-label* attacks

**Main challenge:** keeping a small number of queries to stay undetected



# ZOO: Zeroth order attack

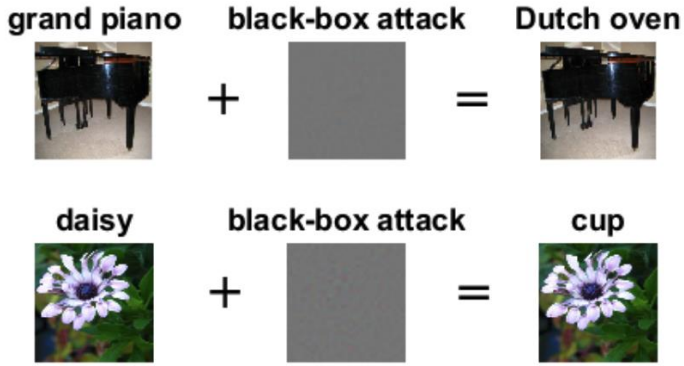
**No surrogate model**
Estimate Hessian (2° order derivative) in each direction, by querying the model locally and around a very small proximity by chosing random directions, no need to train surrogate

$$\hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h},$$

$$\hat{h}_i := \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_{ii}^2} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x} - h\mathbf{e}_i)}{h^2}.$$

**Sparse result**

Attack only uses randomly picked directions to minimize both queries and perturbation size





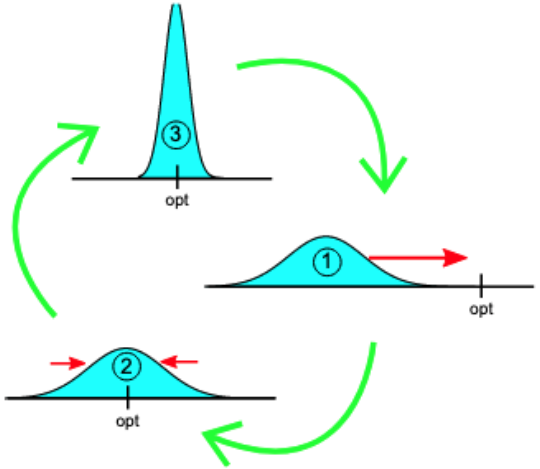

[Chen et al., ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models, AISec 2017]

# Natural Evolutionary Strategies (NES)

- Assume **Normally-distributed** inputs
- Estimate **gradient** w.r.t. distribution parameters (mean, std)

$$J(\theta) = \mathbb{E}_\theta[f(\mathbf{z})] = \int f(\mathbf{z})\ \pi(\mathbf{z}\,|\,\theta)\ d\mathbf{z}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda}\sum_{k=1}^{\lambda} f(\mathbf{z}_k)\ \nabla_\theta \log \pi(\mathbf{z}_k\,|\,\theta)$$




- Use the **natural** gradient instead of the plain gradient
  - **F** is the Fisher information matrix
  - Less dependent on the distribution choice (more robust/trustworthy direction)

$$\widetilde{\nabla}_\theta J = \mathbf{F}^{-1}\nabla_\theta J(\theta)$$

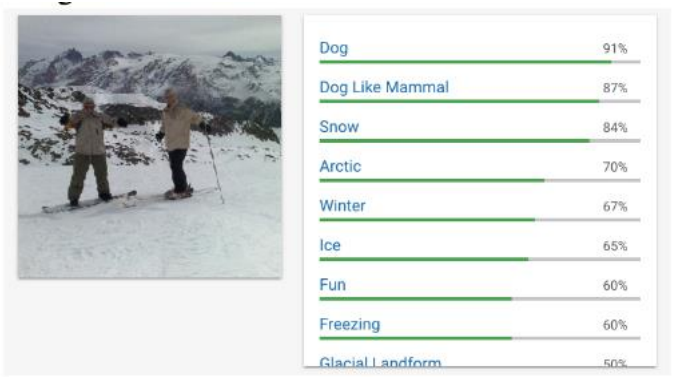


Wiestra et al. Natural evolution strategies, JMLR 2014

# Natural Evolutionary Strategies (NES)

**Tested also against MLaaS**

Effective in real case scenarios, attacking Google Cloud Vision



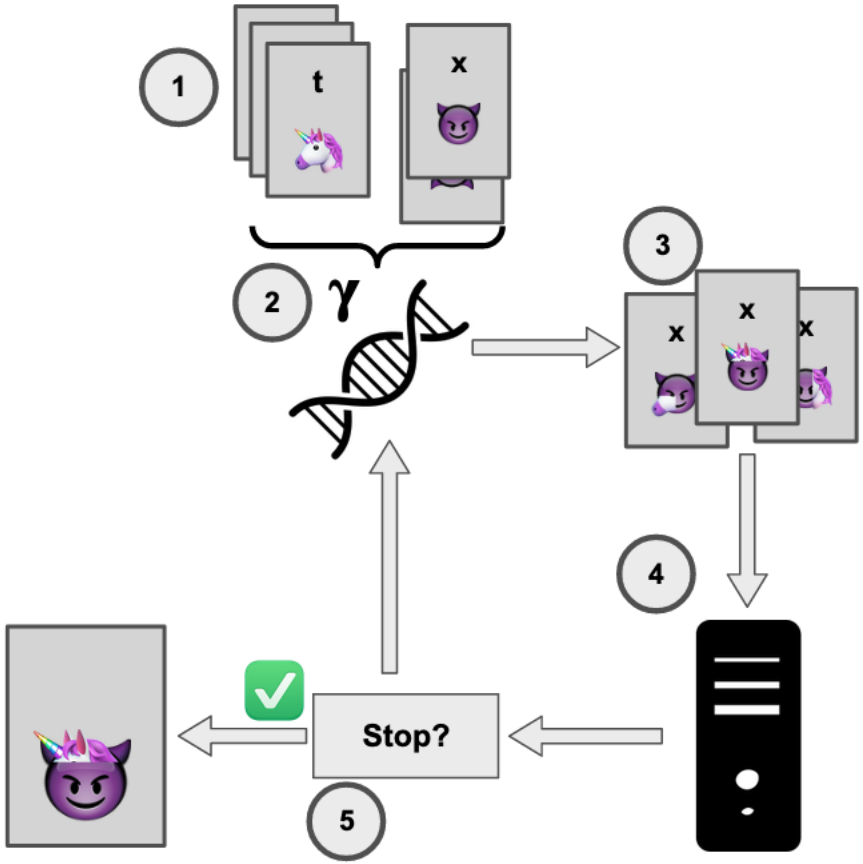



Ilyas et al. Black-box Adversarial Attacks with Limited Queries …, ICML 2018

# Genetic Algorithms

**Evolving solutions**
Sample N points, compute scores, retain best candidates. Follow "genetic evolution" until a suitable solution is found

**No gradient estimation**
No need for computing approximation of best direction, it is taken care of by the mixing of "genes"



# Recap

## Benefits

Bounded number of queries might already be enough for evasion

Ready to target real world targets as MLaaS

A lot of different methods to estimate directions

## Issues

Decision are local, slower than normal gradient-based attack as it is reconstructing best direction from answers

The number of needed queries might still be enormous, depending on the robustness of the target

# Countering Evasion Attacks


What is the rule? The rule is protect yourself at all times (from the movie “Million dollar baby”, 2004)

# Security Measures for Machine Learning

B. Biggio, F. Roli / Pattern Recognition 84 (2018) 317–331

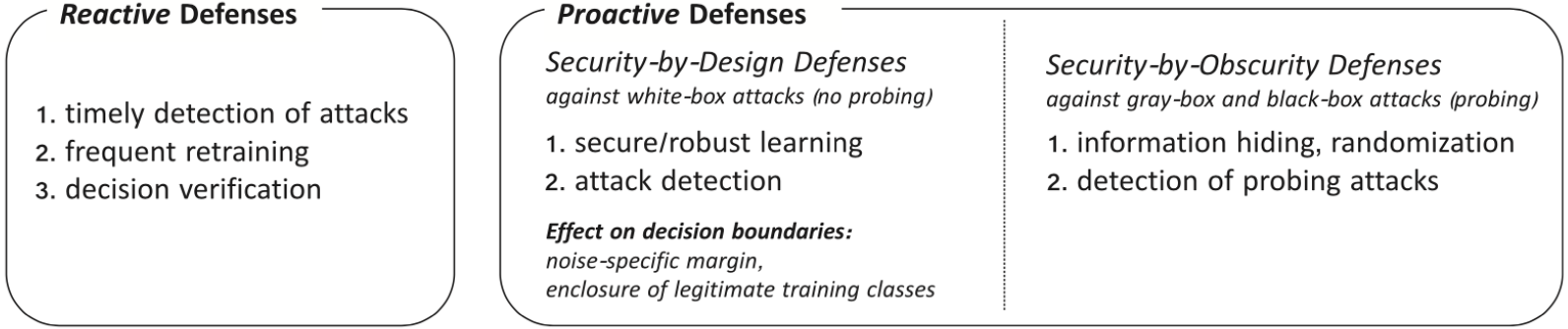


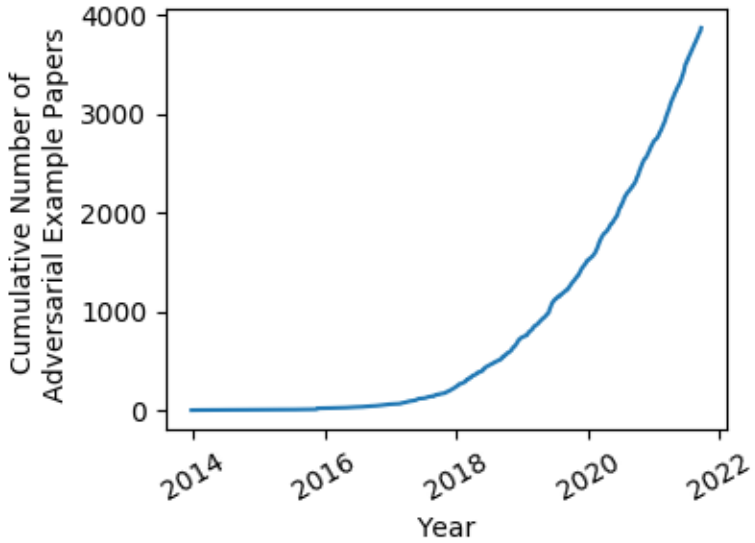

**Fig. 11.** Schematic categorization of the defense techniques discussed in Section 5.

Wild patterns: Ten years after the rise of adversarial machine learning

Battista Biggio[a,b,*], Fabio Roli[a,b]

# A Challenging Problem!

- https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html



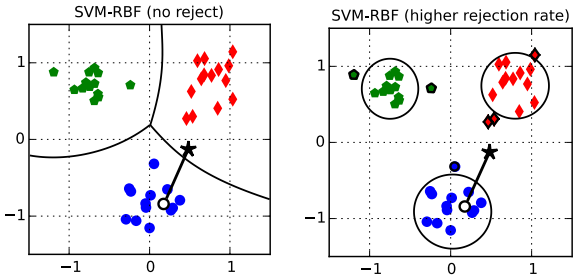
# Security Measures against Evasion Attacks

1. Reduce sensitivity to input changes with **robust optimization**
   - Adversarial Training / Regularization

$$\min_{\boldsymbol{w}} \sum_i \max_{||\boldsymbol{\delta}_i|| \leq \epsilon} \ell(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i))$$

**bounded perturbation!**

2. Introduce ***rejection / detection*** of adversarial examples



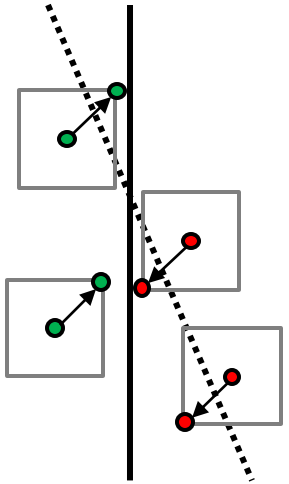
**Countering Evasion:**
***Robust Optimization / Adversarial Training***

# Robust Optimization via Adversarial Training (AT)

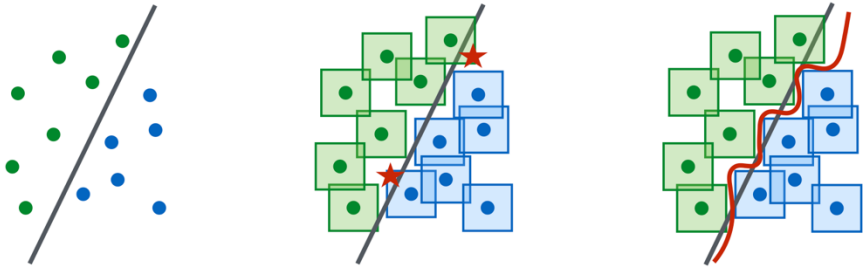
- Robust optimization (a.k.a. *adversarial training*)

$$\min_{\boldsymbol{w}} \max_{||\boldsymbol{\delta}_i||_\infty \leq \epsilon} \sum_i \ell\left(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i)\right)$$

**bounded perturbation!**



- Madry et al., ICLR 2018 (https://arxiv.org/pdf/1706.06083.pdf)
  - PGD-AT better than FGSM-AT but more computationally costly
  - Fast AT (NeurIPS 2020, https://arxiv.org/abs/2007.02617)

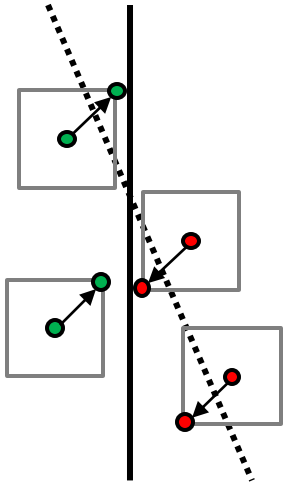


# Robust Optimization via Regularization

- Robust optimization (a.k.a. *adversarial training*)

$$\min_{\boldsymbol{w}} \max_{||\boldsymbol{\delta}_i||_\infty \leq \epsilon} \sum_i \ell\left(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i + \boldsymbol{\delta}_i)\right)$$

**bounded perturbation!**



- Robustness and regularization (Xu et al., JMLR 2009)
  - under linearity of $\ell$ and $f_{\boldsymbol{w}}$, equivalent to robust optimization

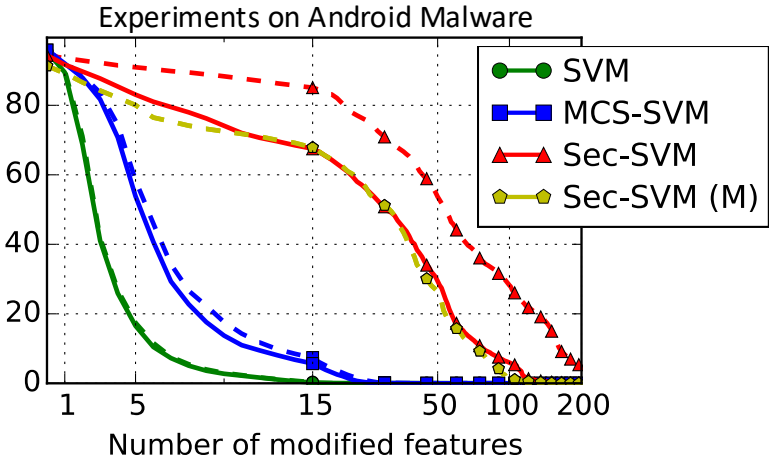
$$\min_{\boldsymbol{w}} \sum_{\boldsymbol{i}} \ell\left(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i)\right) + \epsilon||\nabla_x f||_1$$

**dual norm of the perturbation**
$||\nabla_x f||_1 = ||\boldsymbol{w}||_1$

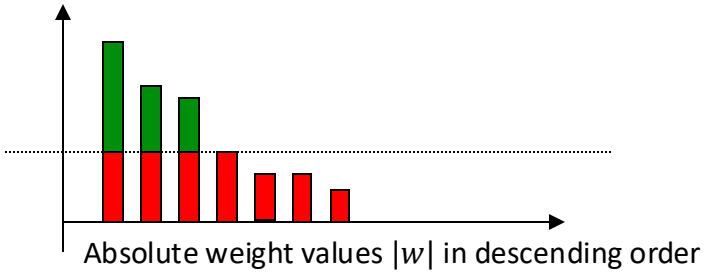
# Results on *Adversarial* Android Malware

- **Infinity-norm regularization** is the optimal regularizer against **sparse evasion attacks**
  - Sparse evasion attacks penalize $||\boldsymbol{\delta}||_1$ promoting the manipulation of only few features

**Sec-SVM** $$\min_{w,b} \|w\|_{¥} + C\sum_i \max(0, 1 - y_i f(x_i)), \quad \|w\|_{¥} = \max_{i=1,\ldots,d} |w_i|$$




***Why?*** It bounds the maximum weight absolute values!





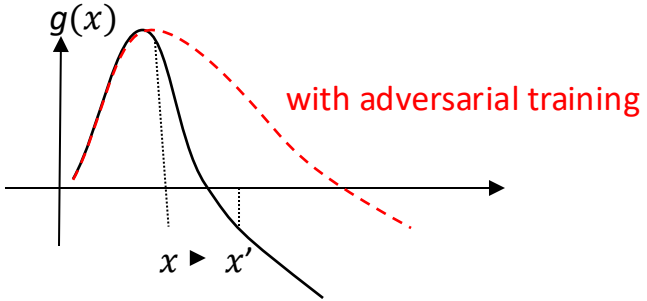

[Demontis, Biggio et al., Yes, ML Can Be More Secure!..., IEEE TDSC 2017]

# Adversarial Training and Regularization

- Adversarial training can also be seen as a form of regularization, which penalizes the (dual) norm of the input gradients $\epsilon||\nabla_{\mathbf{x}}\ell||_{\mathbf{q}}$

- Known as double backprop or gradient/Jacobian regularization
  - see, e.g., *Simon-Gabriel et al., Adversarial vulnerability of neural networks increases with input dimension, ArXiv 2018*; and *Lyu et al., A unified gradient regularization family for adversarial examples, ICDM 2015*.

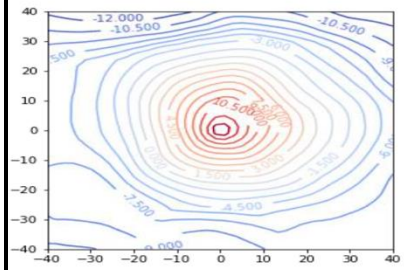
***Take-home message:*** the net effect of these techniques is to make the prediction function of the classifier smoother (increasing the input margin)

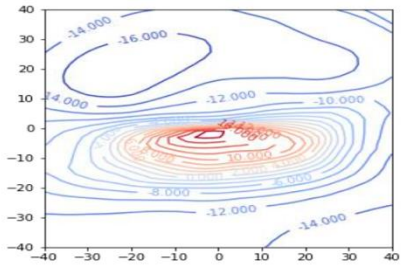


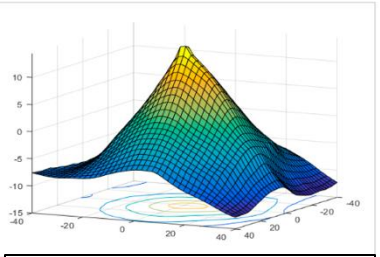
# Why Does Robust Optimization Work?

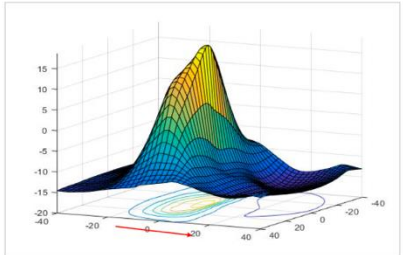
CIFAR-10

**Undefended model – Adversarial accuracy: 0.3%**





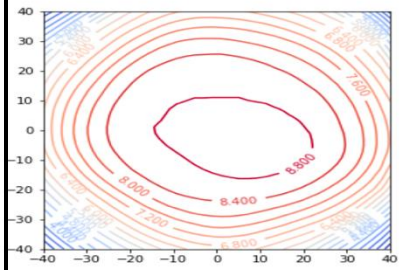


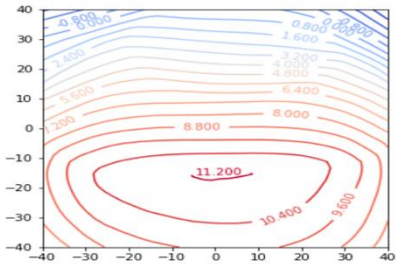


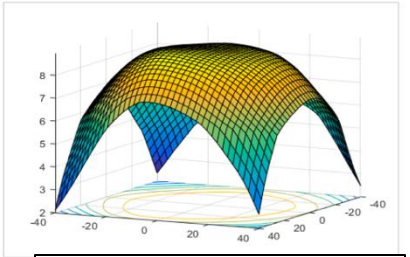
**random perturbation**

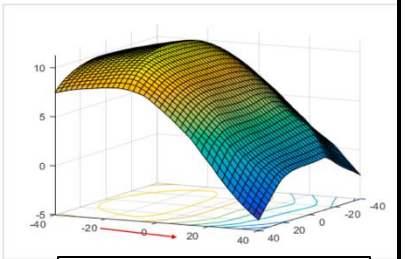
**adv. perturbation**

**Defended model – Adversarial accuracy: 44.7%**









**random perturbation**

**adv. perturbation**



Yu et al., Interpreting and Evaluating NN Robustness, IJCAI 2019

# On Adversarial Training...

2004

## Adversarial Classification

Nilesh Dalvi Pedro Domingos Mausam Sumit Sanghai Deepak Verma
Department of Computer Science and Engineering
University of Washington, Seattle
Seattle, WA 98195-2350, U.S.A.
{nilesh,pedrod,mausam,sanghai,deepak}@cs.washington.edu

2006

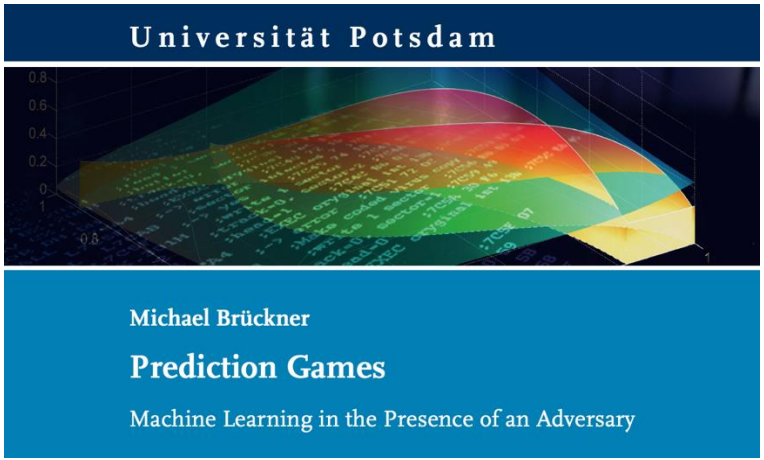
## Nightmare at Test Time: Robust Learning by Feature Deletion

**Amir Globerson** GAMIR@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA
**Sam Roweis** ROWEIS@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Canada

2012

## Static Prediction Games for Adversarial Learning Problems

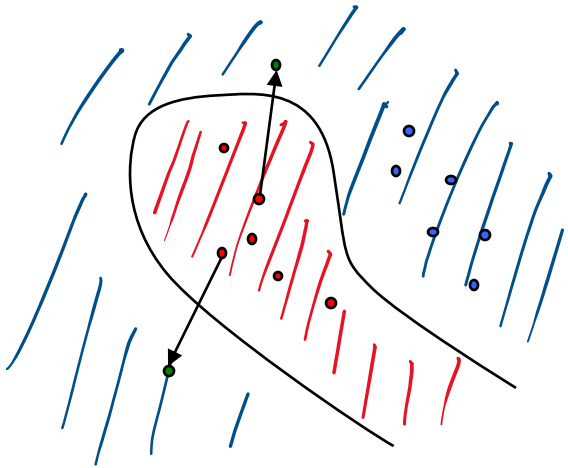
**Michael Brückner** MIBRUECK@CS.UNI-POTSDAM.DE
*Department of Computer Science*
*University of Potsdam*
*August-Bebel-Str. 89*
*14482 Potsdam, Germany*
**Christian Kanzow** KANZOW@MATHEMATIK.UNI-WUERZBURG.DE
*Institute of Mathematics*
*University of Würzburg*
*Emil-Fischer-Str. 30*
*97074 Würzburg, Germany*
**Tobias Scheffer** SCHEFFER@CS.UNI-POTSDAM.DE
*Department of Computer Science*
*University of Potsdam*
*August-Bebel-Str. 89*
*14482 Potsdam, Germany*



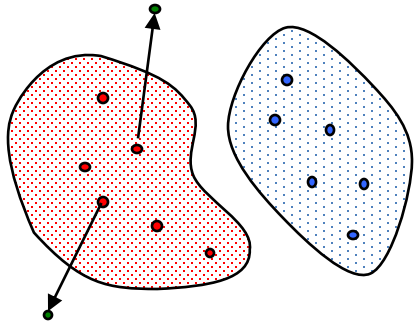
## Countering Evasion:
## *Detecting & Rejecting Adversarial Examples*

# Detecting and Rejecting Adversarial Examples

- Adversarial examples tend to occur in *blind spots*
  - Regions far from training data that are anyway assigned to 'legitimate' classes

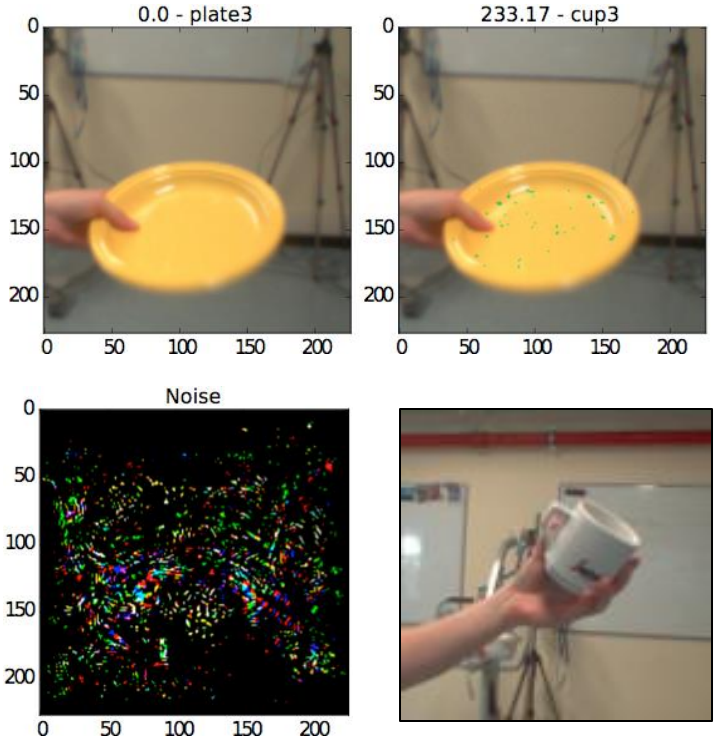


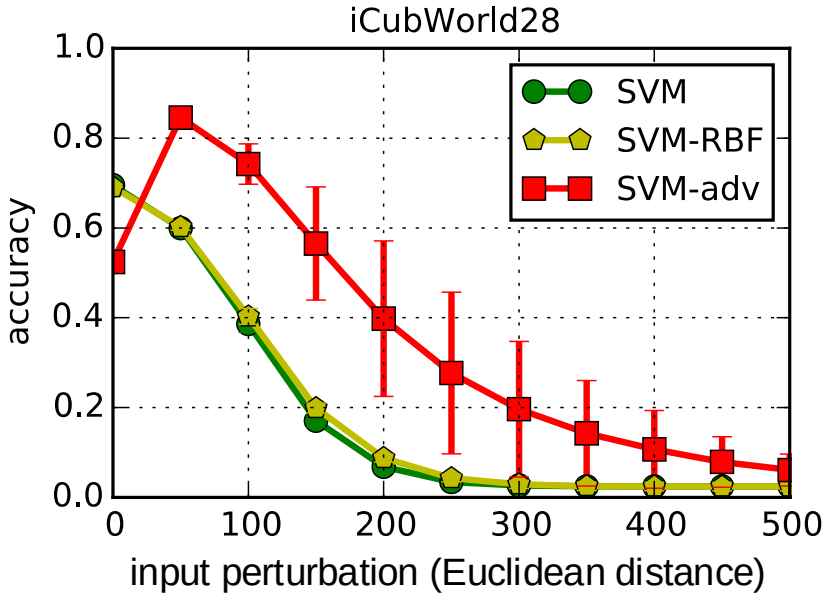
***blind-spot evasion***
(not even required to mimic the target class)



**rejection** of adversarial examples through enclosing of legitimate classes

# Detecting & Rejecting Adversarial Examples

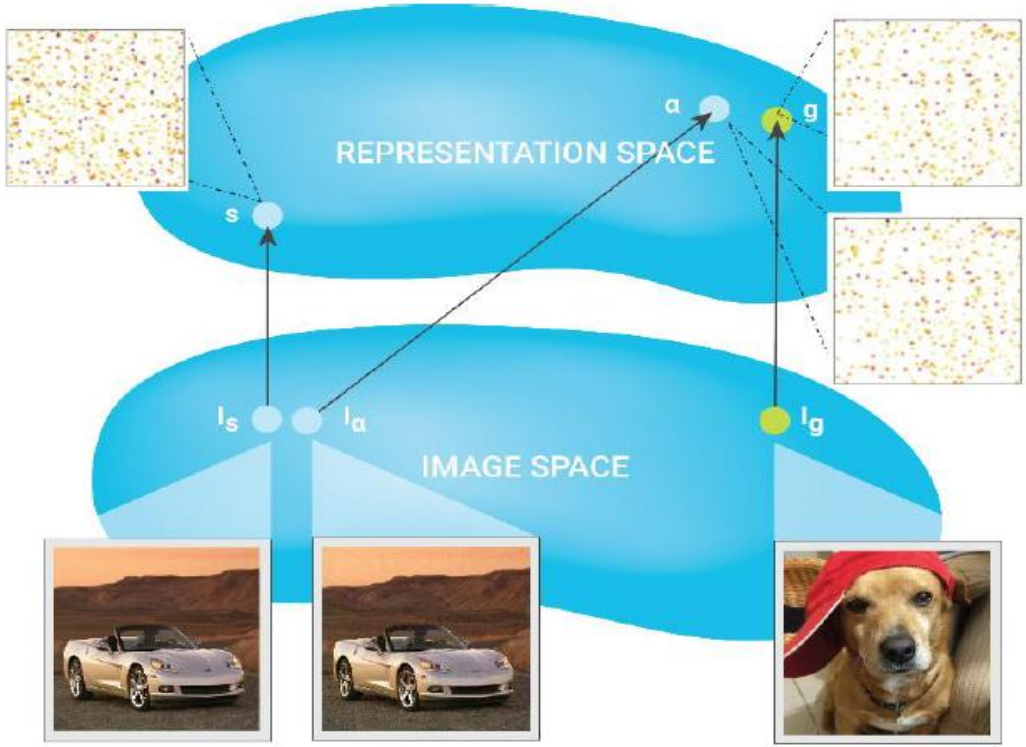





[Melis, Biggio et al., Is Deep Learning Safe for Robot Vision? ICCVW ViPAR 2017]

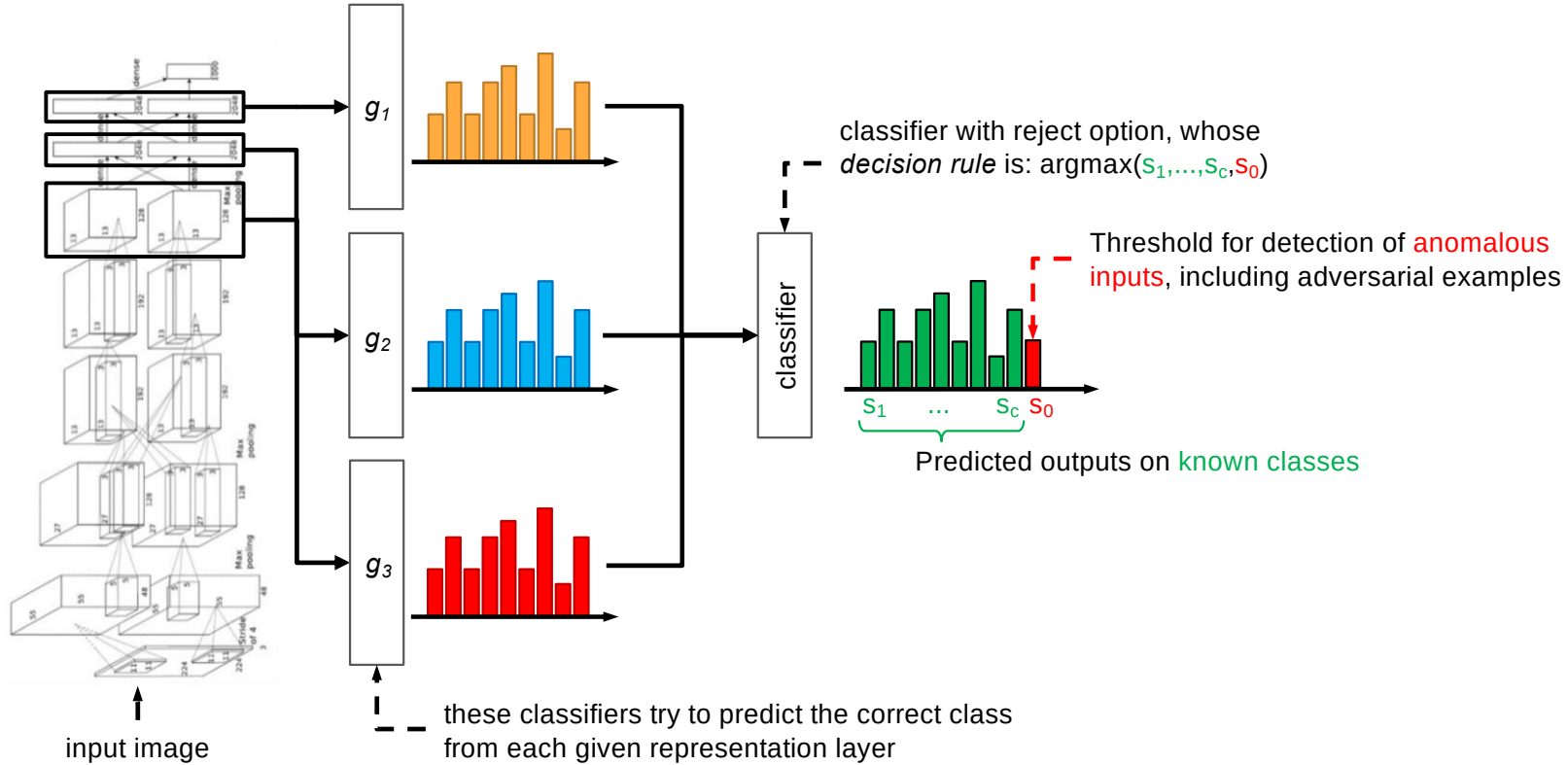
# Why Rejection (in Representation Space) Is Not Enough?



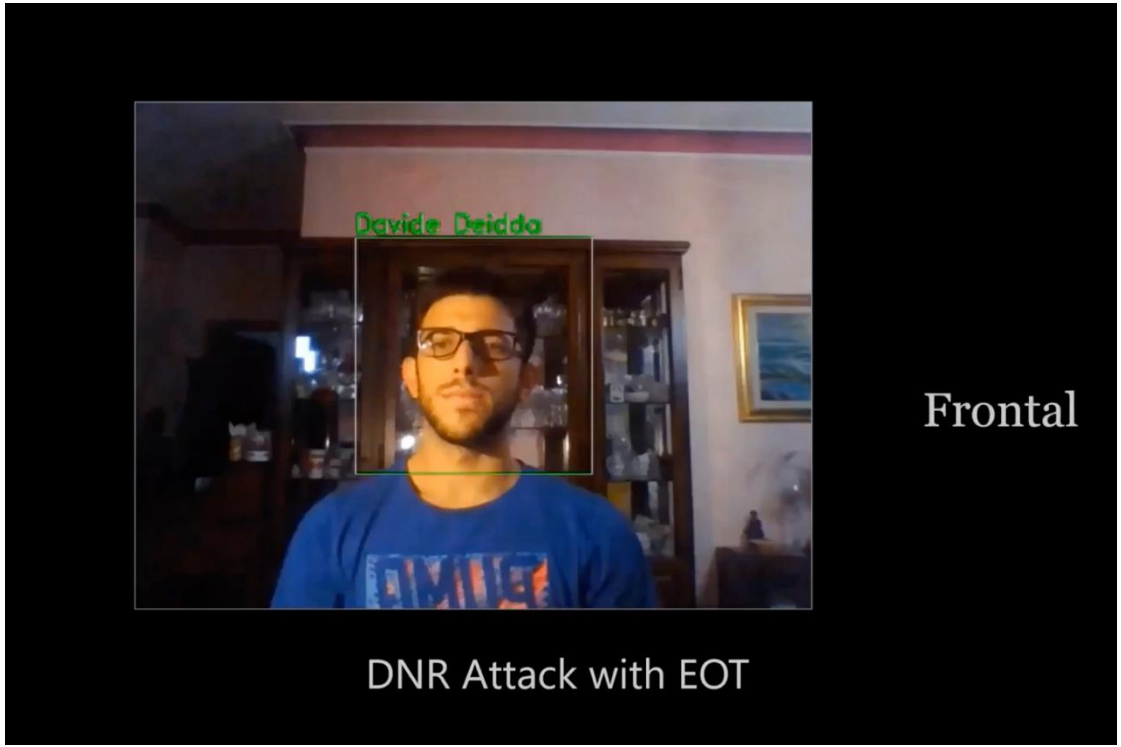



S. Sabour at al., ICLR 2016

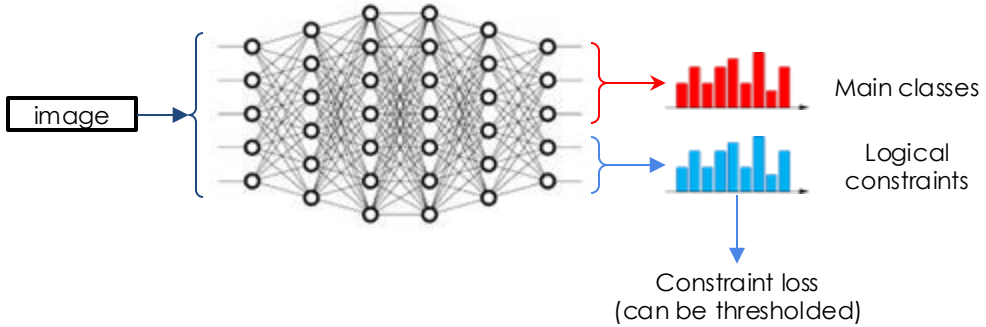
# Deep Neural Rejection against Adversarial Examples



# DNR against Physical Attacks

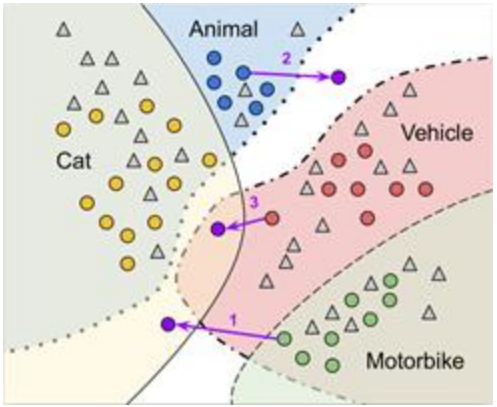


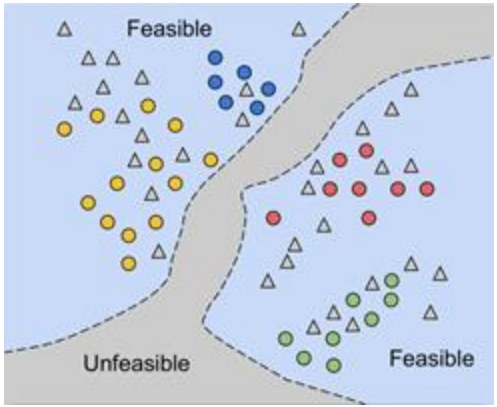
# Robust Learning with Domain Knowledge




$$
\begin{aligned}
&\forall x, \quad \text{CAT}(x) \Rightarrow \text{ANIMAL}(x)\,,\\
&\forall x, \quad \text{MOTORBIKE}(x) \Rightarrow \text{VEHICLE}(x)\,,\\
&\forall x, \quad \text{VEHICLE}(x) \Rightarrow \neg\text{ANIMAL}(x)\,,\\
&\forall x, \quad \text{CAT}(x) \vee \text{ANIMAL}(x) \vee \text{MOTORBIKE}(x) \vee \text{VEHICLE}(x)
\end{aligned}
$$

$$
\min_{\mathbf{f}} = \frac{1}{n}\sum_{i=1}^{l} L_y(\mathbf{f}(\mathbf{x}_i), \mathbf{y}_i) + \sum_{j=1}^{l+u}\sum_{h=1}^{m} \lambda_m \cdot L_\phi(\phi_h(\mathbf{f}(\mathbf{x}_j))) + \lambda\|\mathbf{f}\|
$$





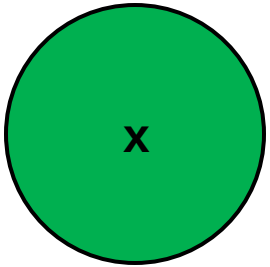

Melacci, Biggio et al., IEEE TPAMI 2022 https://arxiv.org/abs/2006.03833

# Certified Defenses

# *Ideal* World: Evaluating Certified Robustness

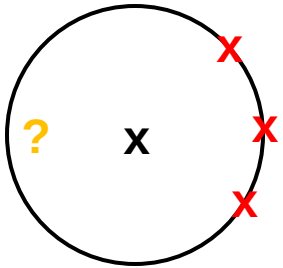
- **Certified robustness:** Ensuring that no adversarial example exists within the given perturbation domain

$$\min_{\delta} \; L(x+\delta, y_t, \theta)$$
$$\text{s.t.} \quad ||\delta|| \leq \epsilon, \;\; x+\delta \in [0,1]^{\mathrm{d}}$$




- Only doable in simple/tractable cases…

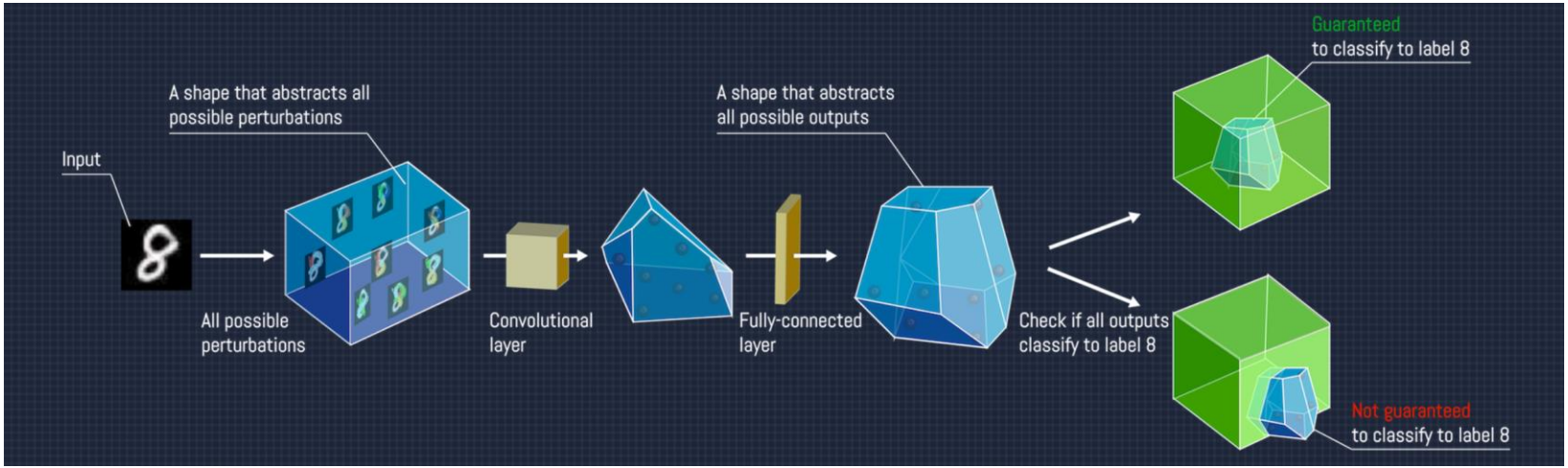
- **Empirical robustness:** run empirical attacks and count their failures
  - But… if the attack fails, we cannot conclude that no adversarial example exists…



# Formal Verification via Abstract Interpretation

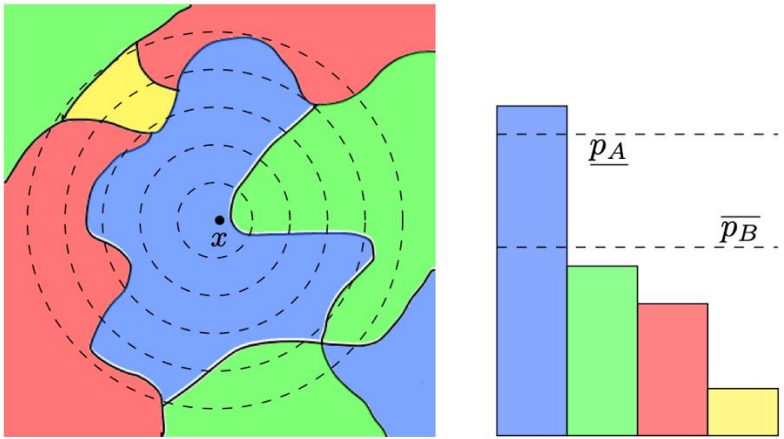
***Interval Bound Propagation (IBP)***

**AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, IEEE S&P 2018**
Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, Martin Vechev



# Randomized Smoothing

- Formal guarantee on adversarial robustness (a.k.a. *provable robustness*)
  - classification is consistent within l2 perturbations of size less than a given radius




$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}(f(x+\varepsilon) = c) \qquad (1)$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$

**Theorem 1.** *Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let $g$ be defined as in (1). Suppose $c_A \in \mathcal{Y}$ and $\underline{p_A}, \overline{p_B} \in [0,1]$ satisfy:*

$$\mathbb{P}(f(x+\varepsilon) = c_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_A} \mathbb{P}(f(x+\varepsilon) = c) \qquad (2)$$

*Then $g(x+\delta) = c_A$ for all $\|\delta\|_2 < R$, where*

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \qquad (3)$$



Cohen et al., ICML 2019 https://arxiv.org/abs/1902.02918

# Certified Defenses against Patch Attacks

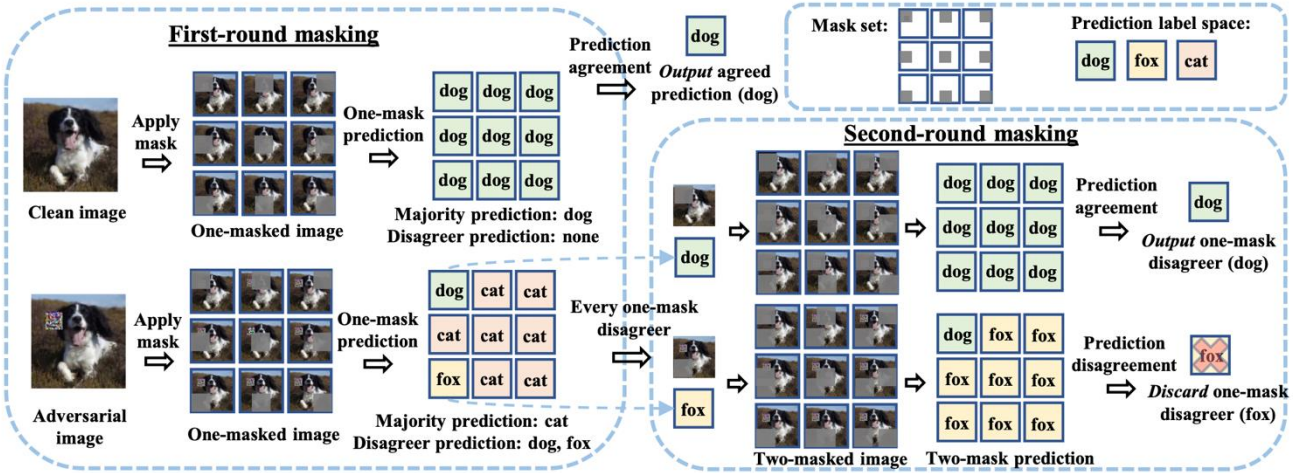



Figure 1: **Overview of double-masking defense.** The defense applies masks to the input image and evaluates model prediction on every masked image. *Clean image:* all one-mask predictions typically agree on the correct label ("dog"); our defense outputs the agreed prediction. *Adversarial image:* one-mask predictions have a disagreement; we aim to recover the benign prediction. We first categorize all one-mask predictions into the *majority prediction* (the one with the highest prediction label occurrence; the label "cat" in this example) and *disagreer predictions* (the ones that disagree with the majority; the labels "dog" and "fox"). For every mask that leads to a disagreer prediction, we add a set of second masks and evaluate two-mask predictions. If all two-mask predictions agree with this one-mask disagreer, we *output* its prediction label (the label "dog"; illustrated in the upper row of the second-round masking); otherwise, we *discard* it (the label "fox"; in the lower row of the second-round masking).

# Certified Defenses against Patch Attacks

- PatchCleanser (PC) can certify classification up to a given patch size and for a fraction of the input samples
  - Robust accuracy here is certified. It means that no attack can decrease it further, but also that empirical attacks may perform worse…
    - i.e., a gradient-based attack may actually turn out to find a higher robustness!

Table 2: Clean accuracy and certified robust accuracy for different defenses and datasets†

| Dataset | ImageNette [15] | | | | | | ImageNet [12] | | | | | | CIFAR-10 [25] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Patch size | 1% pixels | | 2% pixels | | 3% pixels | | 1% pixels | | 2% pixels | | 3% pixels | | 0.4% pixels | | 2.4% pixels | |
| Accuracy (%) | clean | robust | clean | robust | clean | robust | clean | robust | clean | robust | clean | robust | clean | robust | clean | robust |
| PC-ResNet | **99.6** | 96.4 | **99.6** | 94.4 | **99.5** | 93.5 | 81.7 | 58.4 | 81.6 | 53.0 | 81.4 | 50.0 | 98.0 | 88.5 | 97.8 | 78.8 |
| PC-ViT | **99.6** | **97.5** | **99.6** | **96.4** | **99.5** | **95.3** | **84.1** | **66.4** | **83.9** | **62.1** | **83.8** | **59.0** | **99.0** | **94.3** | **98.7** | **89.1** |
| PC-MLP | 99.4 | 96.8 | 99.3 | 95.3 | 99.4 | 94.6 | 79.6 | 58.4 | 79.4 | 53.8 | 79.3 | 50.7 | 97.4 | 86.1 | 97.0 | 78.0 |
| IBP [7] | computationally infeasible | | | | | | | | | | | | 65.8 | 51.9 | 47.8 | 30.8 |
| CBN [67] | 94.9 | 74.6 | 94.9 | 60.9 | **94.9** | 45.9 | 49.5 | 13.4 | 49.5 | 7.1 | 49.5 | 3.1 | 84.2 | 44.2 | 84.2 | 9.3 |
| DS [27] | 92.1 | 82.3 | 92.1 | 79.1 | 92.1 | 75.7 | 44.4 | 17.7 | 44.4 | 14.0 | 44.4 | 11.2 | 83.9 | 68.9 | 83.9 | 56.2 |
| PG-BN [61] | **95.2** | **89.0** | **95.0** | **86.7** | 94.8 | **83.0** | **55.1** | **32.3** | **54.6** | **26.0** | **54.1** | **19.7** | 84.5 | 63.8 | 83.9 | 47.3 |
| PG-DS [61] | 92.3 | 83.1 | 92.1 | 79.9 | 92.1 | 76.8 | 44.1 | 19.7 | 43.6 | 15.7 | 43.0 | 12.5 | 84.7 | 69.2 | 84.6 | 57.7 |
| BagCert‡[35] | – | – | – | – | – | – | 45.3 | 27.8 | 45.3 | 22.7 | 45.3 | 18.0 | **86.0** | **72.9** | **86.0** | **60.0** |

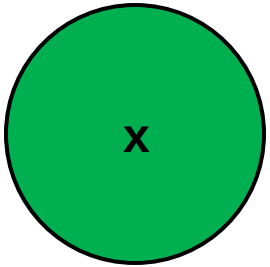
† We mark the best result for PatchCleanser models and the best result for prior works in bold.
‡ The BagCert numbers are provided by the authors [35] through personal communication since the source code is unavailable; results for ImageNette are not provided.

# Shall We Trust Empirical Evaluations?

# *Ideal* World: Evaluating Certified Robustness

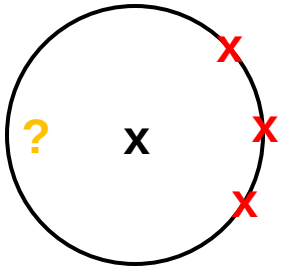
- **Certified robustness:** Ensuring that no adversarial example exists within the given perturbation domain

$$\min_{\delta} \; L(x+\delta, y_t, \theta)$$
$$\text{s.t.} \quad ||\delta|| \leq \epsilon, \;\; x+\delta \in [0,1]^{\mathrm{d}}$$



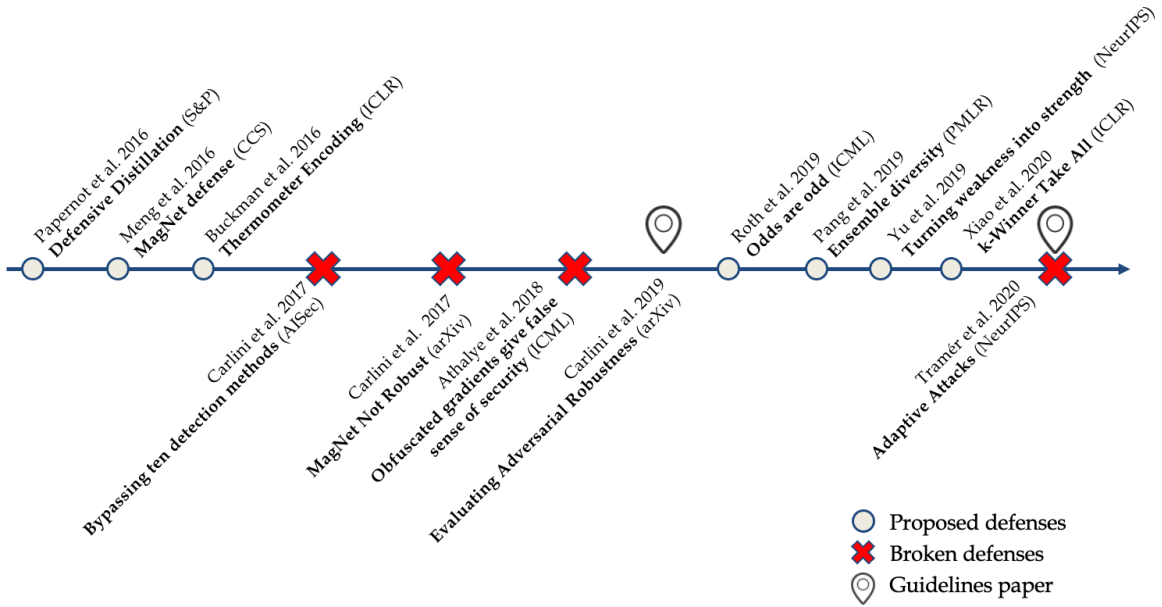

- Only doable in simple/tractable cases…

- **Empirical robustness:** run empirical attacks and count their failures
  - But… if the attack fails, we cannot conclude that no adversarial example exists…



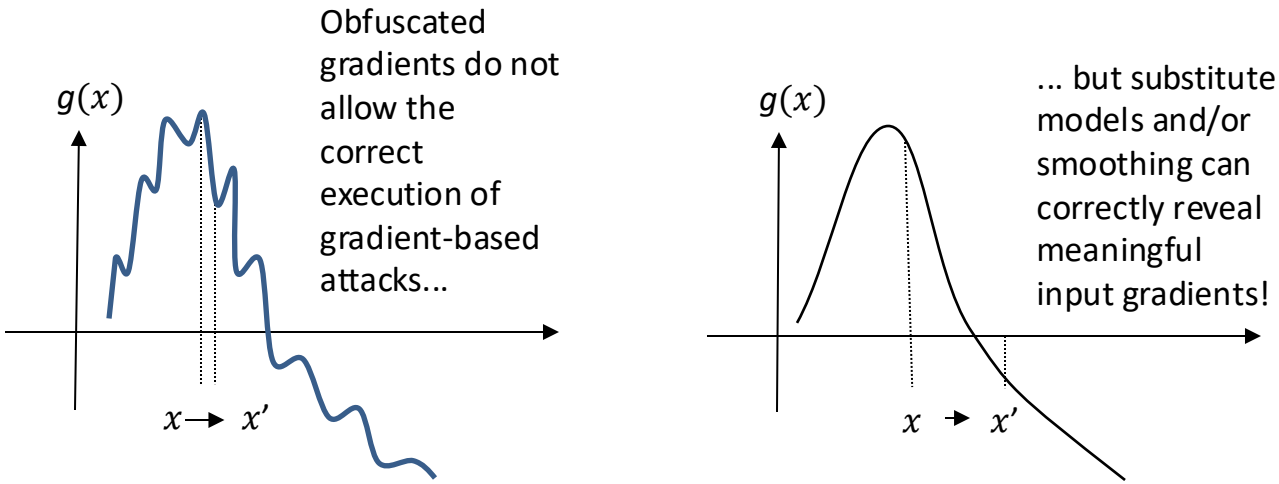
# Flawed Evaluations / Ineffective Defenses

- **Problem**: formal evaluations do not scale, adversarial robustness evaluated mostly empirically, via gradient-based attacks
- **Gradient-based attacks can fail:** many flawed evaluations have been reported, with defenses easily broken by adjusting/fixing the attack algorithms



# Ineffective Defenses: Obfuscated Gradients

- Carlini & Wagner (SP' 17), Athalye et al. (ICML '18), Tramer et al. (NeurIPS '20) have shown that
  - some recently-proposed defenses rely on obfuscated / masked gradients...
  - ... and they can be circumvented



# Backward Pass Differentiable Approximation (BPDA)

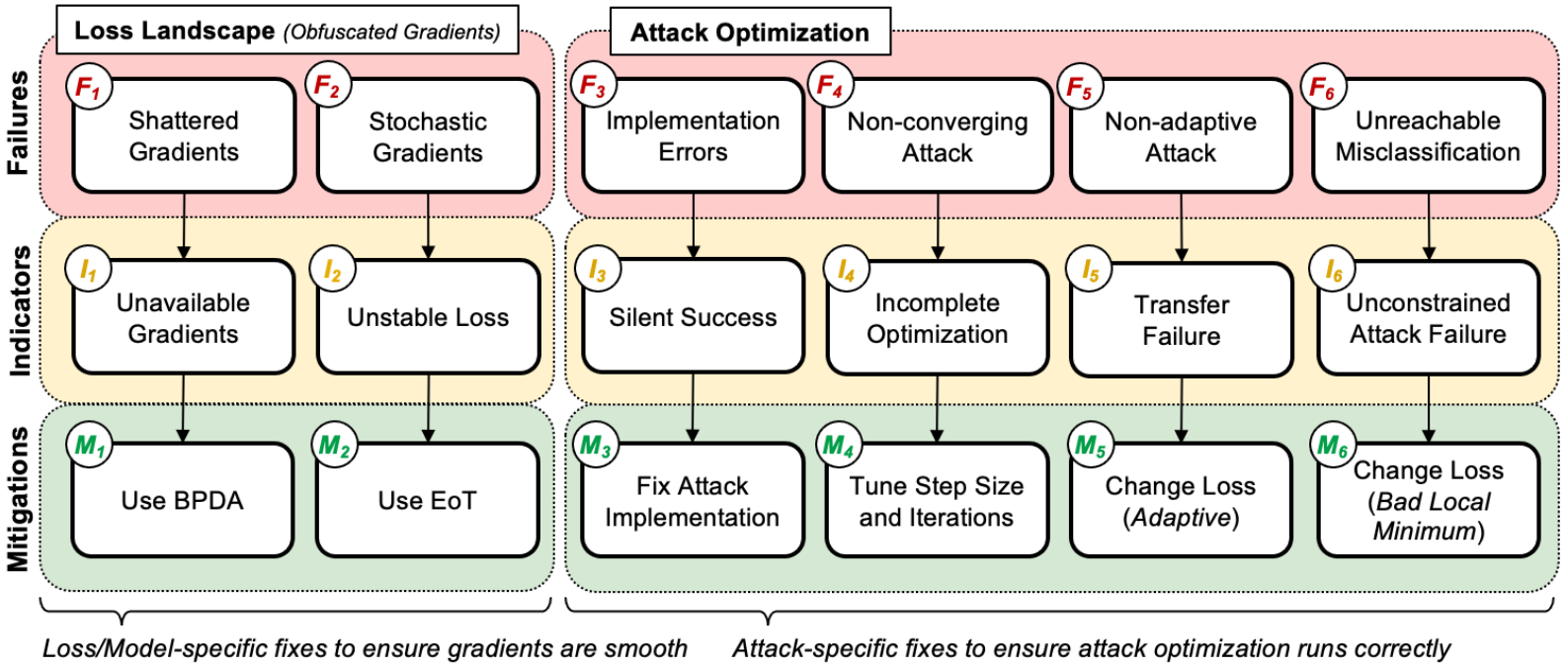
- If gradients of a DNN are "unavailable" at some representation level g(x), e.g. when inputs are discretized or non-differentiable transformations are applied, one can approximate g(x) in the backward pass with a smoother function which approximates it
  - e.g. Distillation, Thermometer Encoding, JPEG compression

$$\nabla_x f(g(x))|_{x=\hat{x}} \approx \nabla_x f(x)|_{x=g(\hat{x})}$$

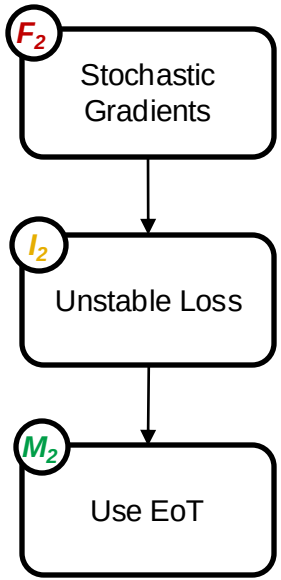
- **Trivial case:** just replace the gradient of g(x) with ones during the backward pass

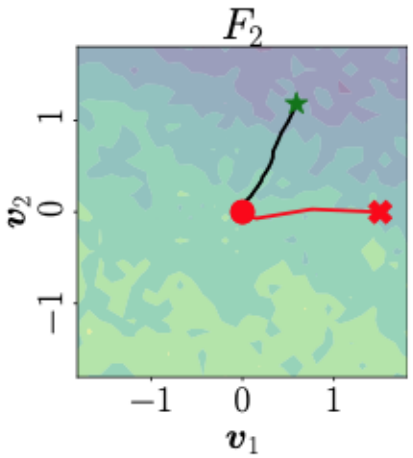
# Detecting Unrealiable Evaluations

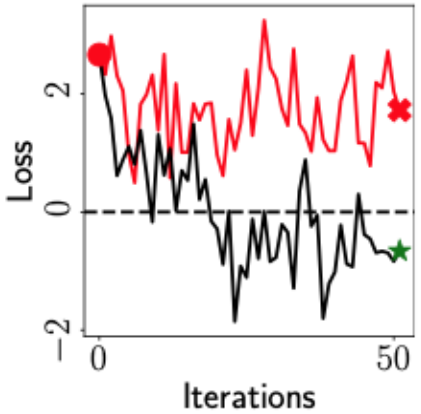
# Indicators of Attack Failure



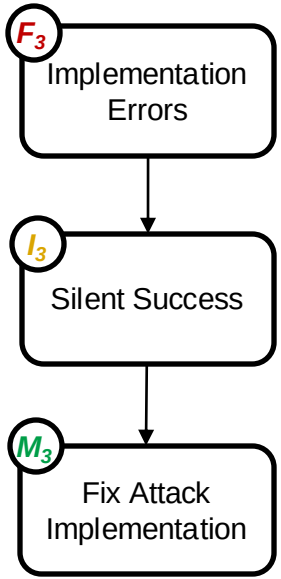
# IoAF: Focus on Stochastic Gradients

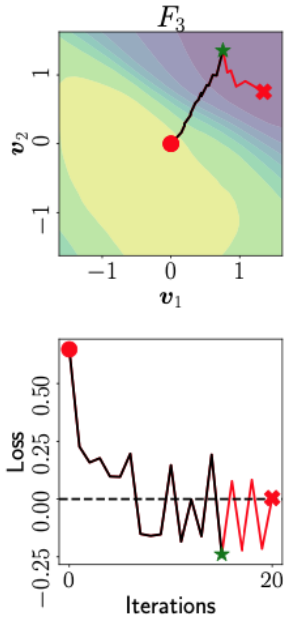


# IoAF: Focus on Implementation Errors



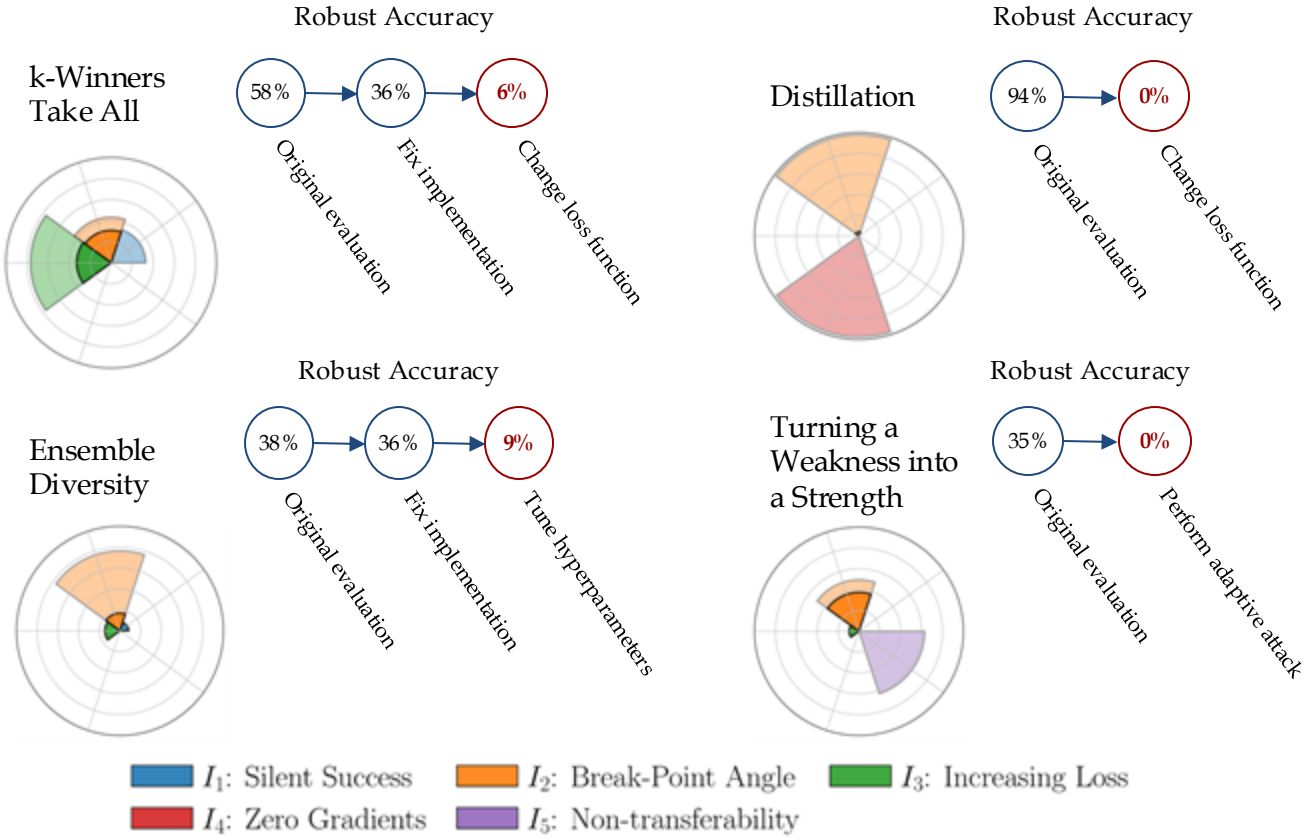

- Wrong PGD attack implementations in widely-used libraries

| Library | Version | GitHub ☆ |
|---|---|---|
| Cleverhans | 4.0.0 | 5.6k |
| ART | 1.11.0 | 3.1k |
| Foolbox | 3.3.3 | 2.3k |
| Torchattacks | 3.2.6 | 984 |



Pintor, Biggio et al., *Indicators of Attack Failure: …*, NeurIPS 2022

# Experiments



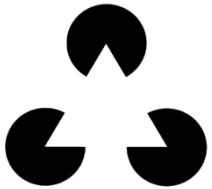
# Issues with Existing Libraries / Tools (2022)

- Flawed attack implementations
  - **Cleverhans** (PyTorch, Tensorflow, JAX),
  - **ART** (NumPy, PyTorch, and Tensorflow),
  - **Foolbox** (EagerPy, which wraps the implementation of NumPy, PyTorch, Tensorflow, and JAX)
  - **Torchattacks** (PyTorch)

| Library | Version | GitHub ☆ |
|---|---|---|
| Cleverhans | 4.0.0 | 5.6k |
| ART | 1.11.0 | 3.1k |
| Foolbox | 3.3.3 | 2.3k |
| Torchattacks | 3.2.6 | 984 |

- **RobustBench/AutoAttack** has a flag to detect *unreliable* evaluations

# Are Indistinguishable Perturbations a Real Security Threat?



# Is This a Real Security Threat?

- **Adversarial examples** can exist in the *physical world*, we can fabricate concrete adversarial objects (glasses, road signs, etc.)

- But the effectiveness of attacks carried out by adversarial objects is still to be investigated with **large-scale experiments** in *realistic* **security scenarios**

- Gilmer et al. (2018) have recently discussed the realism of security threat caused by adversarial examples, pointing out that it should be carefully investigated
  - *Are indistinguishable adversarial examples a real security threat?*
  - For which real security scenarios adversarial examples are the best attack vector? Better than attacking components outside the machine learning component
  - …

Justin Gilmer et al., Motivating the Rules of the Game for Adversarial Example Research, https://arxiv.org/abs/1807.06732

# Indistinguishable Adversarial Examples

- Minimize $\|r\|_2$ subject to:
  1. $f(x+r) = l \quad f(x) \neq l$
  2. $x + r \in [0,1]^m$

The adversarial image $x + r$ is visually hard to distinguish from $x$

*…There is a **torrent of work** that views increased robustness to **restricted perturbations** as making these models **more secure**. While not all of this work requires completely indistinguishable modications, many of the papers focus on specifically small modications, and the language in many suggests or implies that the **degree of perceptibility** of the **perturbations** is an important aspect of their **security risk**…*

Justin Gilmer et al., Motivating the Rules of the Game for Adversarial Example Research, https://arxiv.org/abs/1807.06732

# Indistinguishable Adversarial Examples

- The attacker can benefit by minimal perturbation of a legitimate input; e.g., she could use the attack for a longer period of time before it is detected

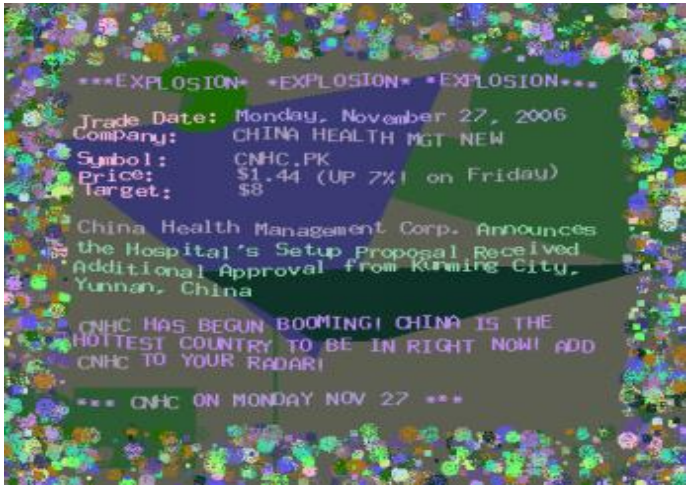
- But is *minimal perturbation* a necessary constraint for the attacker?

# Indistinguishable Adversarial Examples

- Is *minimal perturbation* a necessary constraint for the attacker?

# Attacks with Content Preservation

There are well known security applications where minimal perturbations and indistinguishability of adversarial inputs are not required at all…




# Are Indistinguishable Perturbations a Real Security Threat?

*…At the time of writing, we were **unable** to find a **compelling example** that **required indistinguishability**…*

*To have the largest impact, we should both recast future adversarial example research as a **contribution** to **core machine learning** and develop new abstractions that capture **realistic threat models**.*

Justin Gilmer et al., Motivating the Rules of the Game for Adversarial Example Research, https://arxiv.org/abs/1807.06732

Battista Biggio
battista.biggio@unica.it
@biggiobattista

# Thanks!

*If you know the enemy and know yourself, you need not fear the result of a hundred battles*
Sun Tzu, The art of war, 500 BC