Midterm 4
ON DETECTING ADVERSARIAL PERTURBATIONS

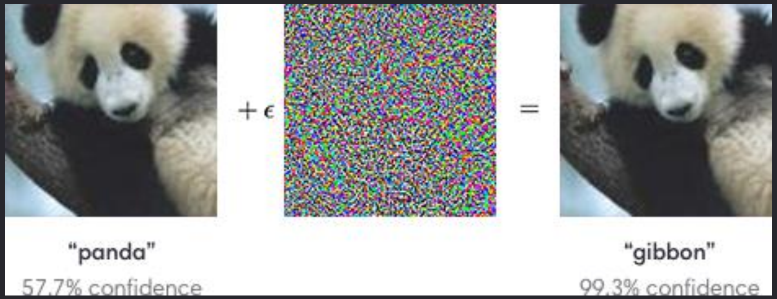
Intelligent Systems for Pattern Recognition
Master Degree in Computer Science, AI Curriculum
A.Y. 2020/2021

Alessandro Ristori

Introduction

Over the last years deep learning models obtained **impressive performances** in different complex tasks, above all:

- **Image classification** (Russakovsky et al., 2015; He et al., 2016) ;
- **Speech recognition** (Amodei et al., 2016);



Small and carefully directed perturbations are still the main concern for those tasks. The so called **adversarial examples** have been in the spotlight in the last years and many techniques have been developed recently for their implementation and to defend against them.

"The vulnerability to adversarial inputs can be problematic and even prevent the application of deep learning methods in safety and security-critical applications, the problem is particularly severe when human safety is involved."

Many methods have been proposed to improve the robustness of networks against such attacks:

- **Augmenting the training data** with adversarial examples (Goodfellow et al., 2015);
- **Applying JPEG compression** to the input (Dziugaite et al., 2016);
- **Distilling a hardened network** from the original classifier network (Papernot et al., 2016);

The paper under examination proposes and shows the training of a **binary detector to discriminate between non-adversarial and adversarial examples**, moreover it displays the generalization and transferability powers of some attacks and, finally, shows how an attack against both the classifier and detector can be developed and how to defend from it.

Adversarial attacks used

Fast method: The applied perturbation is the **direction** in image space which yields the **highest increase of the cost function**.

$$x^{adv} = x + \varepsilon \, \mathrm{sgn}(\nabla_x J_{cls}(x, y_{true}(x)))$$

- x input image;
- $y_{true}(x)$ one-hot encoding of the true class;
- $J_{cls}\left(x_n^{adv}, y_{true}(x)\right)$ cost function of the classifier;
- $\sigma \in [0,1]$ hyperparameter that works as a tradeoff;
- $J_{det}(x, 1)$ cost function of the detector.

Basic iterative mehod: Iterative version of the fast-method.

$$x_0^{adv} = x \qquad x_{n+1}^{adv} = Clip_x^{\varepsilon}\{x_n^{adv} + \alpha \, \mathrm{sgn}(\nabla_x J_{cls}\left(x_n^{adv}, y_{true}(x)\right))\} \qquad L_\infty$$

$$x_0^{adv} = x \qquad x_{n+1}^{adv} = Project_x^{\varepsilon}\{x_n^{adv} + \alpha \frac{\nabla_x J_{cls}\left(x_n^{adv}, y_{true}(x)\right)}{||\nabla_x J_{cls}\left(x_n^{adv}, y_{true}(x)\right)||}\} \qquad L_2$$

DeepFool method: It iteratively perturbs an image x, at each step the classifier is linearized around the n-th x and the closest class boundary is determined. The **attack stops when the actual class given by the classifier changes**. The paper uses both the L_∞ and L_2 norm.

*"In the worst case, an adversary might not only have access to the classification network and its gradient **but also to the adversary detector and its gradient**. In this case, the adversary might potentially generate inputs to the network that fool both the classifier (i.e., **get classified wrongly**) and fool the detector (i.e., **look innocuous**)."*

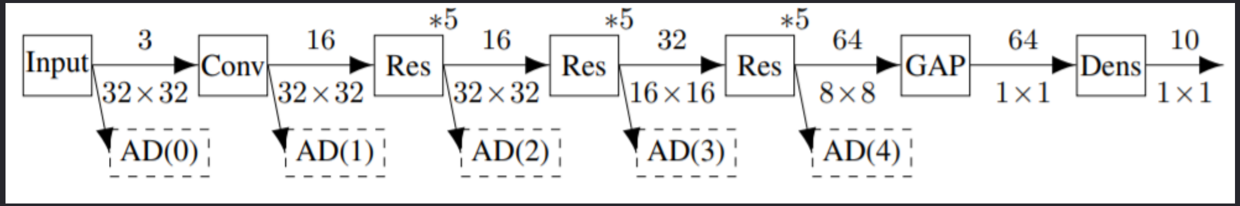
Dynamic attack:

$$x_0^{adv} = x \qquad x_{n+1}^{adv} = Clip_x^{\varepsilon}\{x_n^{adv} + \alpha\left((1-\sigma)\mathrm{sgn}(\nabla_x J_{cls}\left(x_n^{adv}, y_{true}(x)\right)\right) + \sigma \, \mathrm{sgn}\left(\nabla_x J_{det}\left(x_n^{adv}, 1\right)\right))\}$$

Training and test with Cifar-10

Classifier:

- **Epochs:** 100;
- **Momentum:** 0.9 (starting);
- **Learning rate:** 0.1, 0.01 (after 41 epochs), 0.001 (after 61 epochs);
- Uses **stochastic gradient descent**; **performance are evaluated after each epoch**
- Network on the validation data.

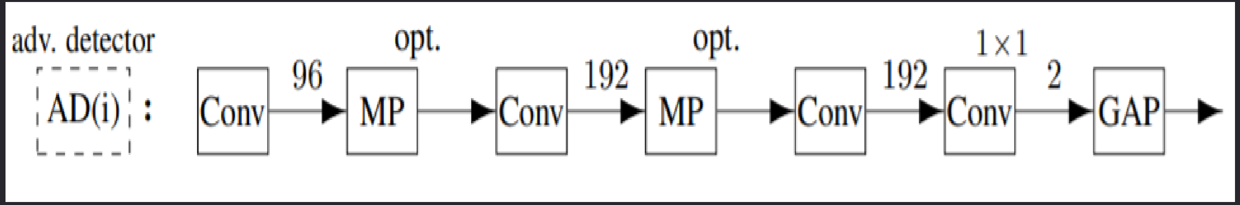


Accuracy on test set: 91,3%

*First we **train the classifier** on the regular dataset;*
*Then **we augment the initial dataset by generating an adversarial example for each image**, doubling the size of the dataset;*
Freeze the weights** of the classifier and **train the detector.

Detector:

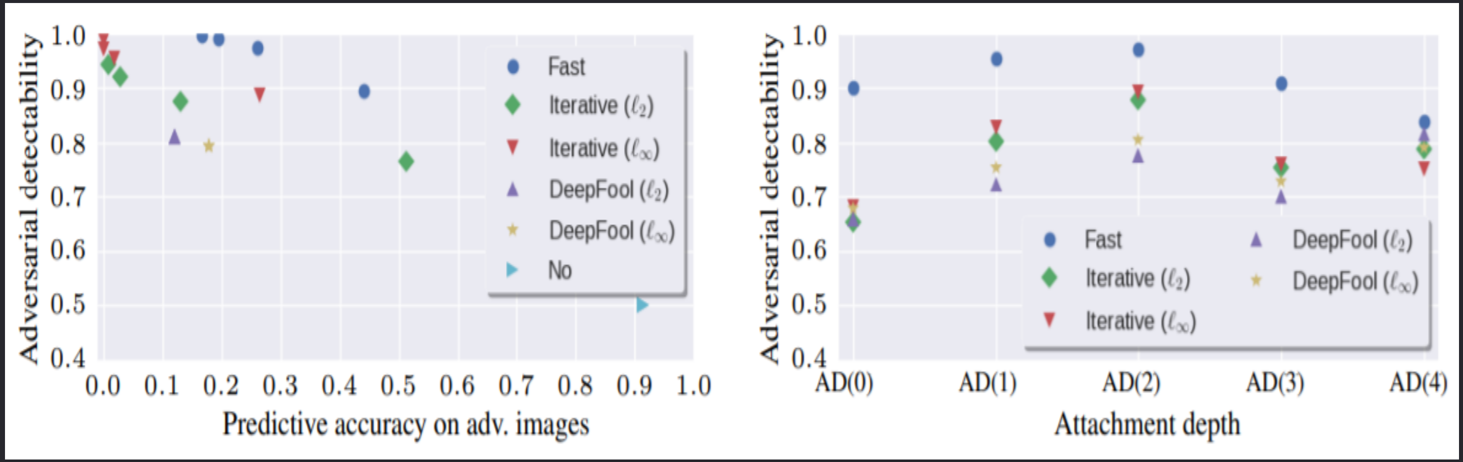
- **Epochs:** 20;
- **Optimizer:** Adam;
- **Learning rate:** 0.0001;
- β1 = 0.99, β2 = 0.999;
- The detector was **attached to position AD(2) for each adversary attack,** except for DeepFool where the detector was **attached on AD(4).**

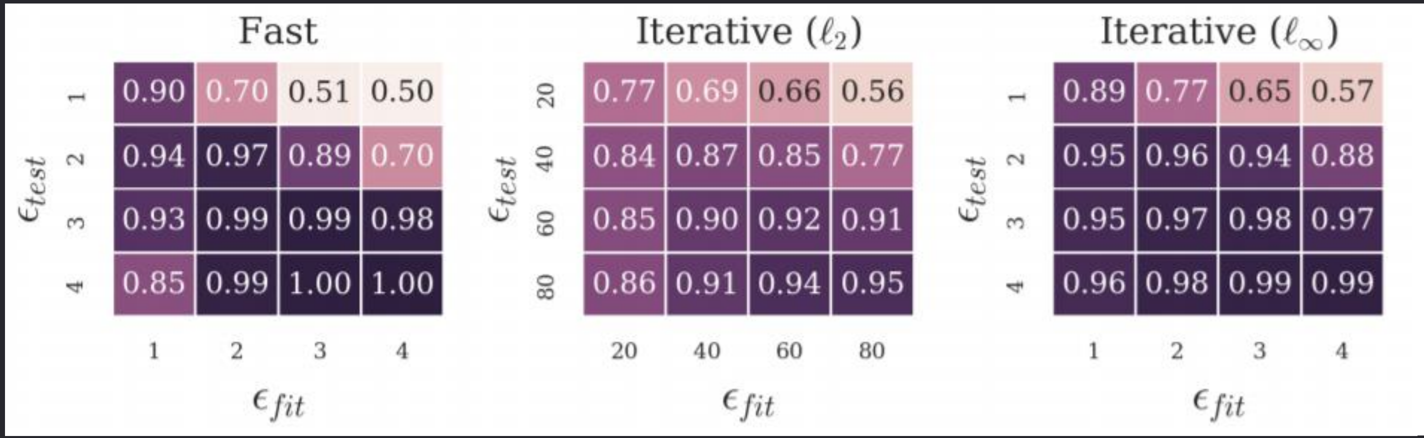


Results on Cifar-10

- **The “Fast” adversary:** weak;
- **DeepFool based methods:** relatively strong adversaries;
- **“Iterative” methods:** somewhere in-between.

For the **“Fast” and “Iterative” adversaries AD(2) is the best,** meanwhile **AD(4) works best for DeepFool.**





Fast

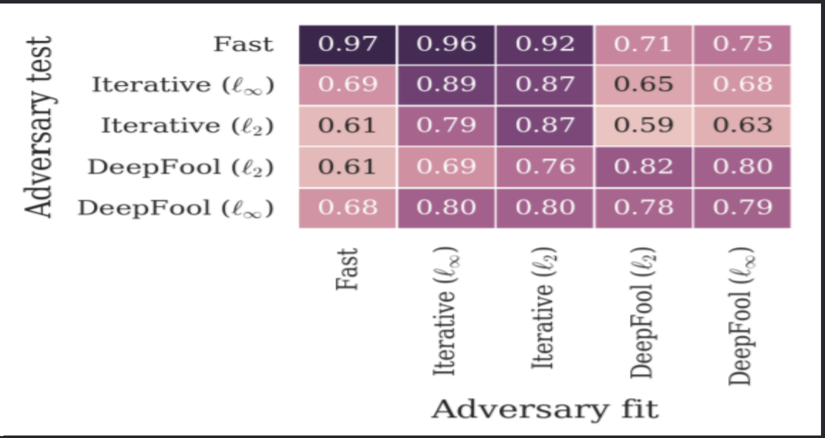
ϵ_{test} \ ϵ_{fit}	1	2	3	4
1	0.90	0.70	0.51	0.50
2	0.94	0.97	0.89	0.70
3	0.93	0.99	0.99	0.98
4	0.85	0.99	1.00	1.00

Iterative (ℓ_2)

ϵ_{test} \ ϵ_{fit}	20	40	60	80
20	0.77	0.69	0.66	0.56
40	0.84	0.87	0.85	0.77
60	0.85	0.90	0.92	0.91
80	0.86	0.91	0.94	0.95

Iterative (ℓ_∞)

ϵ_{test} \ ϵ_{fit}	1	2	3	4
1	0.89	0.77	0.65	0.57
2	0.95	0.96	0.94	0.88
3	0.95	0.97	0.98	0.97
4	0.96	0.98	0.99	0.99



Adversary test \ Adversary fit	Fast	Iterative (ℓ_∞)	Iterative (ℓ_2)	DeepFool (ℓ_2)	DeepFool (ℓ_∞)
Fast	0.97	0.96	0.92	0.71	0.75
Iterative (ℓ_∞)	0.69	0.89	0.87	0.65	0.68
Iterative (ℓ_2)	0.61	0.79	0.87	0.59	0.63
DeepFool (ℓ_2)	0.61	0.69	0.76	0.82	0.80
DeepFool (ℓ_∞)	0.68	0.80	0.80	0.78	0.79

Transferability on **same adversary with different ε** and on **different adversaries**

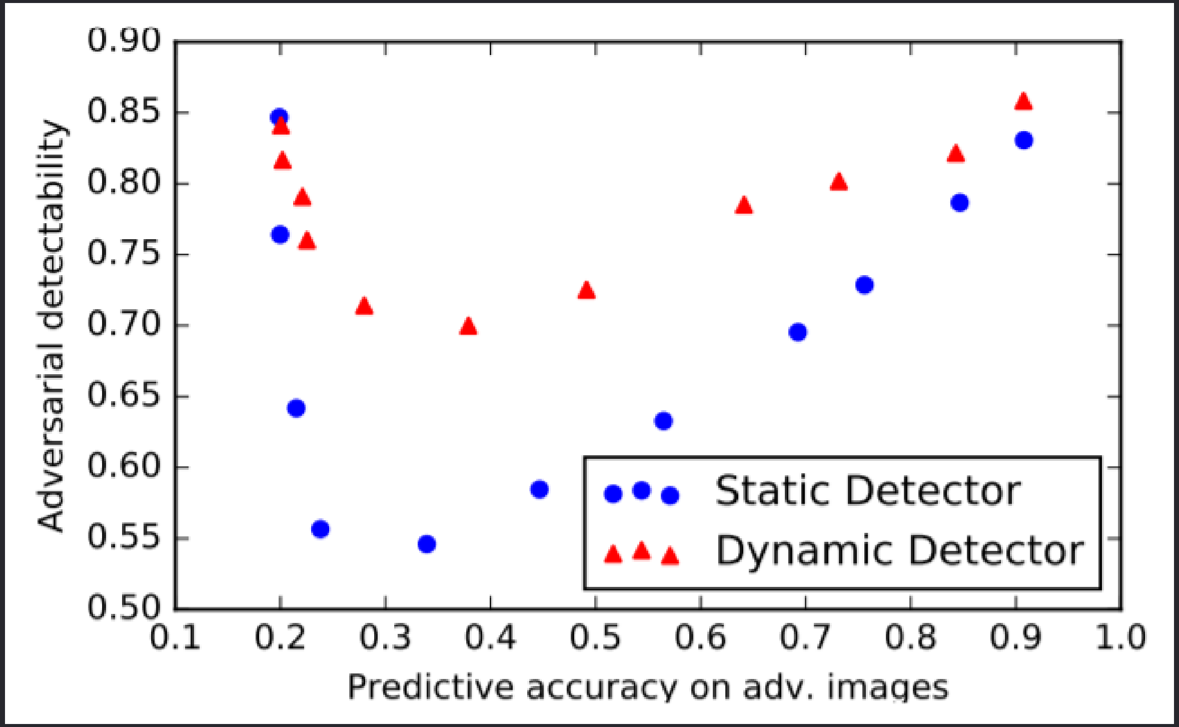
Dynamic detector on Cifar-10

Dynamic adversary training

Instead of **precomputing a dataset of adversarial examples,** we **compute the adversarial examples on-the-fly for each mini-batch** and let the adversary modify each data point with probability 0.5.

Note that a **dynamic adversary will modify a data point differently every time it encounters the data point since it depends on the detector's gradient** and the detector changes over time.

By training the detector in this way, **we implicitly train it to resist dynamic adversaries** for various values of σ.

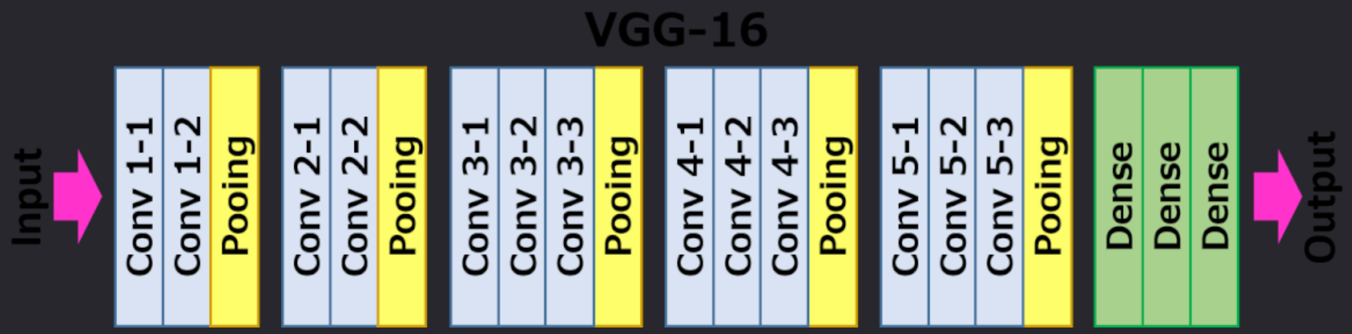


*As we can see from the image, **static detectors performed worse than dynamic ones** (as expected) on dynamic adversarial attacks, the parameter σ has been chosen as σ ∈{0.0, 0.1, . . . , 1.0}, with smaller values of σ corresponding to lower predictive accuracy, i.e., being further on the left.*

10-class ImageNet

Classifier:

- **Pretrained VGG16** (Simonyan & Zisserman, 2015);
- Before the SoftMax another layer is placed so that we can **restrict the classification to only ten classes.**

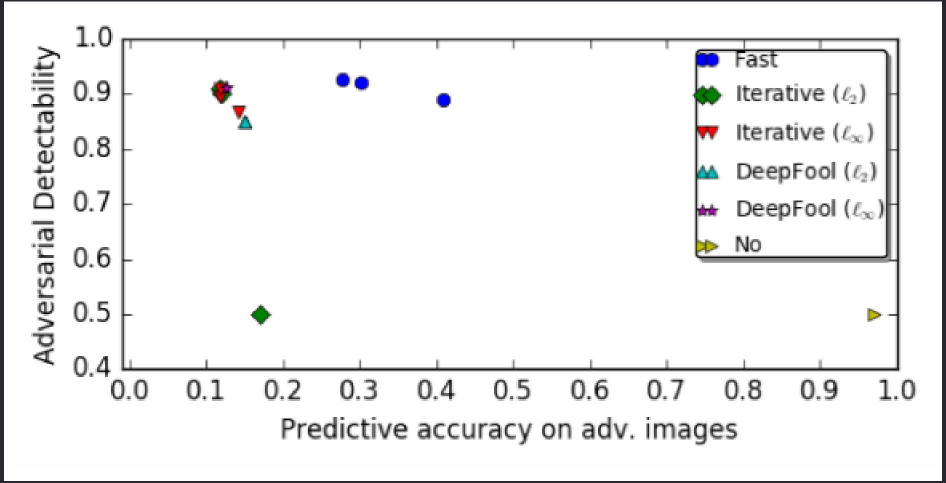


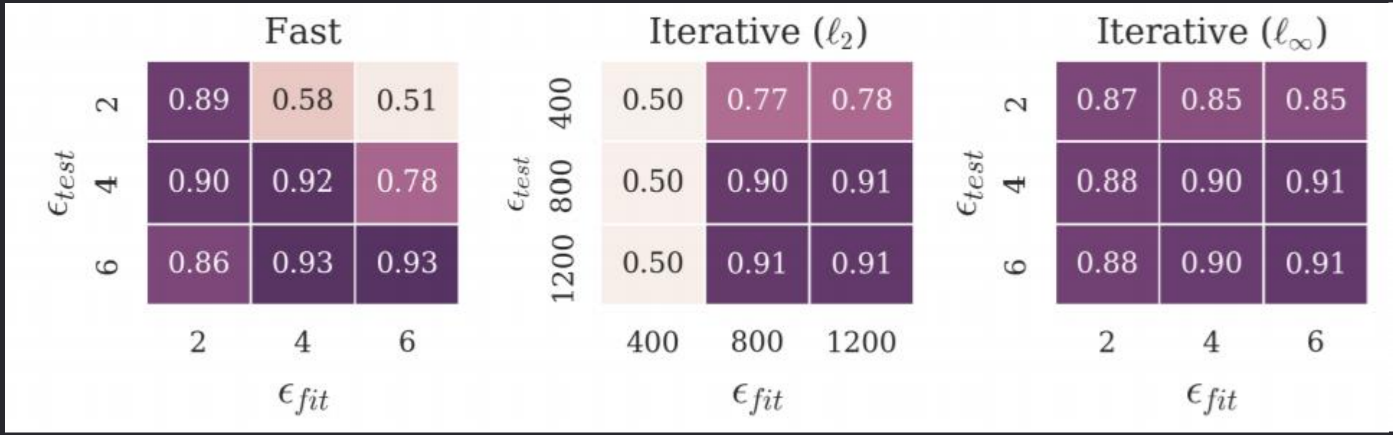
Detector:

- **Epochs:** 500;
- **Optimizer:** Adam;
- **Learning rate:** 0.0001;
- β1 = 0.99, β2 = 0.999;
- The detector **was attached after the fourth max pooling layer;**
- The network consists of a **sequence of five 3x3 convolutions** with 196 feature maps each using **batch-normalization** and **rectified linear units**, followed by a 1x1 convolution which maps onto the 10 classes, **global-average pooling**, and a **softmax layer.** Another 2x2 max-pooling layeris inserted after the first convolution.

Results on 10-class ImageNet

The image shows that **detectability is 85% percent or more** with the exception of the “Iterative” `2-based adversary with ε = 400. For this adversary, the detector only reaches chance level.





Fast (ϵ_{test} rows, ϵ_{fit} columns)

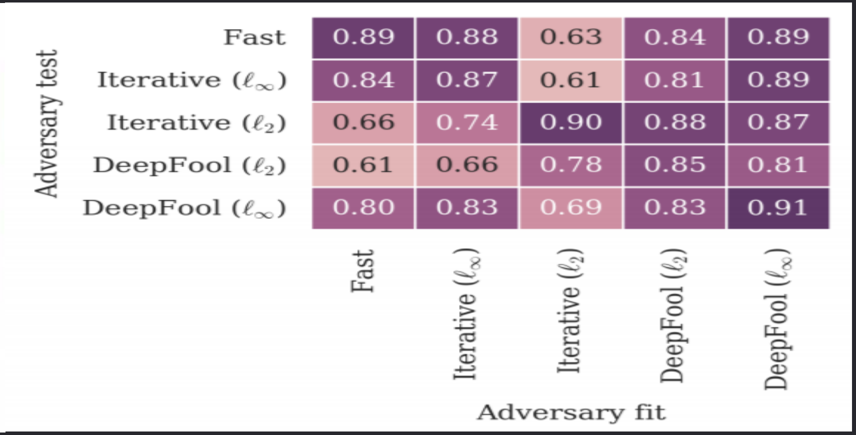
ϵ_{test} \ ϵ_{fit}	2	4	6
2	0.89	0.58	0.51
4	0.90	0.92	0.78
6	0.86	0.93	0.93

Iterative (ℓ_2)

ϵ_{test} \ ϵ_{fit}	400	800	1200
400	0.50	0.77	0.78
800	0.50	0.90	0.91
1200	0.50	0.91	0.91

Iterative (ℓ_∞)

ϵ_{test} \ ϵ_{fit}	2	4	6
2	0.87	0.85	0.85
4	0.88	0.90	0.91
6	0.88	0.90	0.91



Adversary test \ Adversary fit	Fast	Iterative (ℓ_∞)	Iterative (ℓ_2)	DeepFool (ℓ_2)	DeepFool (ℓ_∞)
Fast	0.89	0.88	0.63	0.84	0.89
Iterative (ℓ_∞)	0.84	0.87	0.61	0.81	0.89
Iterative (ℓ_2)	0.66	0.74	0.90	0.88	0.87
DeepFool (ℓ_2)	0.61	0.66	0.78	0.85	0.81
DeepFool (ℓ_∞)	0.80	0.83	0.69	0.83	0.91

Transferability on **same adversary with different ε** (similar to cifar-10) and on **different adversaries** (deepfool generalizes better)

Conclusions

What was shown:

- **Image-dependent perturbations** are sufficiently regular to be detectable.
- **Transferability** is not symmetric and typically **works best between similar adversaries and from stronger to weaker adversary;**
- For a **static detector**, there might be areas which are **adversarial to both classifier and detector**, this will be a (small) subset of the areas which are adversarial to the classifier alone. However, a **dynamic detector is considerably harder to fool:** on one hand, it might further **reduce the number of areas which are both adversarial to classifier and detector;**

What the paper achieved:

Shown that **adversarial examples can be detected surprisingly well using a detector subnetwork, this does not directly allow classifying adversarial examples correctly**, but it allows mitigating adversarial **attacks against machine learning systems** by resorting to **fallback solutions.**

Future work:

Additional future work will be developing **stronger adversaries that are harder to detect** by adding effective randomization. Finally, **developing methods for training detectors** such that they can **detect many different kinds of attacks reliably at the same time** would be essential for safety and security-related applications.

Thanks for the attention