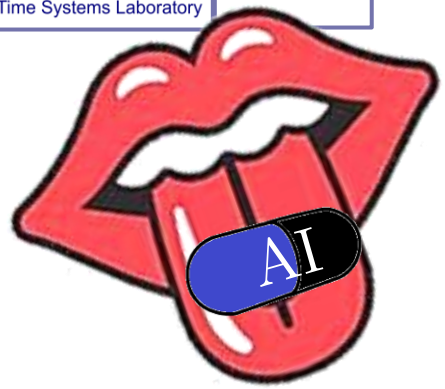
Retis Real-Time Systems Laboratory | AI Pills




# Adversarial examples: attacks and defenses

Federico Nesti, f.nesti@santannapisa.it

# Outline

- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

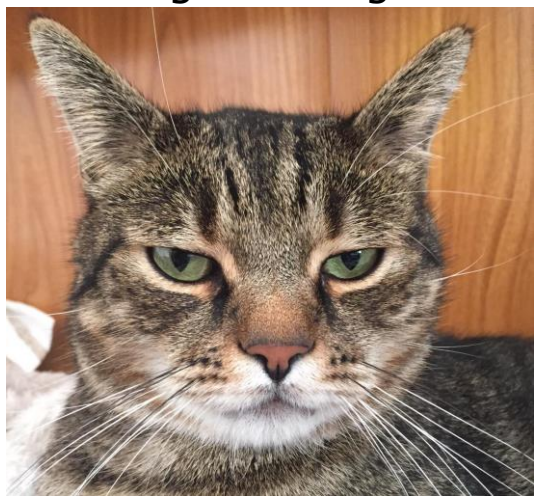
# Outline

- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

# Adversarial Examples

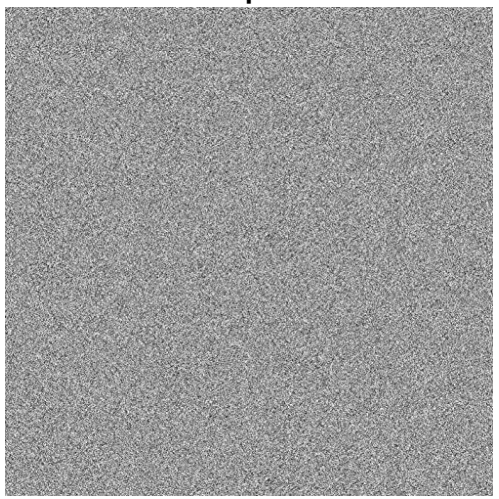
Adversarial Examples are inputs that are maliciously crafted to fool a neural network. They are typically generated by adding a small perturbation, not recognizable by humans.

Original image



**Network Prediction:**
Cat 96%

Adversarial perturbation



Adversarial Example

**Network Prediction:**
Dog 92%

**Note:** any type of input can be transformed into adversarial examples. Focus on images

# Adversarial Examples

**WHO WOULD WIN?**

STATE OF THE ART NEURAL NETWORK

ONE NOISY BOI

Training Accuracy: 95%
Test Accuracy: 94%
Adversarial Example:

# Adversarial Examples

Properties:

- Adversarial examples typically induce **high-confidence (wrong) predictions**.
- **Transferability**: adversarial examples are typically crafted for one specific network. Often, these adversarial examples are able to fool also other models that were trained on the same dataset.
- **Visually similar to original images** (often undistinguishable).
- **Extreme «fragility»**: standard adversarial examples can instantaneously lose the fooling power when slightly modified.

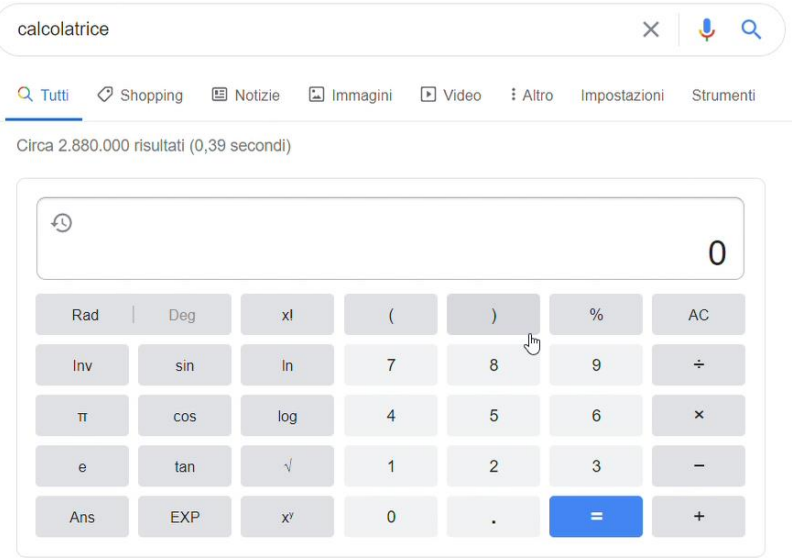
# Why do AEs exist?

Adversarial Examples exist thanks to a combination of two factors:

High dimensionality of the **number of parameters** in the deep models (typically millions)

| Model | Params | SIZE(MB) |
|---|---|---|
| SimpleNet | 6.4M | 24.4 |
| SqueezeNet | 1.25M | 4.7 |
| Inception v4* | 42.71M | 163 |
| Inception v3* | 23.83M | 91 |
| Incep-Resv2* | 55.97M | 214 |
| ResNet-152 | 60.19M | 230 |
| ResNet-50 | 25.56M | 97.70 |
| AlexNet | 60.97M | 217.00 |
| GoogleNet | 7M | 40 |
| NIN | 7.6M | 29 |
| VGG16 | 138.36M | 512.2 |

**HUGE dimensionality of the input space** (e.g., $28 \times 28$ image, single channel, uint8: $256^{28\times28}$)

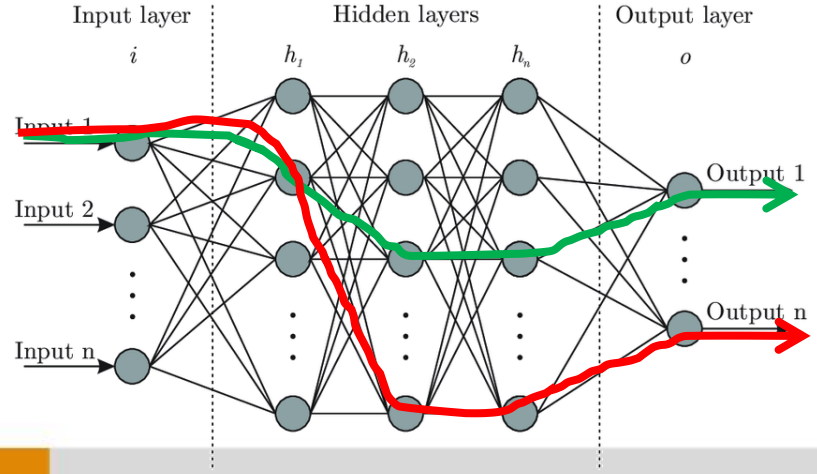


# Why do AEs exist?

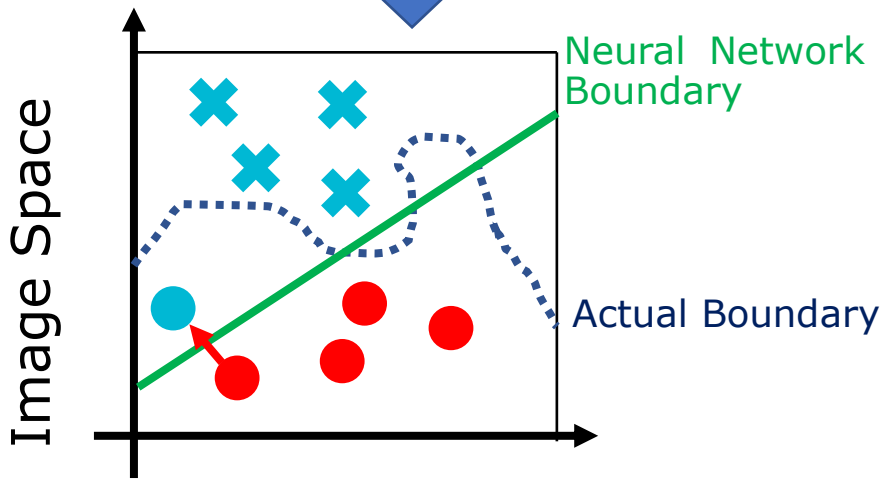
Adversarial Examples exist thanks to a combination of two factors:

High dimensionality of the **number of parameters** in the deep models (typically millions)

Many «useless» neurons and activation patterns that are exploited.




**HUGE dimensionality of the input space** (e.g., $28 \times 28$ image, single channel, uint8: $256^{28\times28}$)




…but this happens in **almost infinite dimensions**!



Carrara, Fabio et al. (2019). Adversarial Examples Detection in Features Distance Spaces

# Other possible explanations

Other point the finger against other factors.

**Algorithm design** has drawn much more attention than **quality of data**, even though stupid algorithms (e.g., k-nearest-neighbor) work well with huge amount of «good» data.

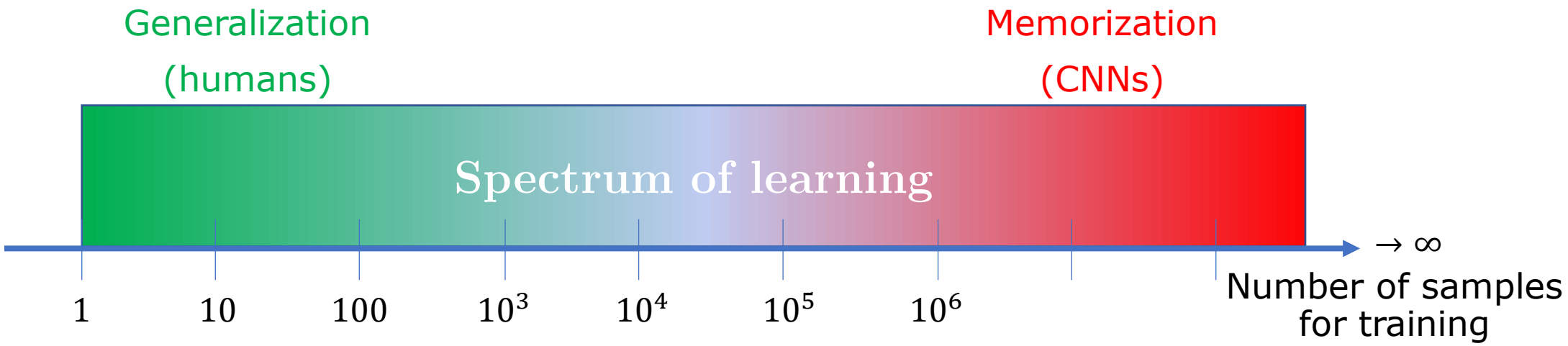
Some of the images of the validation set of the class «Tinca Tinca» in Imagenet. All images share the presence of a human…

The input space is sampled in a very biased way!

# Other possible explanations

Other point the finger against other factors.

**Supervised learning itself (in the form of SEMANTIC SUPERVISION)** might be to blame: experiments showed that CNNs are able to «memorize» noise patterns, even when assigning random target labels.






Zhang, C. et al. (2017). Understanding deep learning requires rethinking generalization.

# Other possible explanations

Other point the finger against other factors.

**Semantic supervision** makes neural networks memorize patterns, not learn to generalize. Can self-supervision (in the reference) overcome these issues?

«Shetland Sheepdog» in both cases!

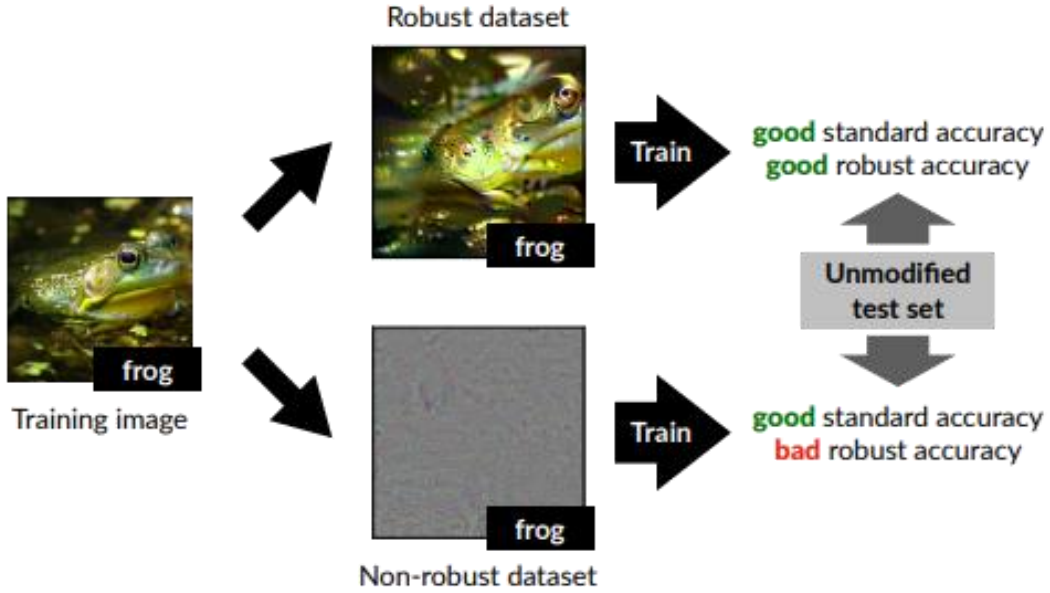


Alexei Efros, Self-supervision for computer vision tasks
https://www.youtube.com/watch?v=_V-WpE8cmpc

# Other possible explanations

Other point the finger against other factors.

Another work claims that supervised learning on regular training datasets makes the classifiers learn to base decisions on **non-robust features** (e.g., high-frequency patterns of the background) rather than on robust ones.

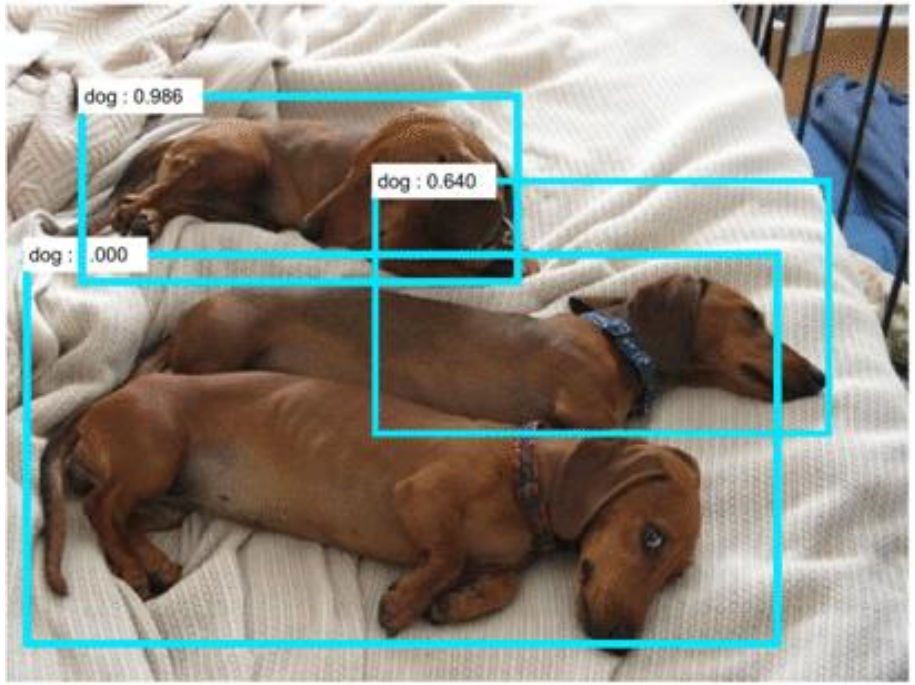


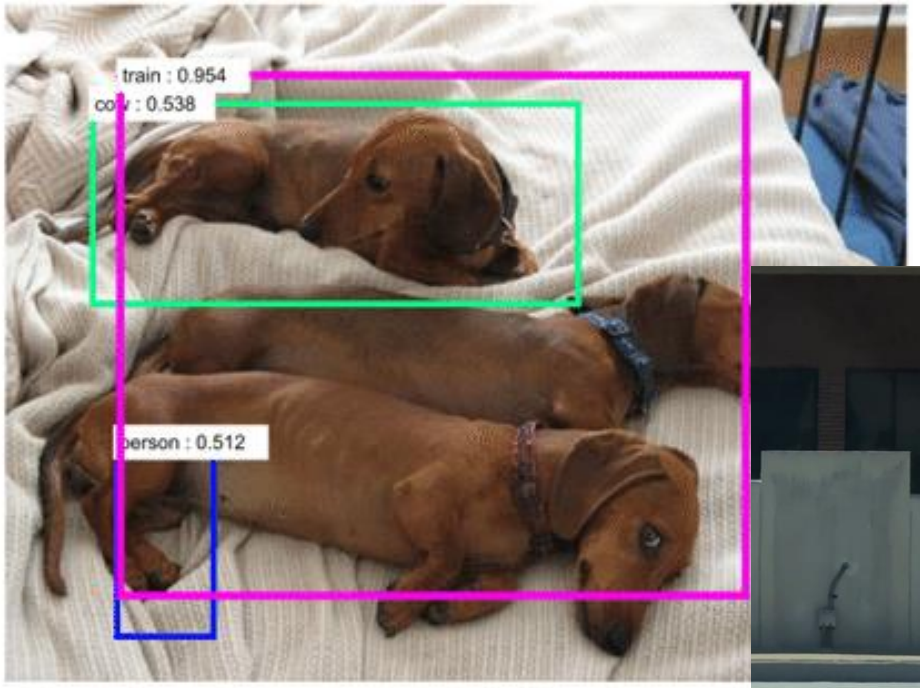


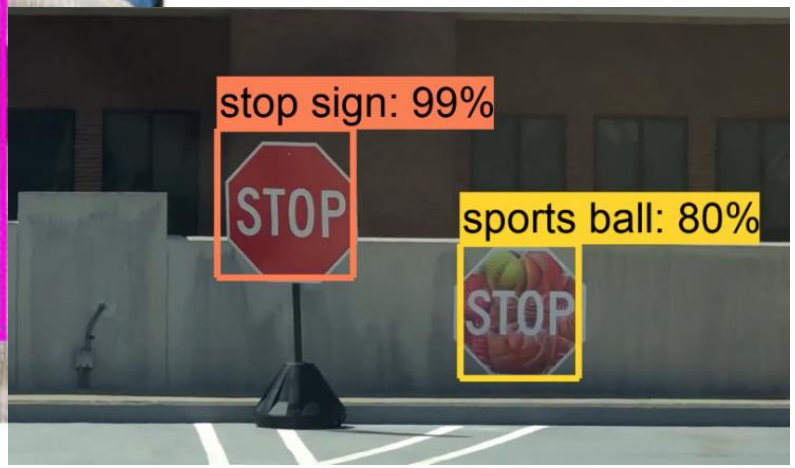

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A., "Adversarial Examples Are Not Bugs, They Are Features", arXiv e-prints, 2019.

# Are AEs a serious concern…

A neural network can be forced towards a wrong prediction by an attacker, causing potential catastrophic damage. Hence, machine learning models are susceptible of cyber-attacks! Imagine a neural network for autonomous driving. **However…**






Yuan, Xiaoyong et al. (2017). Adversarial Examples: Attacks and Defenses for Deep Learning.

# …or a research buzzword?

The neural networks involved in safety-critical tasks are **not easily attackable**!

The neural networks architectures and parameters are **not publicly available.**

In a real self-driving car, neural networks are involved in perception together with a set of **redundant processing algorithms**.

Usually, **tracking techniques** (e.g., Kalman Filter) are used to filter outliers and spurious detections.

It usually takes time to «prepare» an adversarial example, while it is **difficult to run the generation process in real-time.**

# Outline

- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

# Minimal taxonomy of AEs

Adversarial Examples can be classified in many ways.

- A first classification is about the **distance to the original image**. In this presentation we are considering **bounded** adversarial examples, for which the perturbation has bounded value (pixel-wise) $|r_i| < \epsilon \forall i$.
- Adversarial examples differ for the «timing» of attacks. Attacks that target the training process are **poisoning** attacks (since they pollute the dataset). In this presentation we are considering adversarial attacks during **inference**.
- We are focusing on attacks on **classifiers** (regression attacks are also possible)

# Minimal taxonomy of AEs

**Targeted/untargeted**

i.e.: the attacker aims at forcing the classifier to choose any wrong class (**untargeted**) vs one specific class (**targeted**)

i.e.: the attacker is assumed to know both architecture and parameters of the network (**white box** setting) or anything about the model (**black box**). Intermediate settings can be considered.

**Knowledge of the adversary**

**Type of norm**

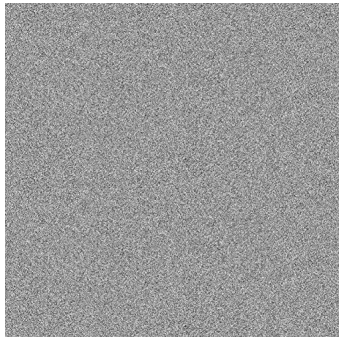
i.e.: different norms (**2-norm, 1-norm, ∞-norm**) can be used to express the magnitude of the adversarial perturbation.

In the following slides no specific norm is considered, any of these will work (with different effects)

# Terminology and notation

Original image and label

$x, t_{true}$

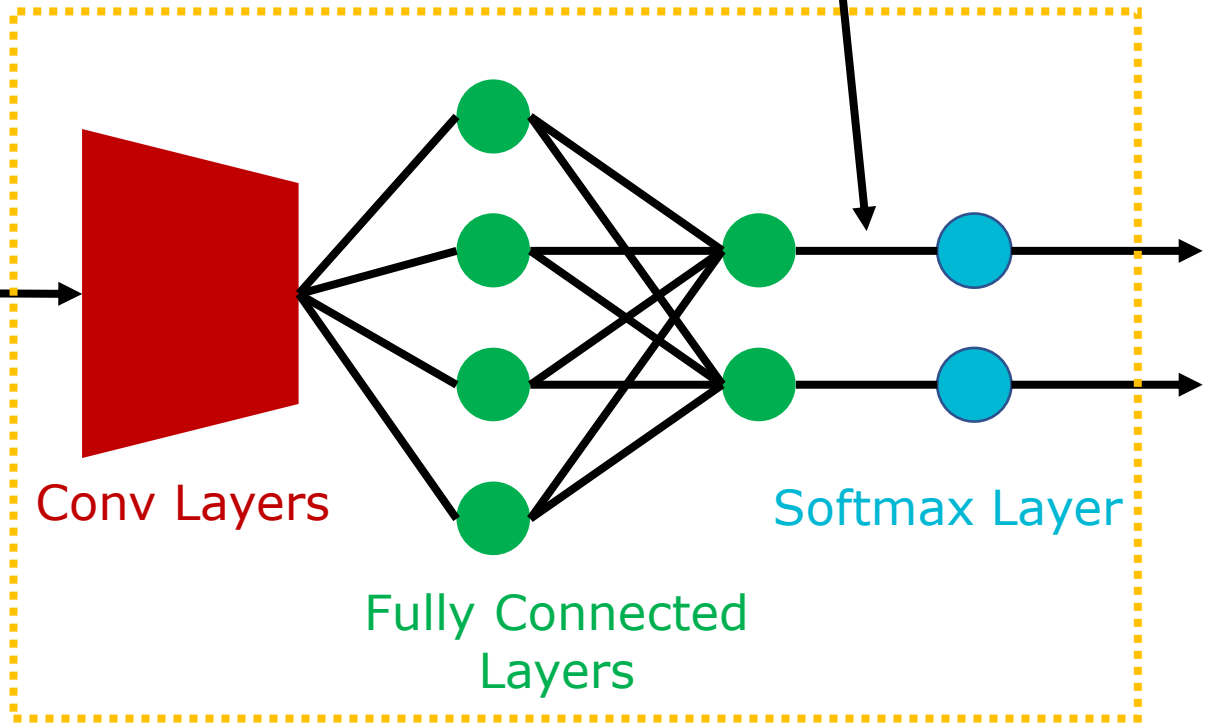


Adversarial Perturbation

$r$

Logits

$Z(x+r)_k, k = 1, \ldots, n_{classes}$




Network output (probability distribution)

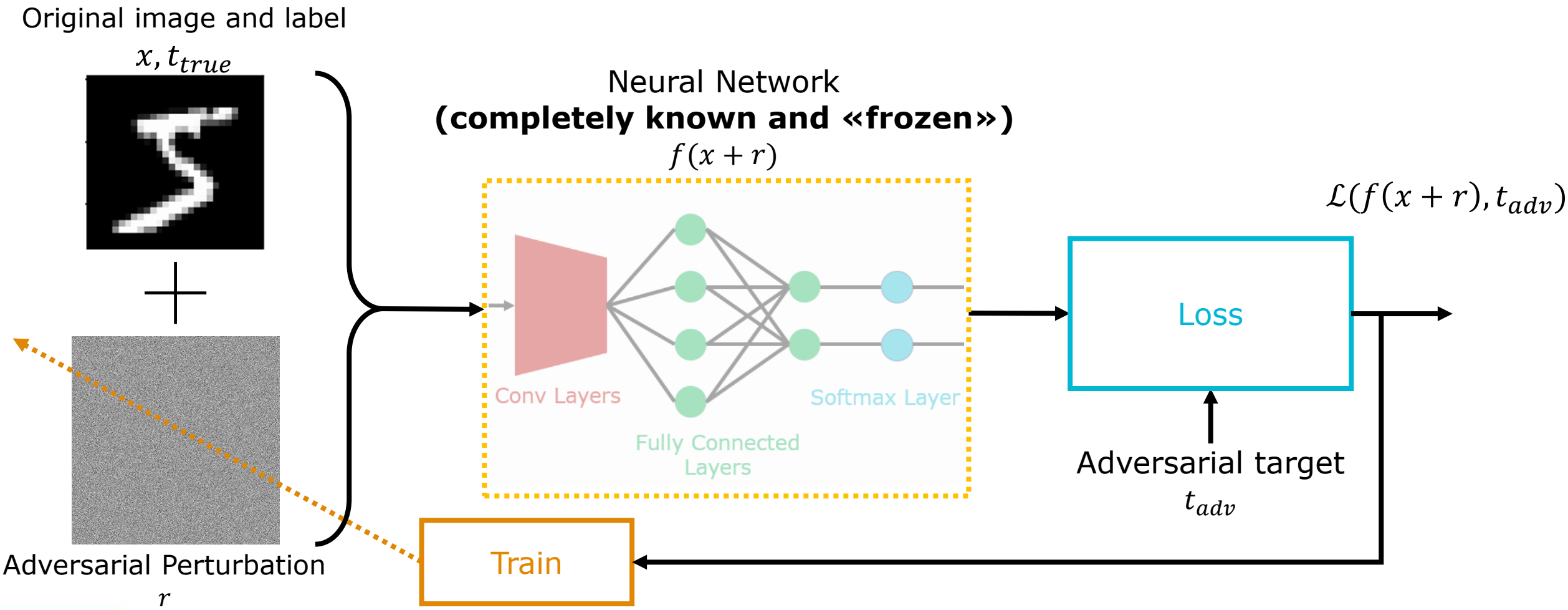
$f(x+r)$

# Outline

- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

# White box attacks

**These are the strongest attacks!**



# Crossentropy loss

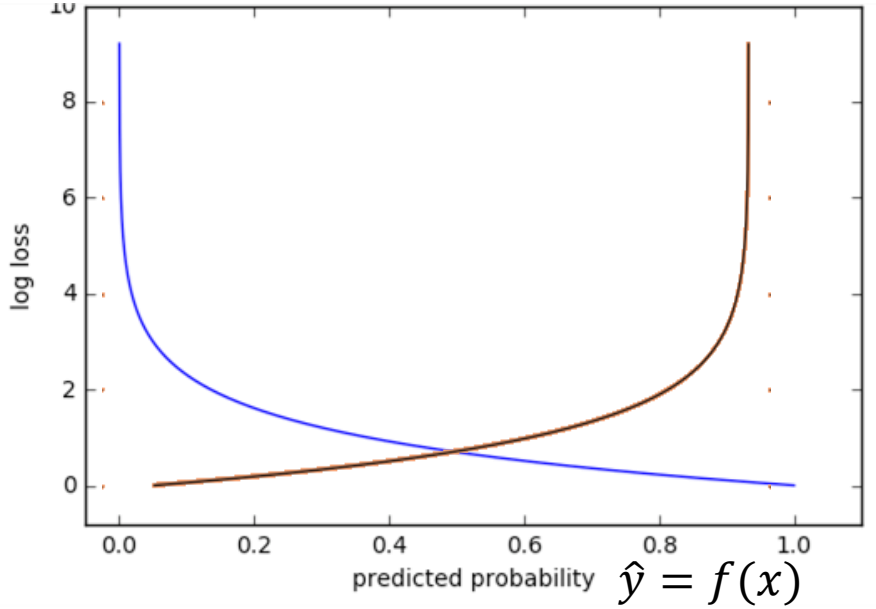
**Binary classification:** $\mathcal{L}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$

Network prediction: $\hat{y} = f(x)$

Target label

What does this mean?

With $y = 1$, $\mathcal{L}(\hat{y}, 1) = -\log \hat{y}$
With $y = 0$, $\mathcal{L}(\hat{y}, 0) = -\log(1 - \hat{y})$




**Multi-class classification:** $\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{M} y \log \hat{y}$

# Crossentropy loss

**Binary classification:** $\mathcal{L}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$

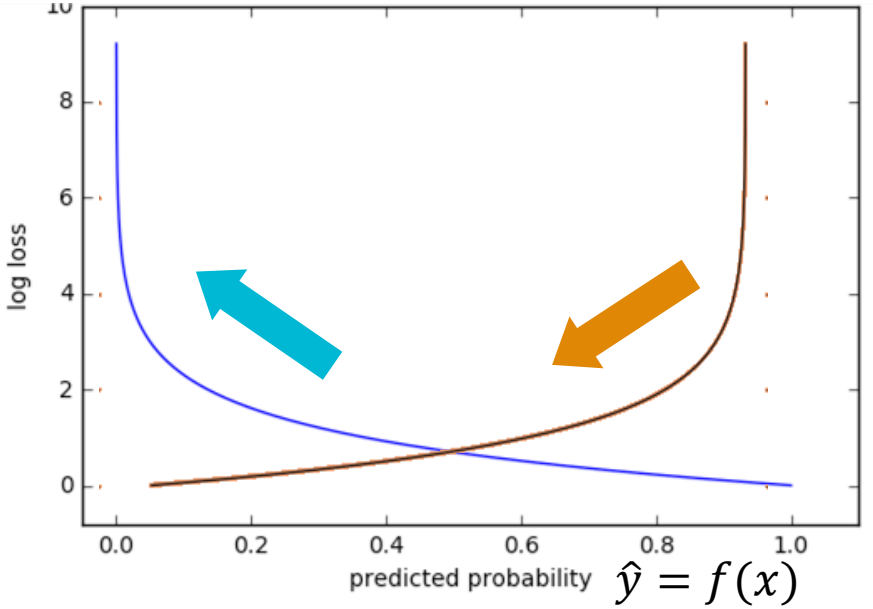
Network prediction: $\hat{y} = f(x)$

Target label

$t_{true} = 1, t_{adv} = 0$

Untargeted attack $\rightarrow \max \mathcal{L}(\hat{y}, t_{true})$

Targeted attack $\rightarrow \min \mathcal{L}(\hat{y}, t_{adv})$




**Multi-class classification:** $\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{M} y \log \hat{y}$

# Fast Gradient Sign Method (FGSM)

**Untargeted**, **one-shot** attack. Linearization of the loss function to get the steepest direction.

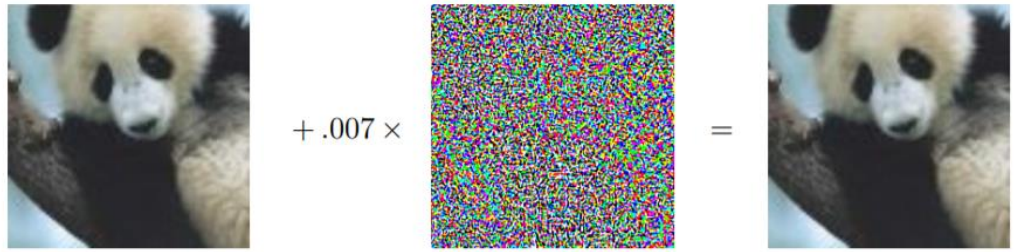
Direction only!

$$r = \epsilon \ \text{sign}\, \nabla_x \mathcal{L}(f(x + r), t_{true})$$

Adversarial strength

Usually crossentropy

True label




It is basically one step of **gradient ascent** on the loss function: we want to get away from the correct prediction on the steepest direction.




Goodfellow, I. J., Shlens, J., and Szegedy, C., "Explaining and Harnessing Adversarial Examples", arXiv e-prints, 2014.

# Basic Iterative Method

Straightforward iterative extension of FGSM (i.e., similar to **gradient ascent**!)

$$x_0^{ADV} = x \quad (i.e., r = 0)$$

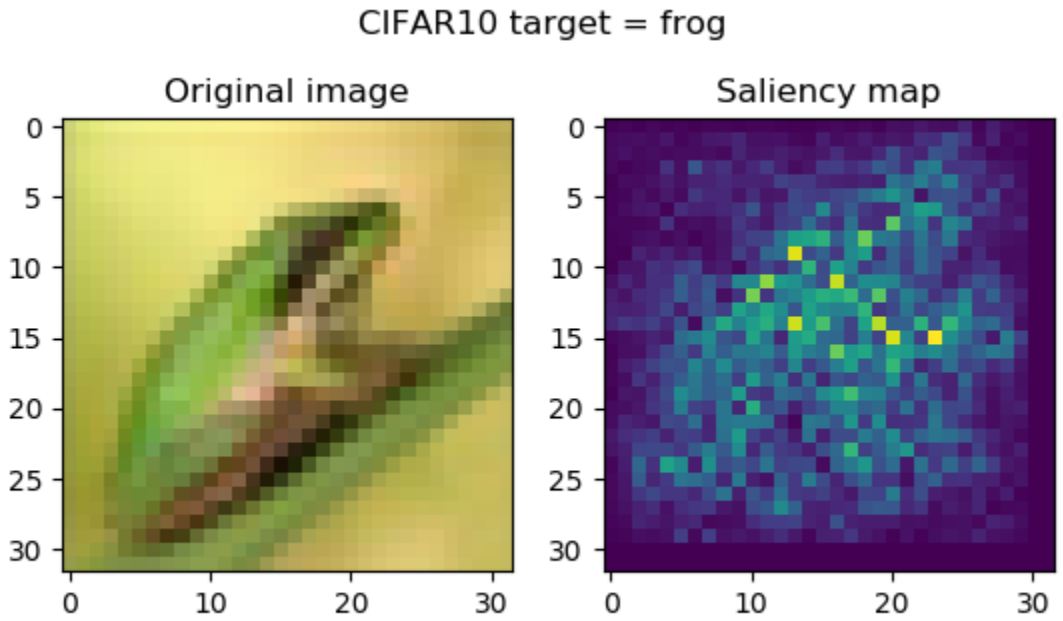
$$x_{k+1}^{ADV} = x_k^{ADV} + \alpha \ \text{sign} \nabla_x \mathcal{L}\left(f\left(x_k^{ADV}\right), t_{true}\right)$$

subject to $|r_i| < \epsilon \ \forall i$



Kurakin, A., I. Goodfellow, and S. Bengio. "Adversarial examples in the physical world. arXiv 2016." *arXiv preprint arXiv:1607.02533*.

# Jacobian Saliency Map Attack (JSMA)




Saliency maps are a simple and effective way of visualizing which pixels are the most influent (both positively and negatively towards a specific class $c$) on the decision of the network. Changing the most influent pixel iteratively could result in misclassifications (still untargeted).



Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016.

# Jacobian Saliency Map Attack (JSMA)

$$S(x,c)_i = \begin{cases} 0 & if \ \frac{\partial f_c(x)}{\partial x_i} < 0 \ or \ \sum_{j \neq c} \frac{\partial f_j(x)}{\partial x_i} > 0 \\ \frac{\partial f_c(x)}{\partial x_i} \left| \sum_{j \neq c} \frac{\partial f_j(x)}{\partial x_i} \right| & otherwise \end{cases}$$

Ignore negative correlations between pixels and chosen class $c$

and positive correlations between pixels and all the other classes.

Keep only pixels that both **influence the decision towards $c$** and **away from all the other classes**!



Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016.

# Jacobian Saliency Map Attack (JSMA)

1. Compute Saliency Map $S(x, c)$
2. Choose most influent pixel $i^* = \arg\max S(x, c)_i$
3. Change it by a certain quantity! $x_i \leftarrow x_i + \theta$
4. Iterate 1-3 until $||x^{ADV} - x|| > \epsilon$



Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016.

# L-BFGS attack

**Targeted, iterative** attack (first adversarial examples). L-BFGS refers to the optimization algorithm used to find the optimal $c$, in order to minimize the perturbation.

$$r \leftarrow \arg\min\{\mathcal{L}(f(x+r), t_{adv}) + \mathrm{c}||r||^2\}$$
$$\text{subject to } |r_i| < \epsilon, \forall i$$

Crossentropy

Regularization term



Szegedy, Christian, et al. "Intriguing properties of neural networks. arXiv 2013." *arXiv preprint arXiv:1312.6199* (2013).

# L-BFGS attack in TF1

```python
# Definition of inputs and variable to be optimized
input = tf.placeholder(dtype='float32', shape=(None, 28 * 28))
adv_target = tf.placeholder(dtype='float32', shape=(None, 10))
adv_perturb_var = tf.get_variable(shape=(1, 28*28), dtype=tf.float32,
                                  initializer=tf.initializers.random_normal(), name="adv_pert")

# Propagation through model
epsilon = 0.3
adv_perturb = tf.clip_by_value(adv_perturb_var, -epsilon, epsilon)
adv_input = tf.clip_by_value(input + adv_perturb, 0, 1)
output = model(adv_input)

# Loss and optimization
loss = tf.reduce_sum(tf.nn.softmax_cross_entropy_with_logits(labels=adv_target, logits=output)) + 0.05 * tf.norm(tf.to_float(adv_perturb), ord=2)
train_var = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, "adv_pert")
adv_opt = tf.train.AdamOptimizer(learning_rate=0.01).minimize(loss, var_list=train_var)
```

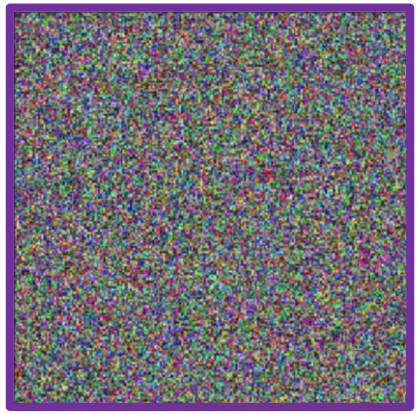
$$r = \arg\min \mathcal{L}(f(x+r), t_{adv}) + \mathrm{c}||r||^2$$

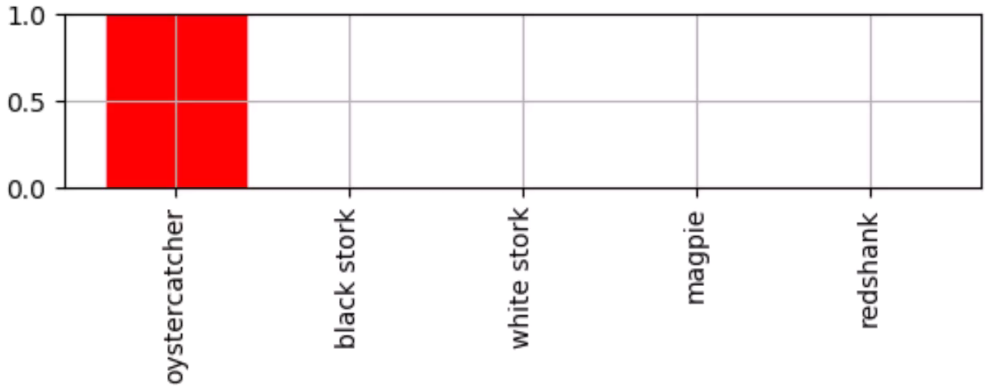
subject to $|r_i| < \epsilon, \forall i$

# L-BFGS attack

Input Image

Adversarial Perturbation $r$
(to be optimized)






Adversarial Target
(Triceratops)

ResNet-152

Loss
(Cross-entropy)

$L$

$Z_i$ are logits (before softmax) for output class $i$

# Carlini-Wagner attack

**Targeted, iterative** attacks based on a new set of loss functions that make the adversarial examples more tricky to detect. Same white-box framework!

$$r = \arg\min\{\mathcal{L}(f(x+r), t_{adv}) + c||r||^2\}$$
$$\text{subject to } |r_i| < \epsilon, \forall i$$

$$\mathcal{L} = \max\{0, \max\{Z(x+r)_k : k \neq t_{adv}\} - Z(x+r)_{t_{adv}}\}$$

Max logit among all output classes but the adversarial target

This term is >0 only if the chosen output is not the adversarial target

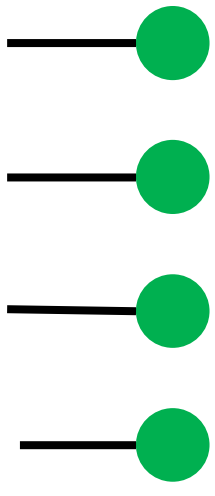


Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." *2017 ieee symposium on security and privacy (sp)*.

# Carlini-Wagner attack (example)

**Example** $r = \arg\min\{\mathcal{L}(f(x+r), t_{adv}) + c||r||^2\}$

$$\mathcal{L} = \max\{0, \max\{Z(x+r)_k : k \neq t_{adv}\} - Z(x+r)_{t_{adv}}\}$$



0.4

0.8 (Adversarial target)

-0.2

7.4 (Original target)

**Beginning of training**

$$\mathcal{L} = \max\{0, \max\{(0.4, -0.2, 7.4)\} - 0.8\}$$
$$= \max\{0, 6.6\} = 6.6$$

# Carlini-Wagner attack (example)

**Example** $r = \arg\min\{\mathcal{L}(f(x+r), t_{adv}) + c||r||^2\}$

$$\mathcal{L} = \max\{0, \max\{Z(x+r)_k : k \neq t_{adv}\} - Z(x+r)_{t_{adv}}\}$$

-0.1

5.4 (Adversarial target)

-0.9

5.3 (Original target)

**After some epochs of training**

$$\mathcal{L} = \max\{0, \max\{(-0.1, -0.9, 5.3)\} - 5.4\}$$
$$= \max\{0, -0.1\} = 0$$

Crossentropy is 0 only when prediction reaches **100% confidence.**

# Targeted FGSM?

Can you devise a strategy (very similar!) to make FGSM a targeted attack?

## Regular FGSM

$$r = \epsilon \ \text{sign} \ \nabla_x \mathcal{L}(f(x + r), t_{true})$$

It is basically one step of **gradient ascent** on the loss function: we want to get away from the correct prediction on the steepest direction.

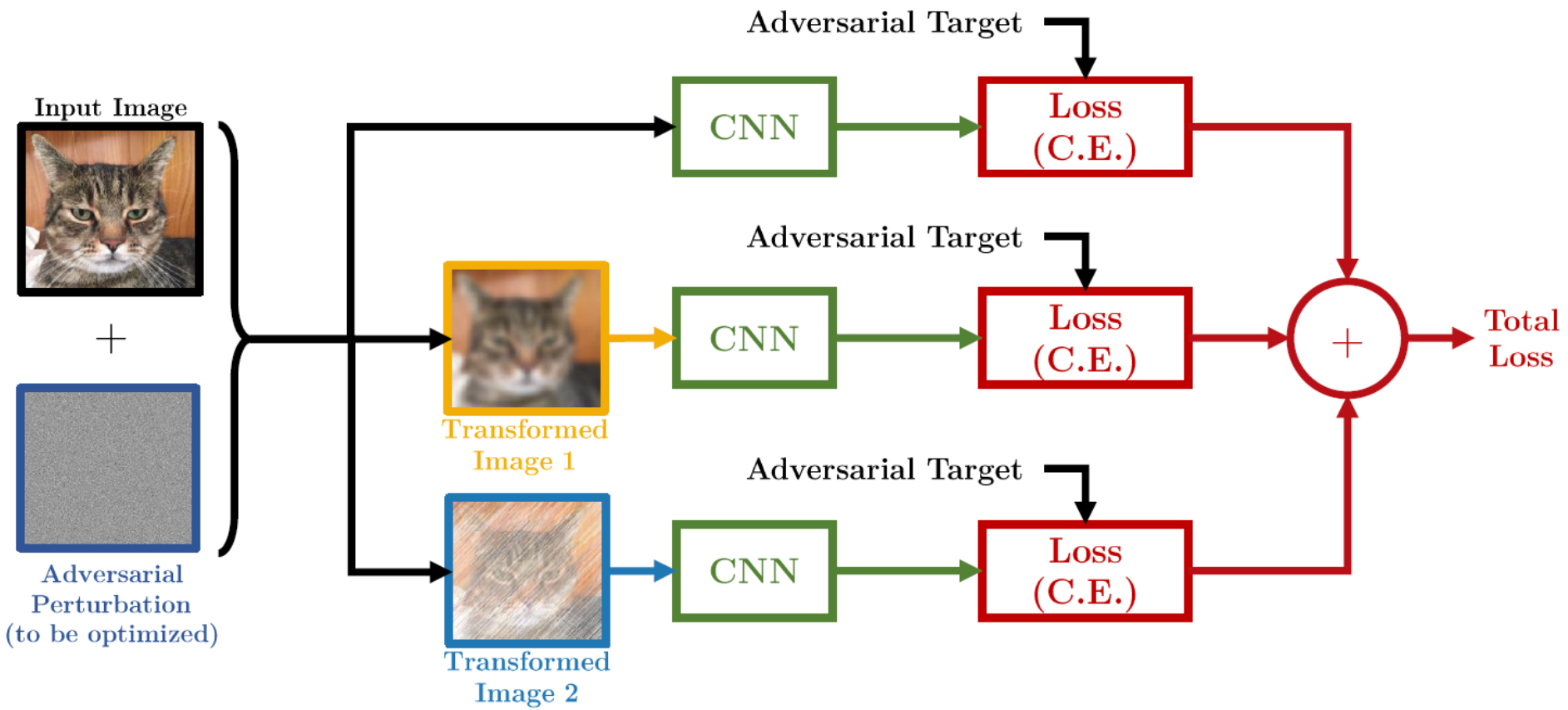
## **Targeted** FGSM

$$r = -\epsilon \ \text{sign} \ \nabla_x \mathcal{L}(f(x + r), t_{adv})$$

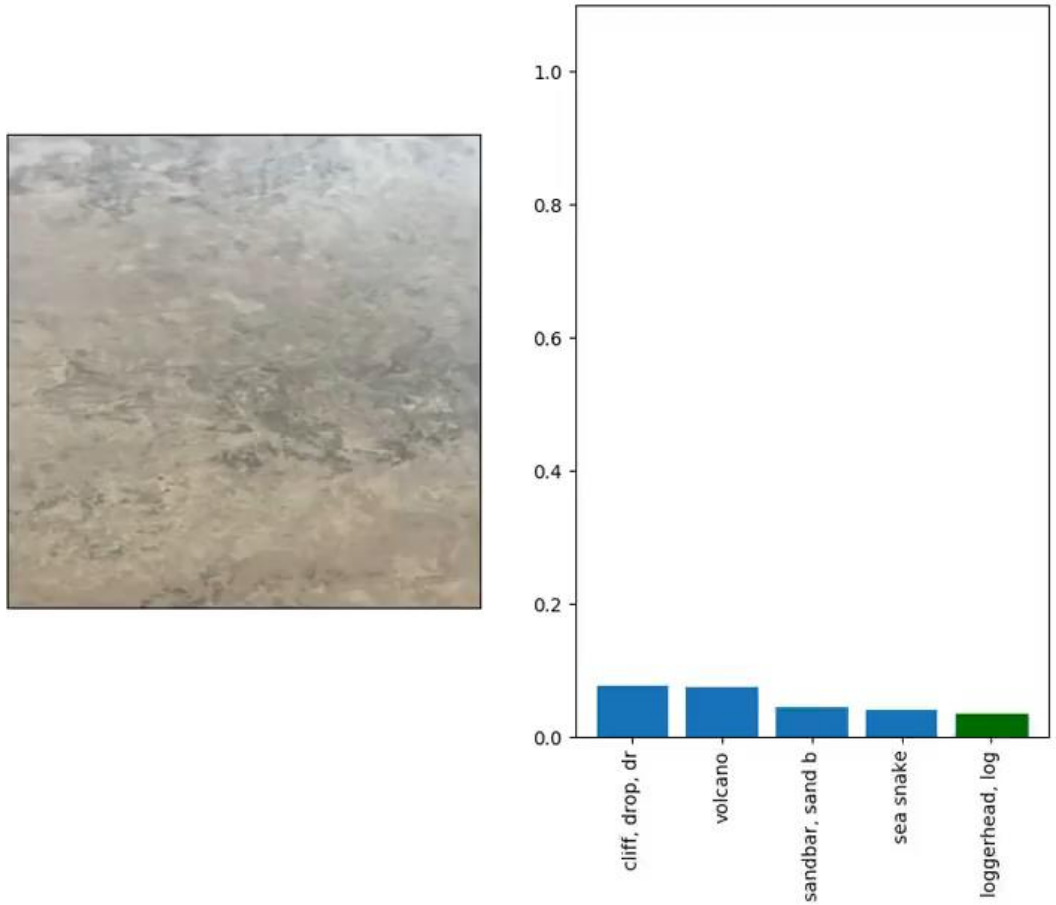
It turns into one step of **gradient descent** on the loss function: we want to get towards that specific wrong prediction.

# Robust AEs

The transformations are randomized during AE craft procedure: «expectation over transformation» makes AE robust to a range of transformations



# Robust AEs

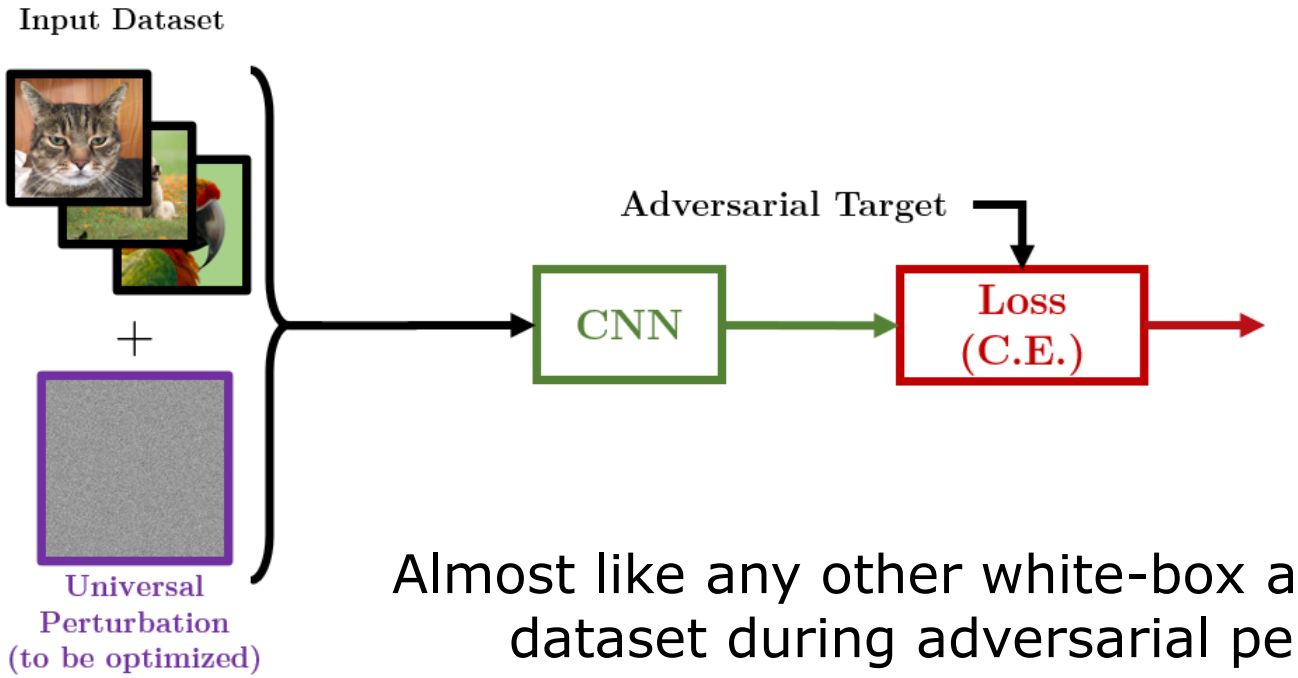


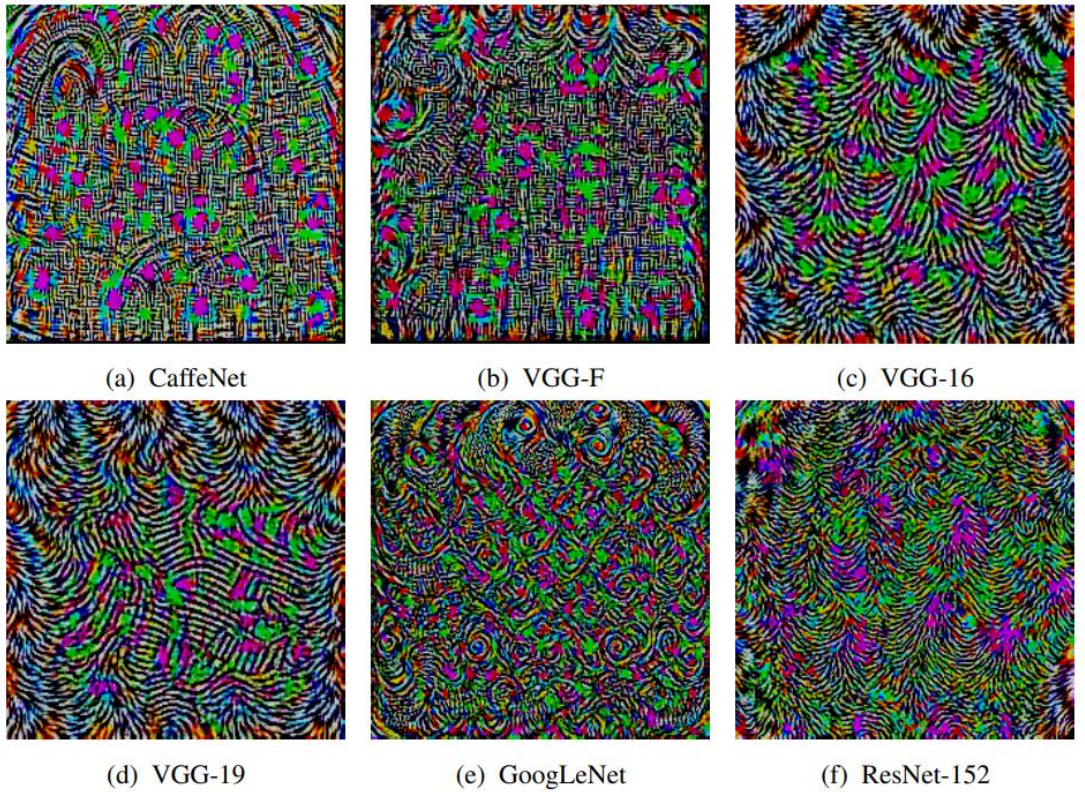

The authors were able to 3D print a turtle that is recognized as a rifle under almost any viewpoint.



Athalye, Anish, and Ilya Sutskever. "Synthesizing robust adversarial examples. arXiv." *arXiv preprint arXiv:1707.07397* (2017).

# Universal Attacks




Almost like any other white-box attack, but we loop on the input dataset during adversarial perturbation craft procedure.

This produces image-agnostic adversarial perturbations.



Moosavi-Dezfooli, S. M., et al. "Universal adversarial perturbations. arXiv 2017." *arXiv preprint arXiv:1610.08401*.
(The original paper uses untargeted attacks)

# Other attacks

Many other different attacks are not considered here. But these works are the foundation to understand white-box adversarial examples. All the others are extensions or combinations of methods.


Evtimov, I., et al. "Robust physical-world attacks on deep learning models. arXiv preprint 2017." *arXiv preprint arXiv:1707.08945*.

Thys, Simen, Wiebe Van Ranst, and Toon Goedemé. "Fooling automated surveillance cameras: adversarial patches to attack person detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

# Outline

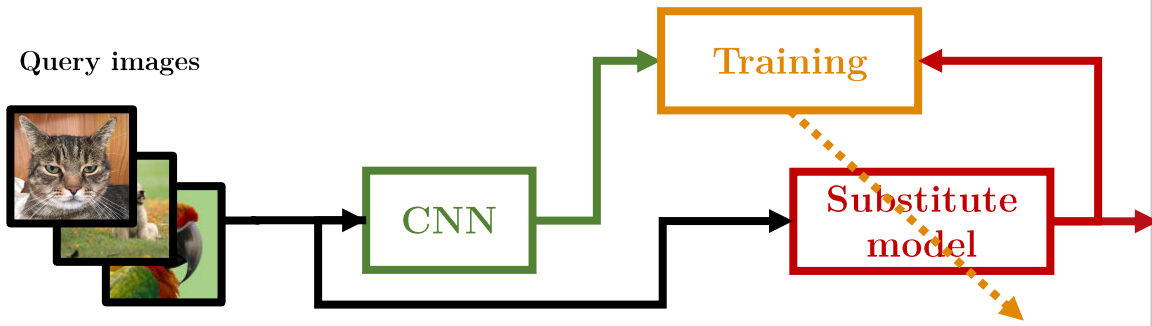
- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

# Black-box attacks

A much more realistic attack framework is when the **model and the parameters are not known to the attacker**. We usually don't get to know the exact model!

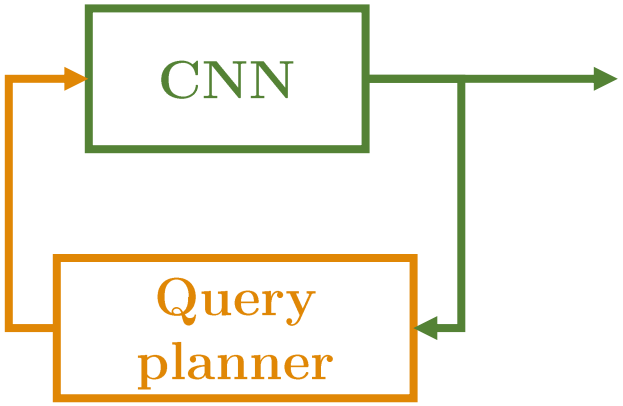
The attacker can only **make queries** on input data $x$, to get $f(x)$ or even simply the predicted class $\hat{c} = \arg\max f(x)$.

# Black-box attacks

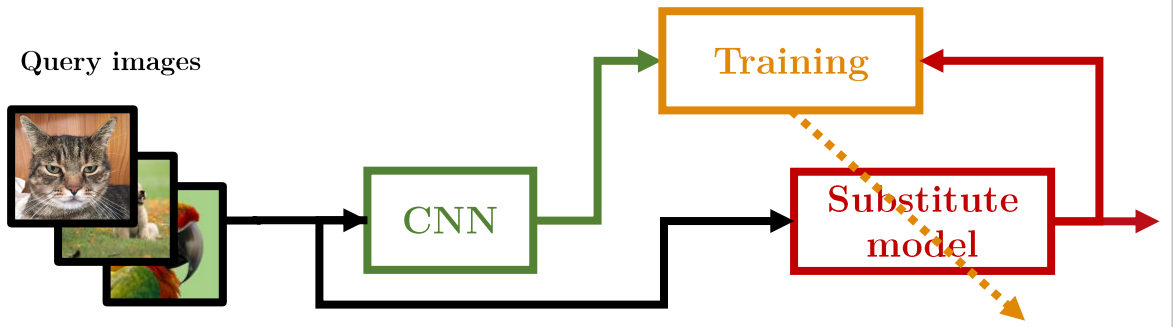



Two main strategies:

1. Train a **substitute model** to craft white-box transferable adversarial examples.
2. **Query Feedback mechanism**



# Substitute model




Main idea: train a substitute model $\hat{f} \approx f$

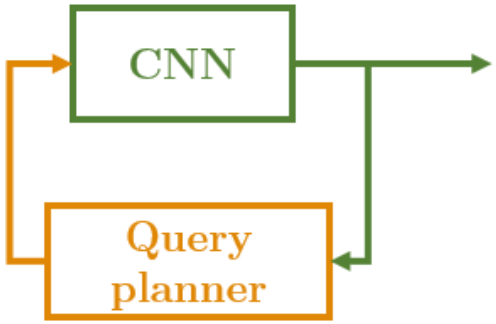
Then craft white-box adversarial examples for $\hat{f}$, exploiting transferability of adversarial examples.

Many papers focus on how to reduce the number of queries, and how to make the substitute model more accurate.



Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017.

# Gradient estimation




Zero-th Order Optimization (ZOO)

1. Define **loss function** (as in white-box case)
2. Estimate **gradient**:

$$\hat{g} \approx \frac{\mathcal{L}(f(x + h\,r), t_{adv}) - \mathcal{L}(f(x - h\,r), t_{adv})}{2h}$$

3. Decide **next** $r$ based on gradient information.



Chen, Pin-Yu, et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models." *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017.

# Gradient estimation

Other works estimate the gradients with different techniques, while the main work around black-box adversarial examples crafting procedures is on query number optimization.

- using **genetic algorithms**
- using **bandits**
- **autoencoders**
- … many others!



Bhambri, Siddhant, et al. "A Survey of Black-Box Adversarial Attacks on Computer Vision Models." *arXiv* (2019): arXiv-1912.

# Outline

- Introduction and Existence of Adversarial Examples
- Taxonomy
- White-box attacks
- Black-box attacks
- Adversarial defenses

# Is it possible to defend from AEs?

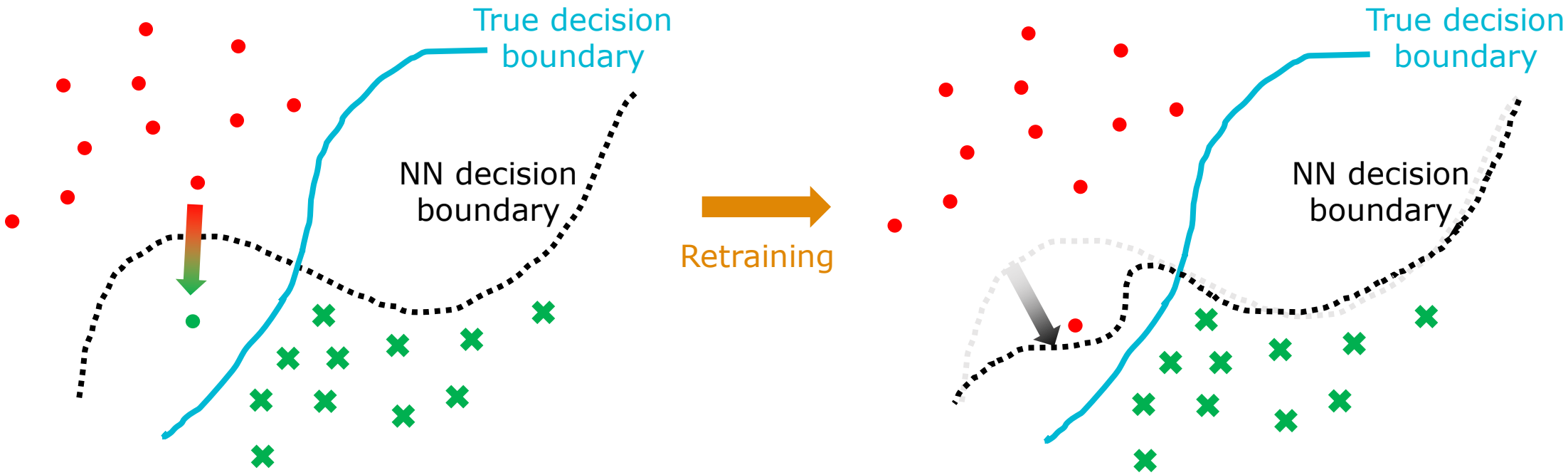
Many defenses against adversarial examples have been proposed. They mainly rely on:

- Modify the **model**
- Modify the **data**
- **Auxiliary tools**

# Adversarial Training

1. Train a network on the dataset $D$
2. Craft **adversarial examples** (with all the attacks that we think our adversary can use) and put them in the dataset $D^{ADV}$ (with TRUE labels).
3. Train the network on the dataset $D' = D \cup D^{ADV}$
4. (Optional) iterate

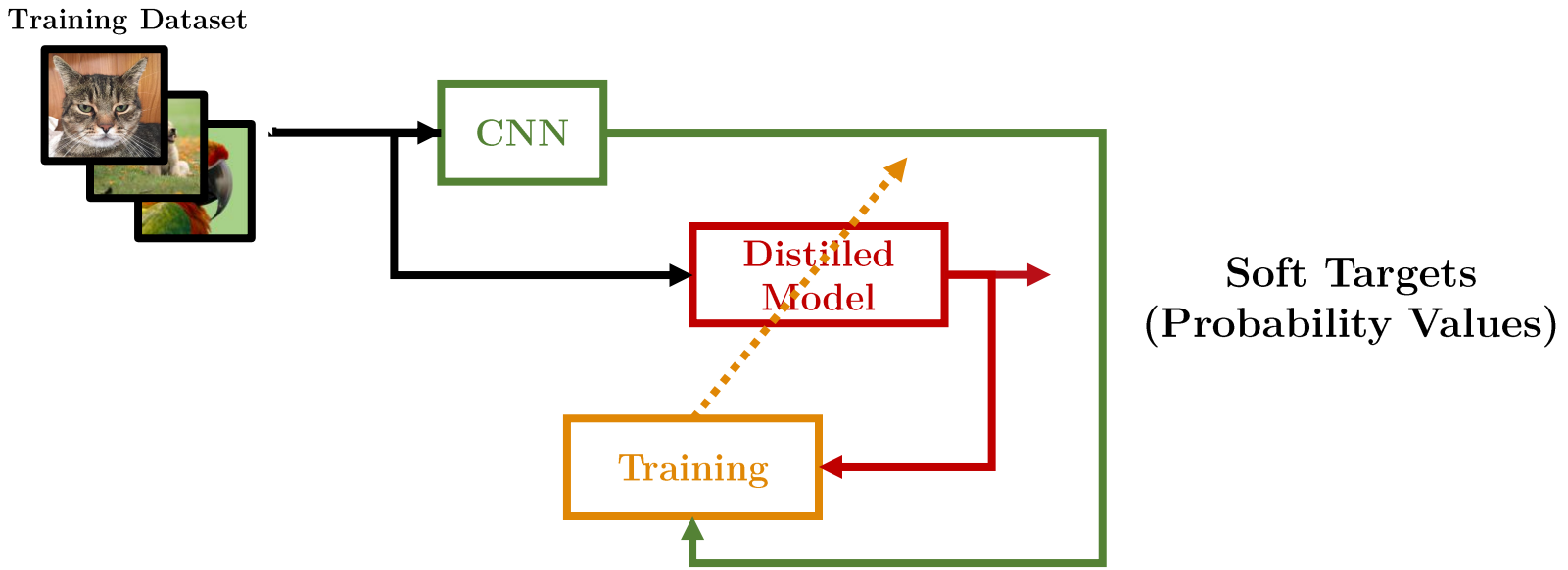
# Adversarial Training




Effect of adversarial training is **regularization**. Main issue: being this in millions of dimensions, it **just makes a little harder** to craft new adversarial examples…

# Defensive Distillation

Uses **network distillation** to increase generalization of the network, while **compressing its size**. It acts as a **regularizer**.






Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016.

# What is distillation?

It is a method to **compress the information** of a neural network into a **smaller network**. The distilled model is trained not with hard labels (one-hot encoded vectors), but with **soft labels** (probability values of the prediction of the original network).

$$y_{soft}(x) = \frac{e^{\frac{Z_i(x)}{T}}}{\sum_{c=1}^{M} e^{\frac{Z_c(x)}{T}}}$$

Distillation Temperature (>1)

$$Z(x) = \begin{bmatrix} -0.1 \\ 0.4 \\ 2.1 \\ 4.8 \end{bmatrix}$$

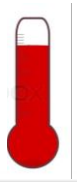
$T = 1$ (regular softmax)

$$y_{soft}(x) = \begin{bmatrix} 0{,}007 \\ 0{,}011 \\ 0{,}062 \\ 0{,}920 \end{bmatrix}$$

$T = 2$

$$y_{soft}(x) = \begin{bmatrix} 0.059 \\ 0.076 \\ 0.178 \\ 0.687 \end{bmatrix}$$

$T \to \infty$



$$y_{soft}(x) = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$



Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

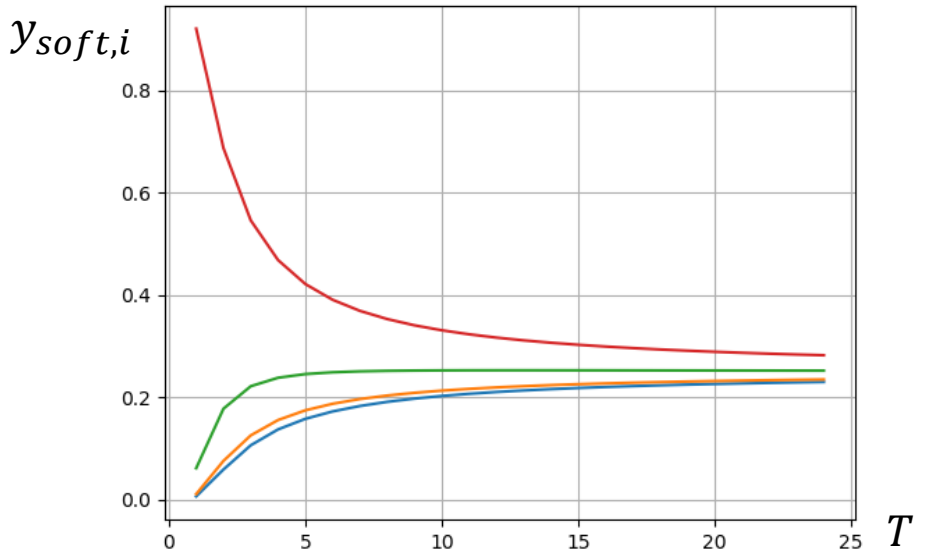
# What is distillation?

It is a method to **compress the information** of a neural network into a **smaller network**. The distilled model is trained not with hard labels (one-hot encoded vectors), but with **soft labels** (probability values of the prediction of the original network).

$$y_{soft}(x) = \frac{e^{\frac{Z_i(x)}{T}}}{\sum_{c=1}^{M} e^{\frac{Z_c(x)}{T}}}$$

Distillation Temperature (>1)

$$Z(x) = \begin{bmatrix} -0.1 \\ 0.4 \\ 2.1 \\ 4.8 \end{bmatrix}$$



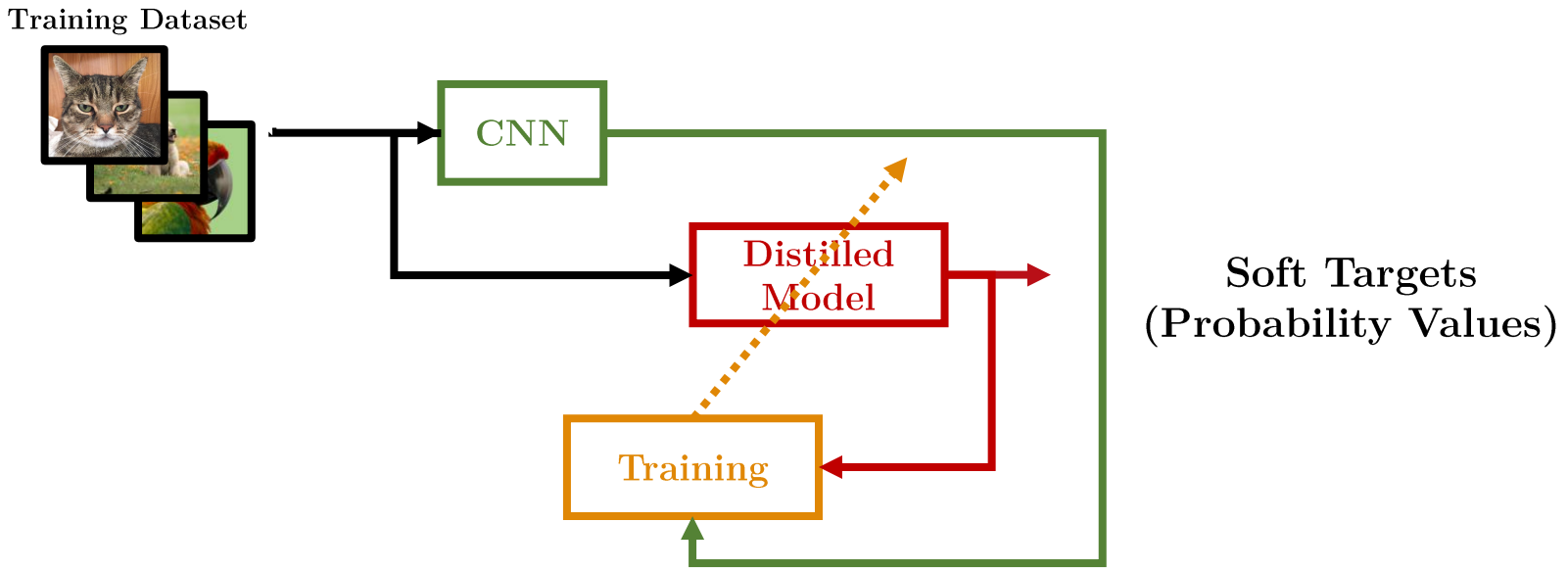




Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

# Defensive Distillation

Distillation helps with generalization of the network (regularization), that makes just more difficult the search for adversarial examples…

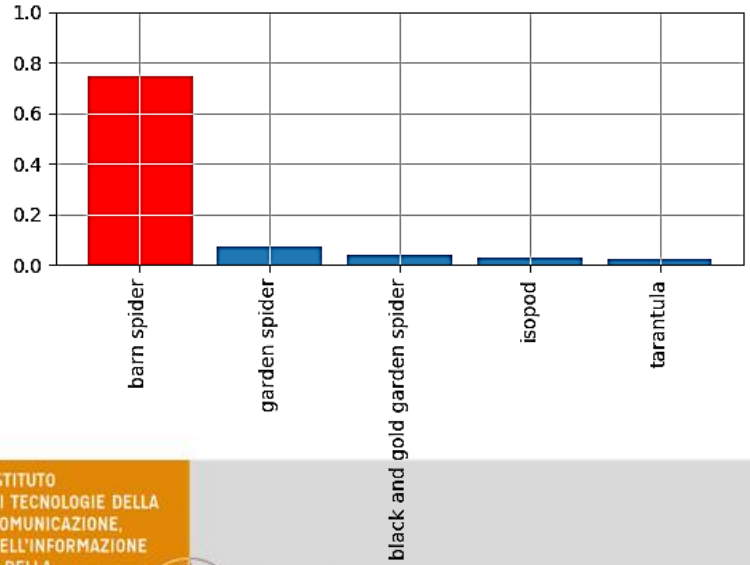





Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016.

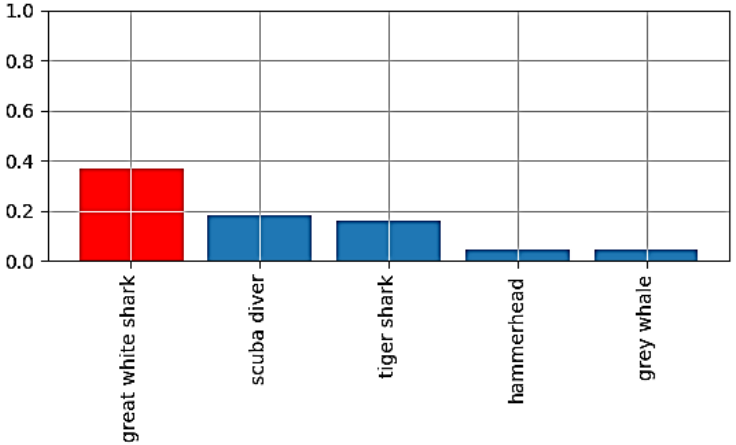
# Input Transformations

**Small Translation**

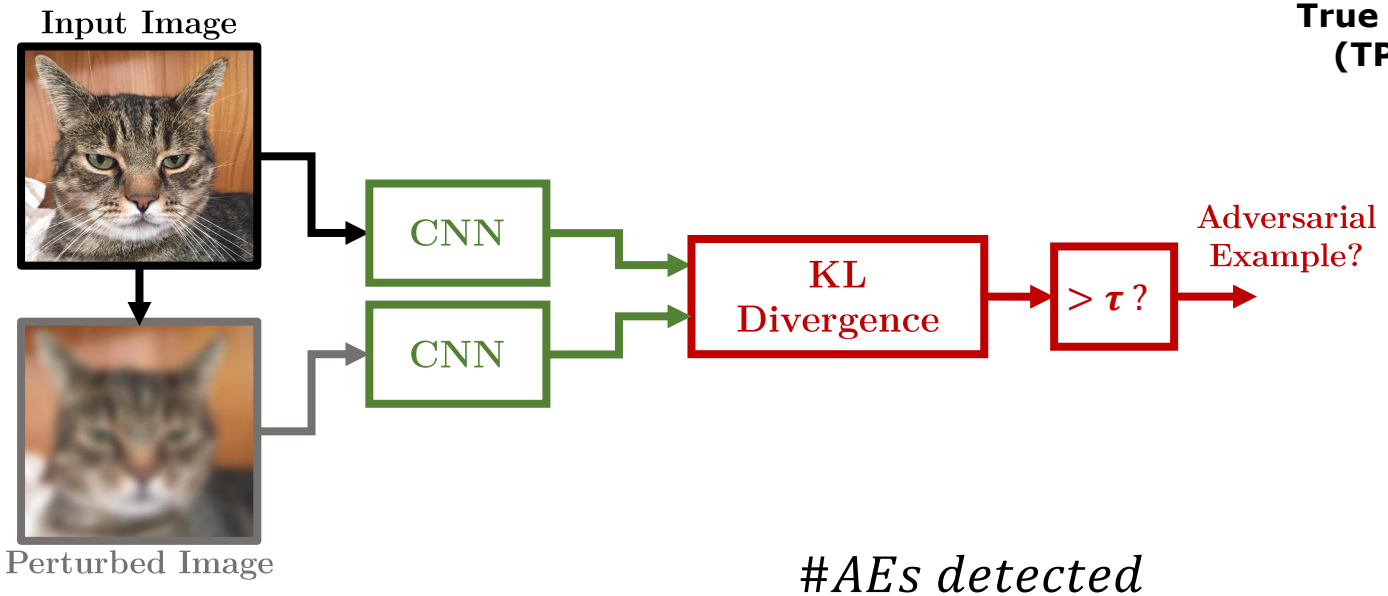



**Other transformations:**

- Rotation
- Mirror
- Shear
- Blur
- Gaussian noise



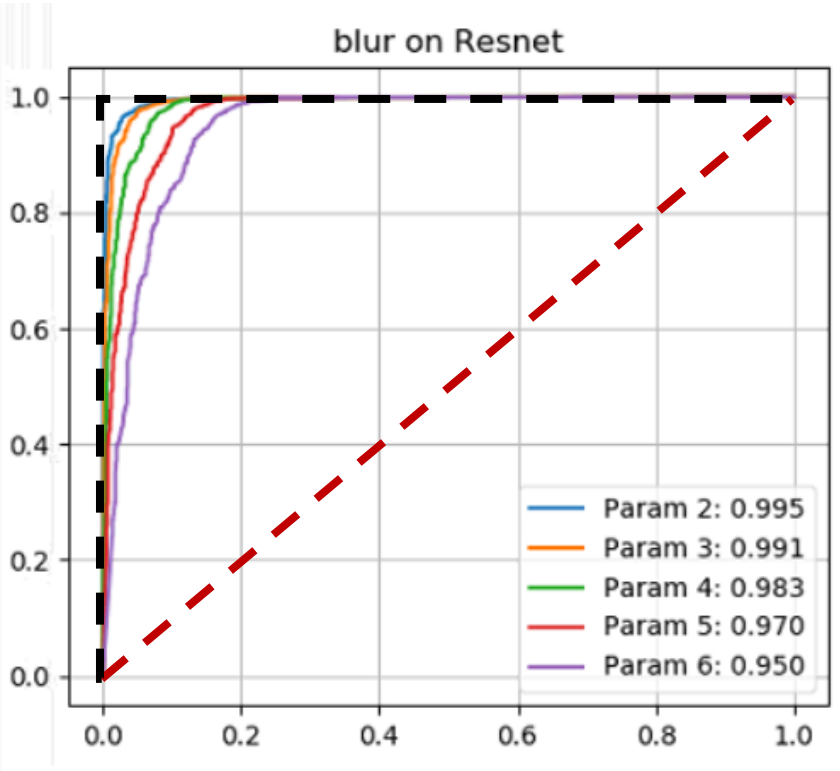
# Adversarial detection




$$TP = \frac{\#AEs\ detected}{\#AEs} \rightarrow 1$$

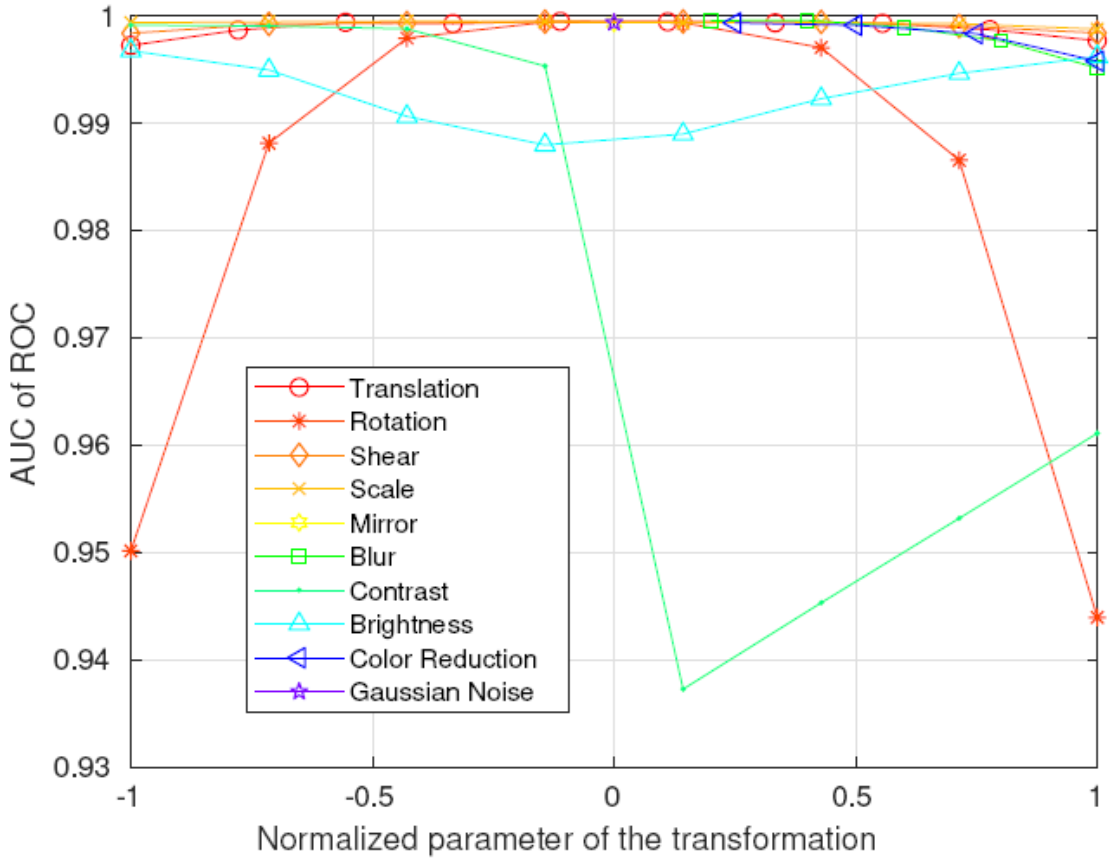
$$FP = \frac{\#noAEs\ detected}{\#noAEs} \rightarrow 0$$

**Receiver Operating Characteristic (ROC) graph**




**Baseline performance (random): (0.5)**
**Best achievable (perfect): (1)**

# Input Transformations

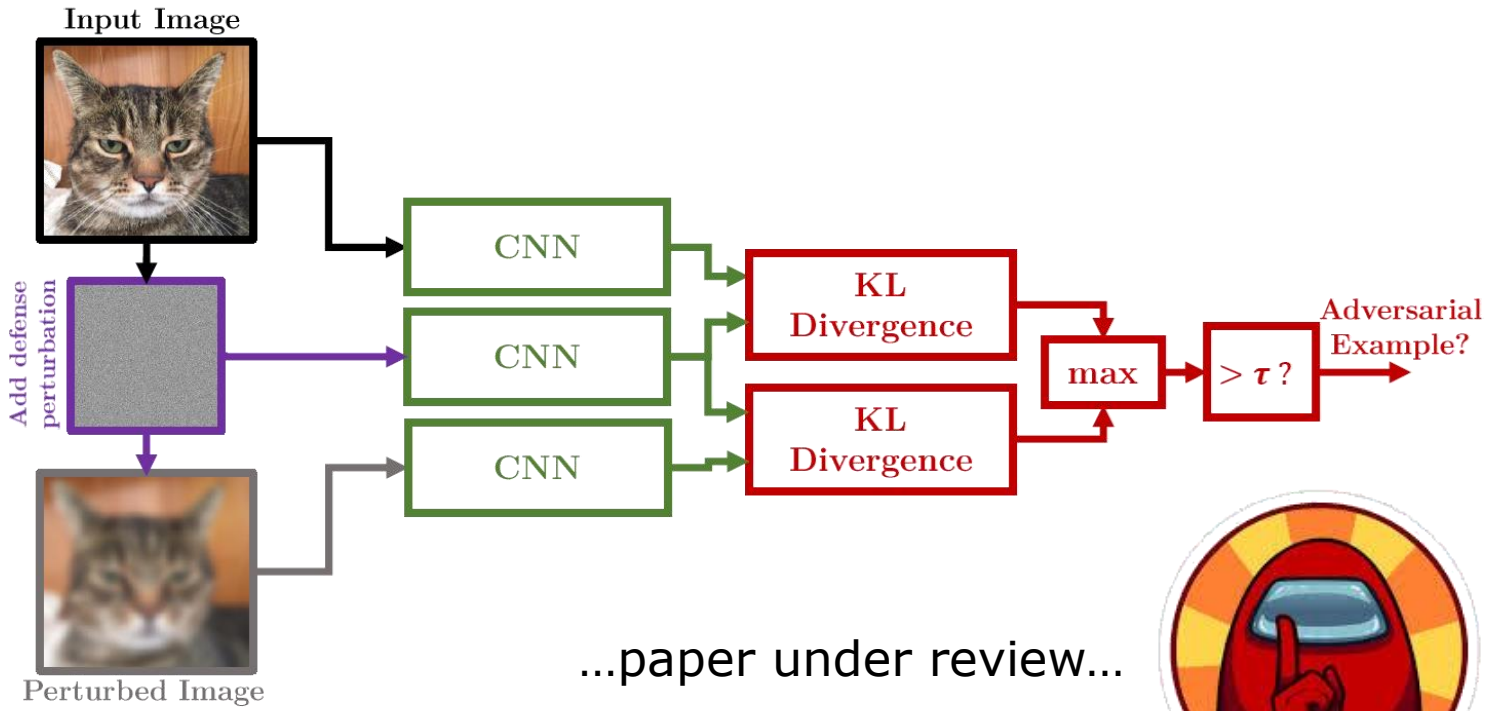



Detection performance (Area Under Curve of ROC graph) of several input transformations against L-BFGS adversarial examples.

# Input Transformations against robust AEs

Robust adversarial examples are not detected with input transformations…

The idea is to craft a **defence perturbation** that is able to **remove the «robustness»** from robust adversarial examples. In this way, we can use the input transformations again!




…paper under review…

# Other works…

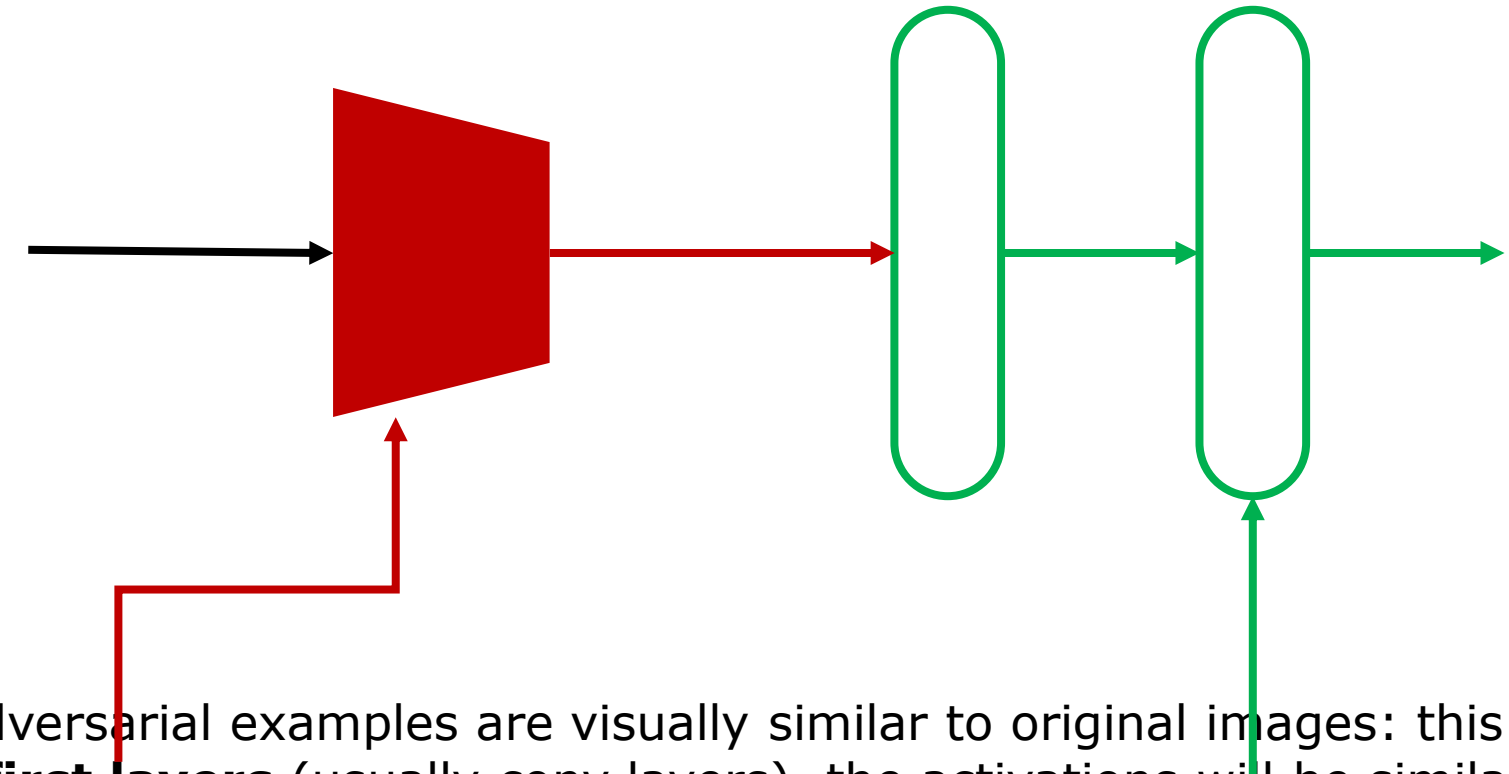
- MagNet

Meng, Dongyu, and Hao Chen. "Magnet: a two-pronged defense against adversarial examples." *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017.

- PixelDefend

Song, Yang, et al. "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples." *arXiv preprint arXiv:1710.10766* (2017).

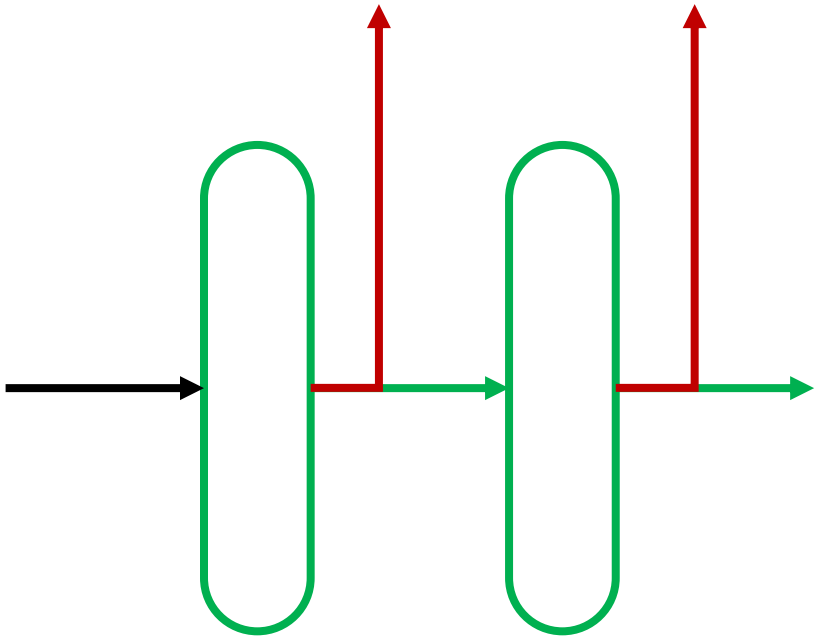
# Clustering of network activations

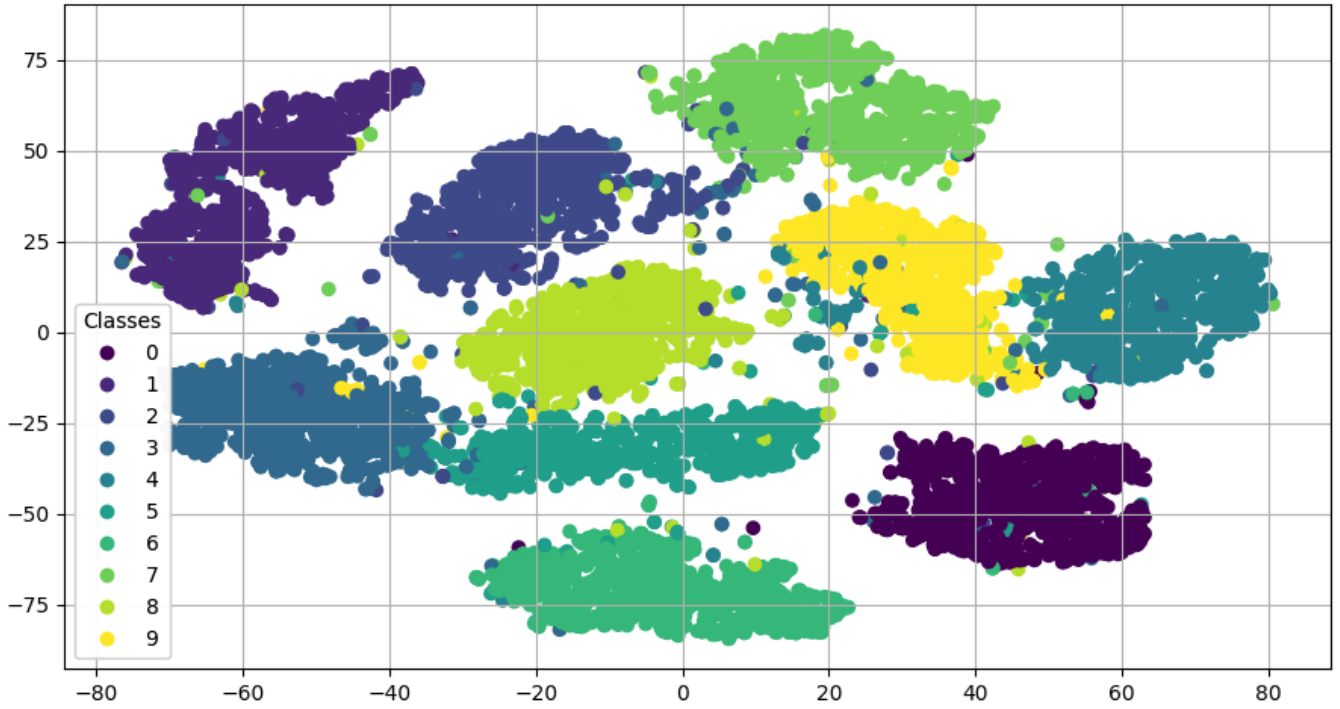


Adversarial examples are visually similar to original images: this means that in the **first layers** (usually conv layers), the activations will be similar to the **original image**, while in the **last layer** the **adversarial power** will be effective.

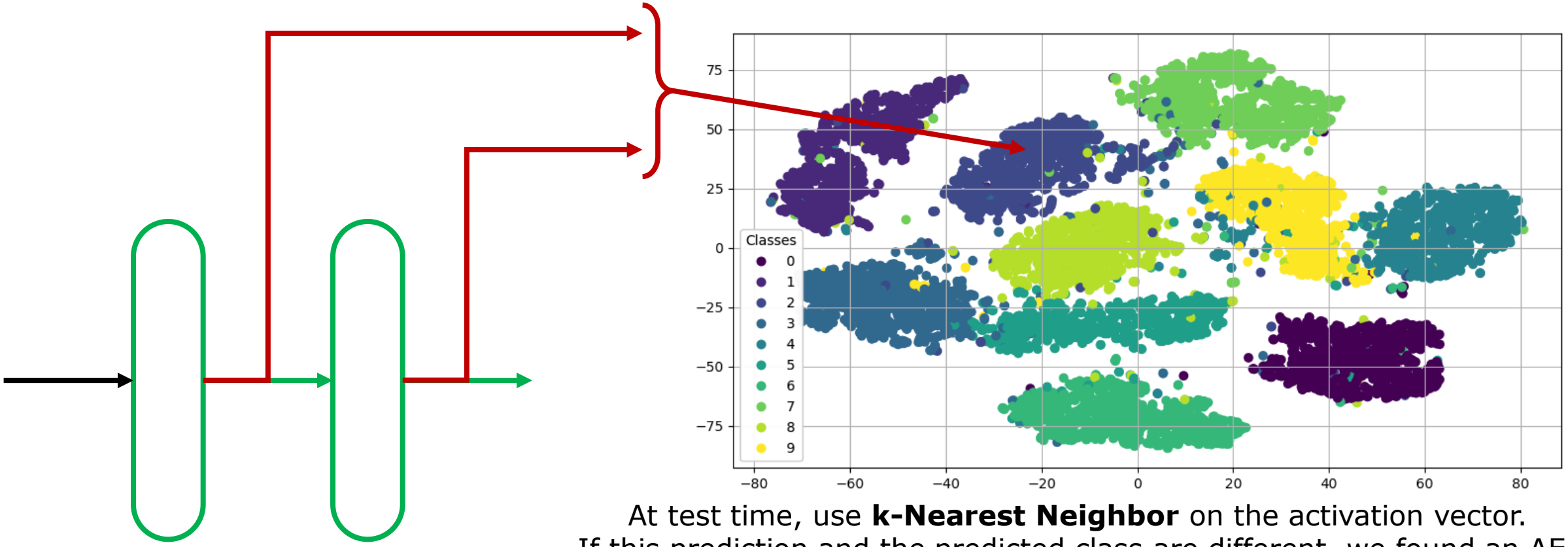
# Clustering of network activations

Collect activations from **training inputs** and save them in a database.





# Clustering of network activations




At test time, use **k-Nearest Neighbor** on the activation vector.
If this prediction and the predicted class are different, we found an AE!

# Clustering of network activations

Preliminary results: **MNIST** dataset, 2-hidden-layer **fully connected network** (64 neurons per layer)

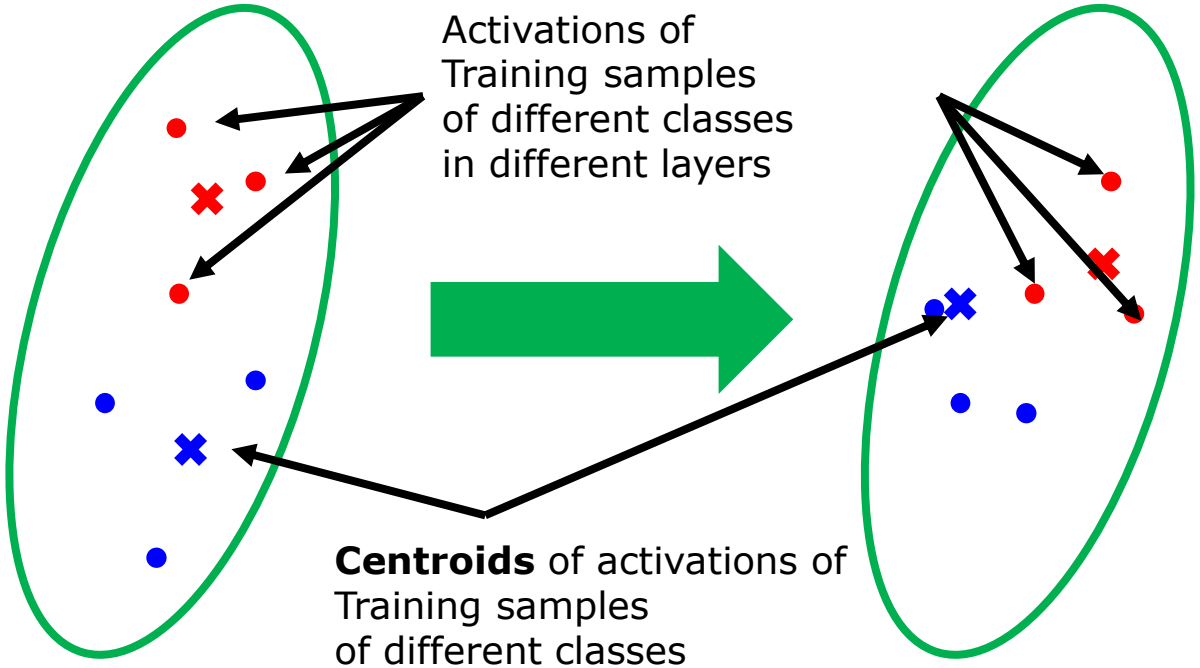
Considering only **FIRST HIDDEN LAYER** activation: **81.8% AEs detected**, 2.9% false positives.

Considering both **FIRST AND SECOND HIDDEN LAYERS** activation: **52.8% AEs detected**, 2.3% false positives (the last layer ruins the performance).

To be tried on convolutional neural network, but seems promising (did not search extensively in the literature though).

# Clustering of network activations

Other idea: encode the behavior of the network in a different way.




Some distance

$$B_f(x) = \begin{bmatrix} d(x^*_{l_1}, \bar{x}^{c_1}_{l_1}) & \cdots & d(x^*_{l_n}, \bar{x}^{c_M}_{l_n}) \\ \vdots & \ddots & \vdots \\ d(x^*_{l_1}, \bar{x}^{c_M}_{l_1}) & \cdots & d(x^*_{l_n}, \bar{x}^{c_M}_{l_n}) \end{bmatrix}$$

Activations for test input

# Clustering of network activations

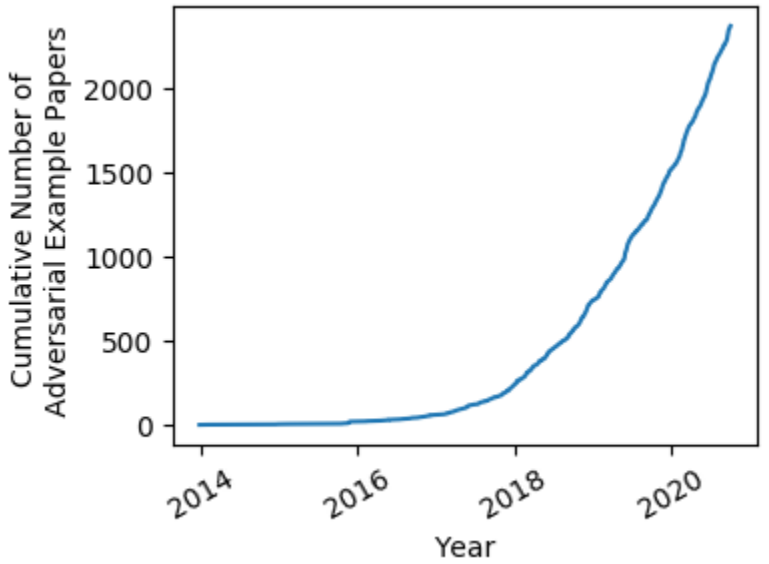
Similar works already out:

- Papernot, Nicolas, and Patrick McDaniel. "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning." *arXiv preprint arXiv:1803.04765* (2018).
- Carrara, Fabio et al. (2019). Adversarial Examples Detection in Features Distance Spaces

# Conclusions

Adversarial examples, if crafted considering the right assumptions, must be considered a serious threat for real AI-based systems.

Papers in the adversarial examples literature have gone exponential in the last years, so it is difficult to keep track of them.




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

# Conclusions

At the moment the attackers have a great advantage against the defenders. This can be due to:

- «Offense sells tickets, but defense wins championships»
- Attacks are easier to craft. Slight changes to the assumptions, to the method, to the optimization, …
- No actual general counter-measure has been found to protect from adversarial examples… Have we understood what AEs are and how neural networks work?

So… Still an incredibly florid and open topic!

(links beautifully with XAI)