

# Lecture 8: Intro to Adversarial Attacks

Siddharth Garg  
sg175@nyu.edu

- ▶ State-of-the-art deep neural networks have achieved near human-level performance in image classification tasks.
- ▶ However, deep neural networks have been shown to be susceptible to **adversarial input attacks**.
- ▶ A small, intentionally chosen perturbation added to a correctly classified image can mislead the classifier to output a totally different label, even though the perturbation is small enough that it appears imperceptible to humans.

## Motivation Example



Figure 1: Clean and adversarial images with different prediction labels.

Adversarial input attacks can be broadly classified into two types, one is non-targeted attack and the other is targeted attack.

- ▶ **Non-targeted attack:**

Aiming to fool the neural network and output a label different than the original one.

- ▶ **Targeted attack:**

Intentionally misleading the network to output a specific label designed by the attacker.

- ▶ E.g. a face recognition system for security entrance control:

- ▶ Non-targeted attacks could lead to denial of legal access.
- ▶ Targeted attacks bring the jeopardy of illegal entrance.

## Adversary's Objective

Given an image  $x$  with a classification label  $y = \text{classifier}(x)$ , where  $\text{classifier}$  is the function of the neural network. The attacker aims to:

- ▶ find an image  $x'$  whose classification label is  $y'$ , such that  $y' = \text{classifier}(x') \neq y$ .
- ▶ ensure  $\|x' - x\| \leq \delta$ , where  $\delta$  is an upper bound of the distortion from  $x$  to  $x'$ .

- Non-targeted and targeted FGS methods are expressed as in Equation 2 and Equation 3:

$$x' \leftarrow \text{clip}(x + \epsilon \text{sign}(\nabla \ell_{F, y^*}(x))) \quad (2)$$

$$x' \leftarrow \text{clip}(x - \epsilon \text{sign}(\nabla \ell_{F, y'}(x))) \quad (3)$$

Here  $\epsilon$  is a small constraint scalar,  $\ell$  refers to the loss function and  $\text{clip}(x)$  ensures each pixel value falls in the setting range.

Iterative fast gradient sign methods:

- ▶ FGS methods have been extended to iterative versions, naming IFGS, that perturb each pixel with a small amount for multiple times.
- ▶ IFGS methods for non-targeted and targeted attacks are shown in Equation 4 and Equation 5:

$$x'_0 = x, \quad x'_{N+1} \leftarrow \text{clip}_\epsilon(x'_N + \alpha \text{sign}(\nabla \ell_{F,y^*}(x))) \quad (4)$$

$$x'_0 = x, \quad x'_{N+1} \leftarrow \text{clip}_\epsilon(x'_N - \alpha \text{sign}(\nabla \ell_{F,y'}(x))) \quad (5)$$

- ▶ IFGS methods are capable of generating adversarial inputs with smaller distortion when compared to basic FGS methods.

**Carlini and Wagner:** [https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7958570&casa\\_token=p2rOKd\\_n1IQAAAAA:efZbm8j3b7A7IAVBllhs9M-J3rVXhscAnzFwFa7n9zq\\_lvmnmla9OZLHn6cNizC4kyUr7G6\\_JuQ](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7958570&casa_token=p2rOKd_n1IQAAAAA:efZbm8j3b7A7IAVBllhs9M-J3rVXhscAnzFwFa7n9zq_lvmnmla9OZLHn6cNizC4kyUr7G6_JuQ)

- ▶ CW method is capable of making much smaller perturbation than previous attacks to fool the network.
- ▶ When the pixel values are quantized to form valid inputs, the newly generated images often fail to mislead the network with the specific target label.
- ▶ A greedy search on the lattice defined by the discrete neighbor integers is essential after optimization.



# Projected Gradient Descent (PGD)

- Used when optimizing a function with constraints

$$\min_x f(x) \quad \text{subject to} \quad x \in C$$

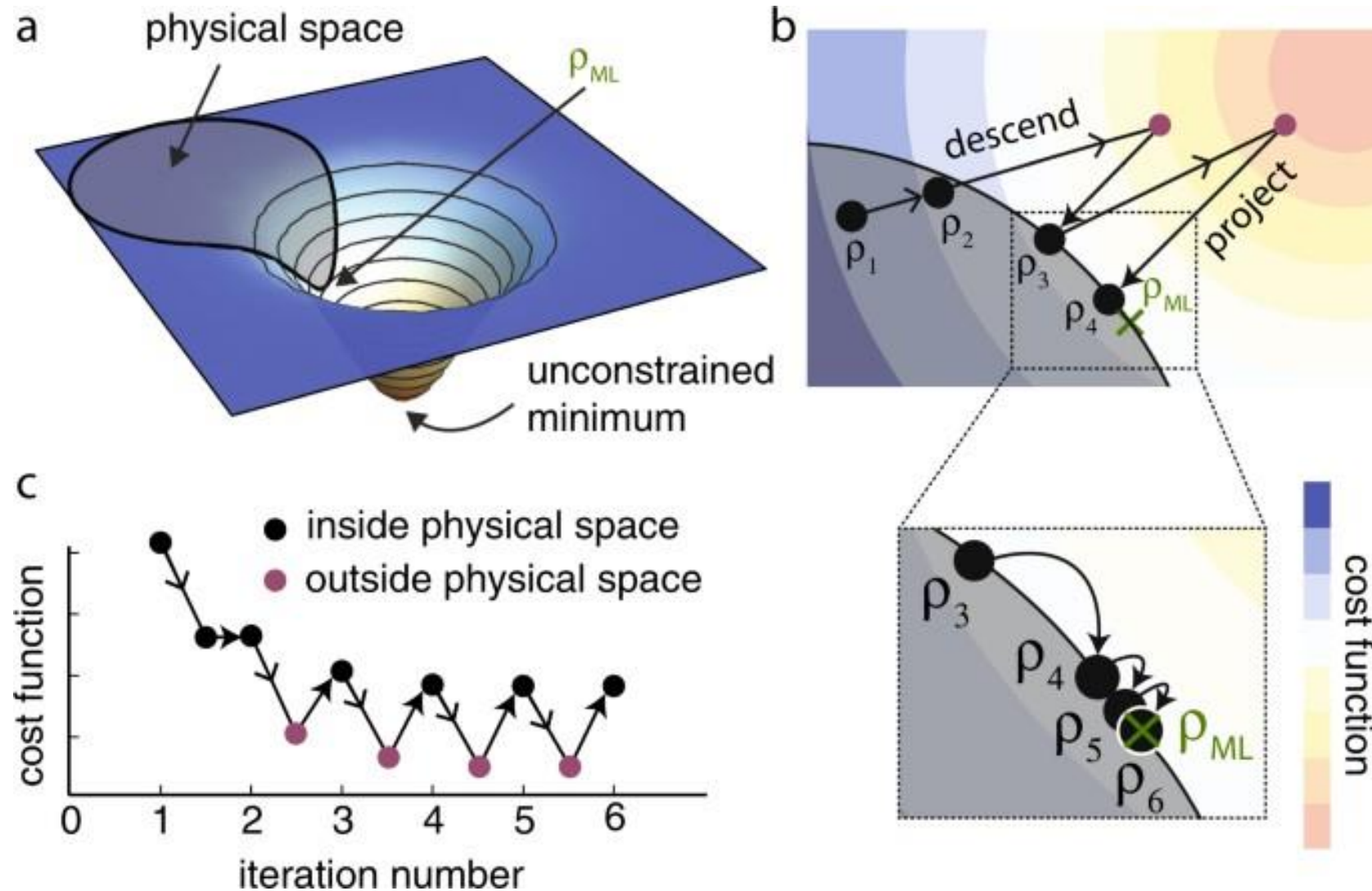
where  $f$  is convex and smooth, and  $C$  is convex. Recall projected gradient descent uses an initial  $x^{(0)}$ , and then updates for  $k = 1, 2, 3, \dots$  by first performing gradient descent on the then current solution and then projecting it back onto the constraint set. This can be expressed as

$$x^{(k)} = P_C(x^{(k-1)} - t_k \nabla f(x^{(k-1)}))$$

where  $P_C$  is the projection operator onto the set  $C$ . Special case of proximal gradient, motivated by local

# PGD Visualized

<https://www.nature.com/articles/s41534-017-0043-1>



## Attacks Comparison































adv. label	1	9	5	4	3	4	7	8	1	1
FGS										
IFGS										
CW										

Figure 3: MNIST adversarial images generated by FGS, IFGS, CW, and their prediction labels.

# Adversarial Perturbations Applied to FaceRec

- Attacks implemented on Face Recognition DNN implemented by Parkhi et al.
  - “Dodging” = Untargeted attack
  - “Impersonation” = Targeted attack



Figure 2: A dodging attack by perturbing an entire face. Left: an original image of actress Eva Longoria (by Richard Sandoval / CC BY-SA / cropped from <https://goo.gl/7QUvRq>). Middle: A perturbed image for dodging. Right: The applied perturbation, after multiplying the absolute value of pixels' channels  $\times 20$ .



Figure 3: An impersonation using frames. Left: Actress Reese Witherspoon (by Eva Rinaldi / CC BY-SA / cropped from <https://goo.gl/a2sCdc>). Image classified correctly with probability 1. Middle: Perturbing frames to impersonate (actor) Russel Crowe. Right: The target (by Eva Rinaldi / CC BY-SA / cropped from <https://goo.gl/AO7QYu>).



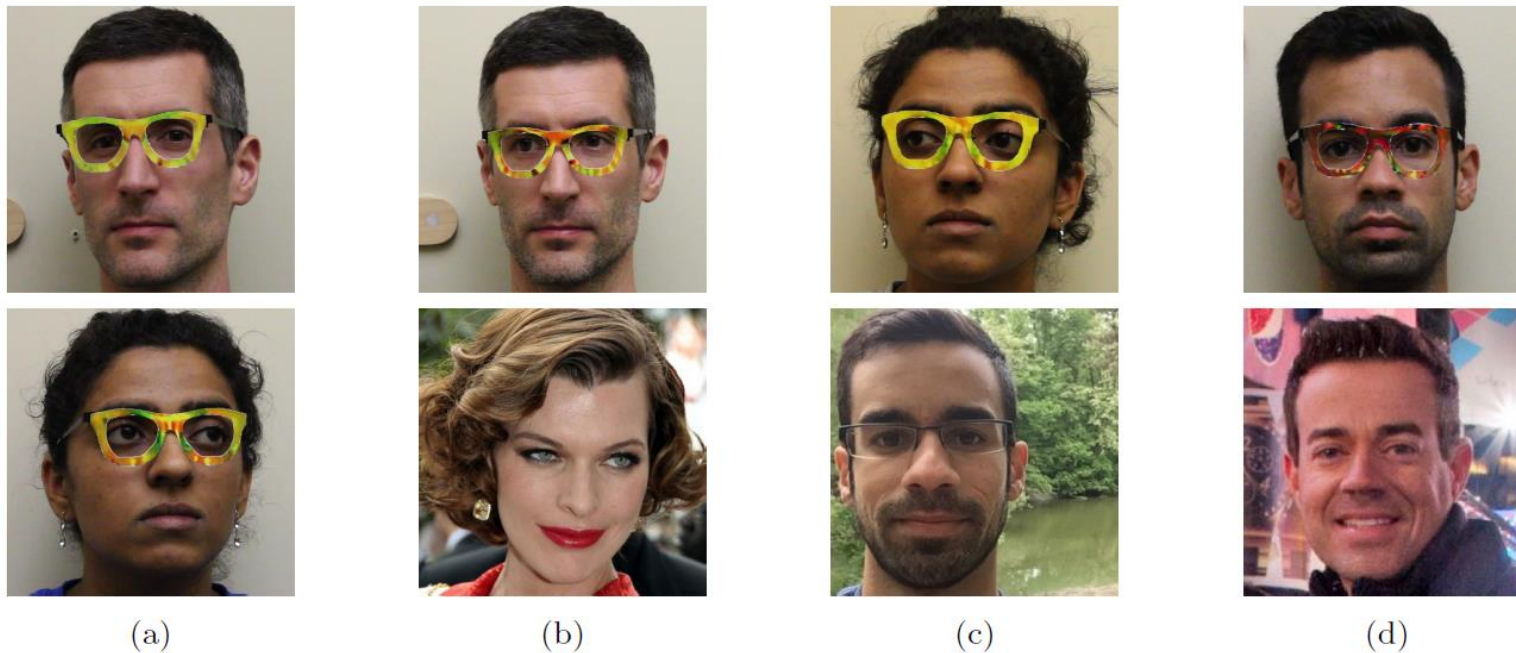


Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows  $S_A$  (top) and  $S_B$  (bottom) dodging against  $DNN_B$ . Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows  $S_A$  impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from <https://goo.gl/GlsWlC>); (c)  $S_B$  impersonating  $S_C$ ; and (d)  $S_C$  impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from <https://goo.gl/VfnDct>).

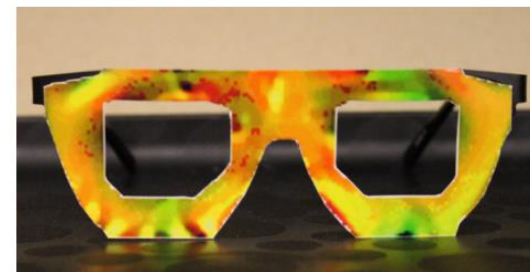


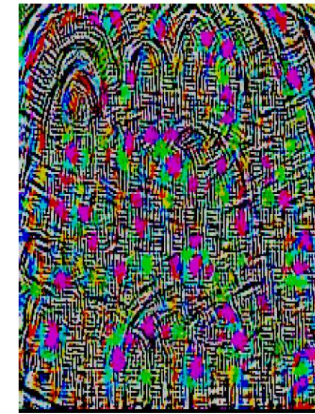
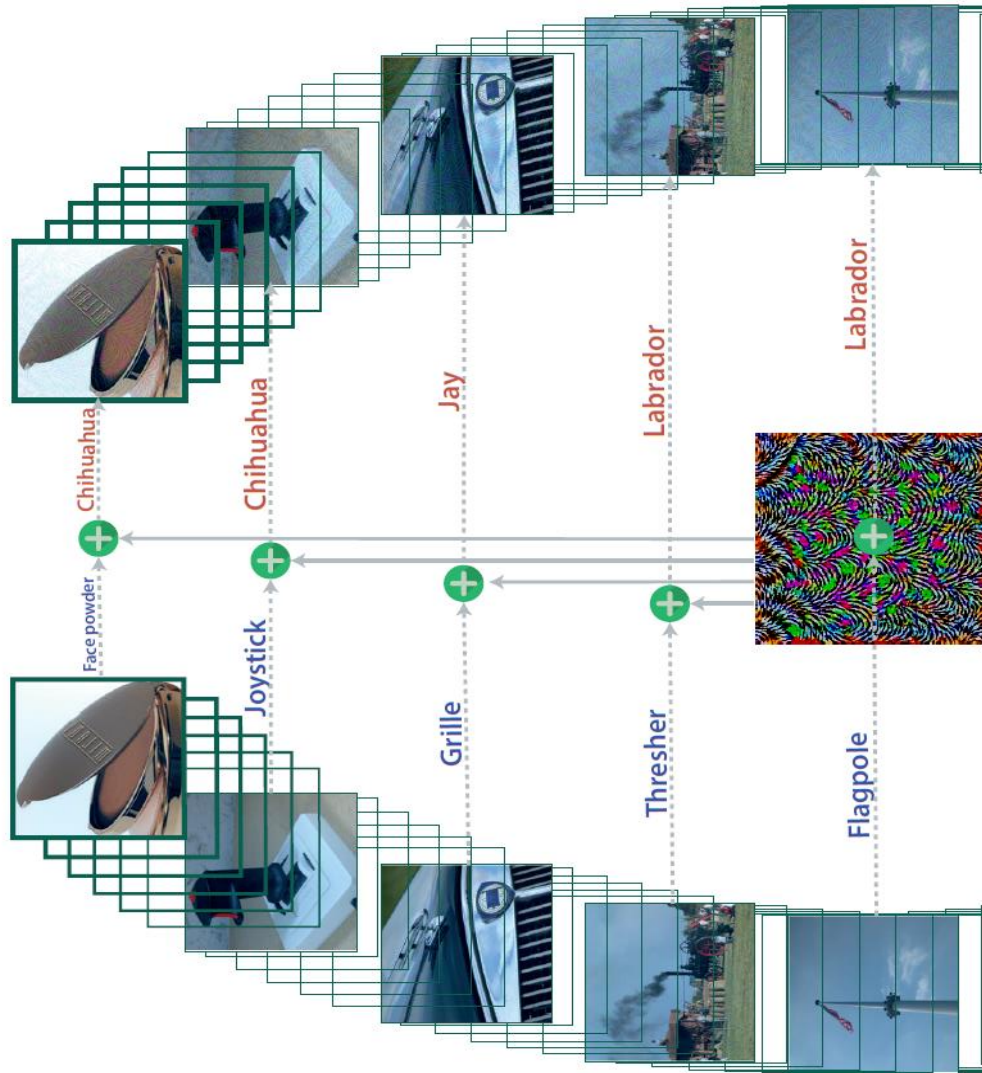
Figure 5: The eyeglass frames used by  $S_C$  for dodging recognition against  $DNN_B$ .



# Universal Perturbations

Mosavi-Dezfooli et al., CVPR 2017:

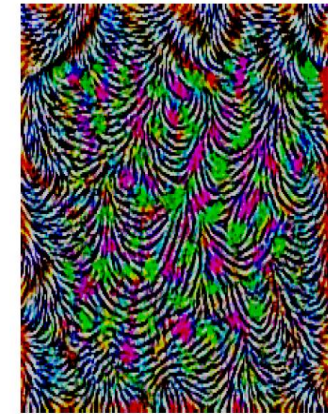
[https://openaccess.thecvf.com/content\\_cvpr\\_2017/papers/Mosavi-Dezfooli\\_Universal\\_Adversarial\\_Perturbations\\_CVPR\\_2017\\_paper.pdf](https://openaccess.thecvf.com/content_cvpr_2017/papers/Mosavi-Dezfooli_Universal_Adversarial_Perturbations_CVPR_2017_paper.pdf)



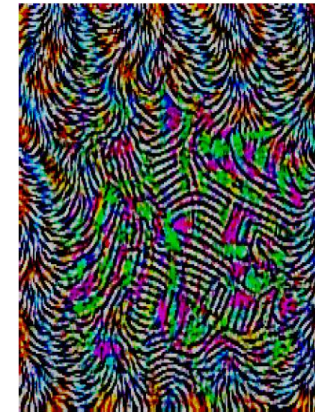
(a) CaffeNet



(b) VGG-F



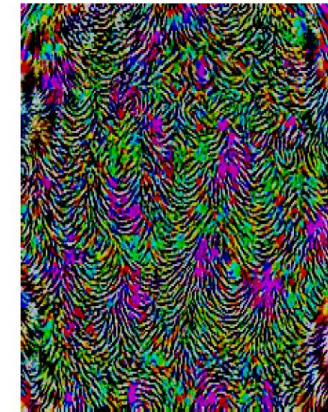
(c) VGG-16



(d) VGG-19



(e) GoogLeNet



(f) ResNet-152

Figure 4: Universal perturbations computed for different deep neural network architectures. Images generated with  $p = \infty$ ,  $\xi = 10$ . The pixel values are scaled for visibility.

# Computing Universal Perturbations

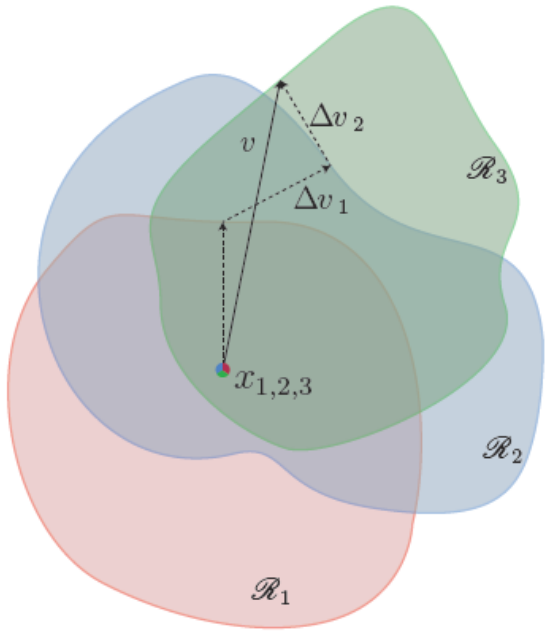


Figure 2: Schematic representation of the proposed algorithm used to compute universal perturbations. In this illustration, data points  $x_1, x_2$  and  $x_3$  are super-imposed, and the classification regions  $\mathcal{R}_i$  (i.e., regions of constant estimated label) are shown in different colors. Our algorithm proceeds by aggregating sequentially the minimal perturbations sending the current perturbed points  $x_i + v$  outside of the corresponding classification region  $\mathcal{R}_i$ .

---

## Algorithm 1 Computation of universal perturbations.

---

- 1: **input:** Data points  $X$ , classifier  $\hat{k}$ , desired  $\ell_p$  norm of the perturbation  $\xi$ , desired accuracy on perturbed samples  $\delta$ .
- 2: **output:** Universal perturbation vector  $v$ .
- 3: Initialize  $v \leftarrow 0$ .
- 4: **while**  $\text{Err}(X_v) \leq 1 - \delta$  **do**
- 5:     **for** each datapoint  $x_i \in X$  **do**
- 6:         **if**  $\hat{k}(x_i + v) = \hat{k}(x_i)$  **then**
- 7:             Compute the *minimal* perturbation that sends  $x_i + v$  to the decision boundary:

$$\Delta v_i \leftarrow \arg \min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

- 8:             Update the perturbation.

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

- 9:         **end if**
  - 10:     **end for**
  - 11: **end while**
- 

"Projects the perturbation back onto  $\ell_p$  ball if it exceeds the budget  $\xi$ "



# Transferability of Adversarial Perturbations

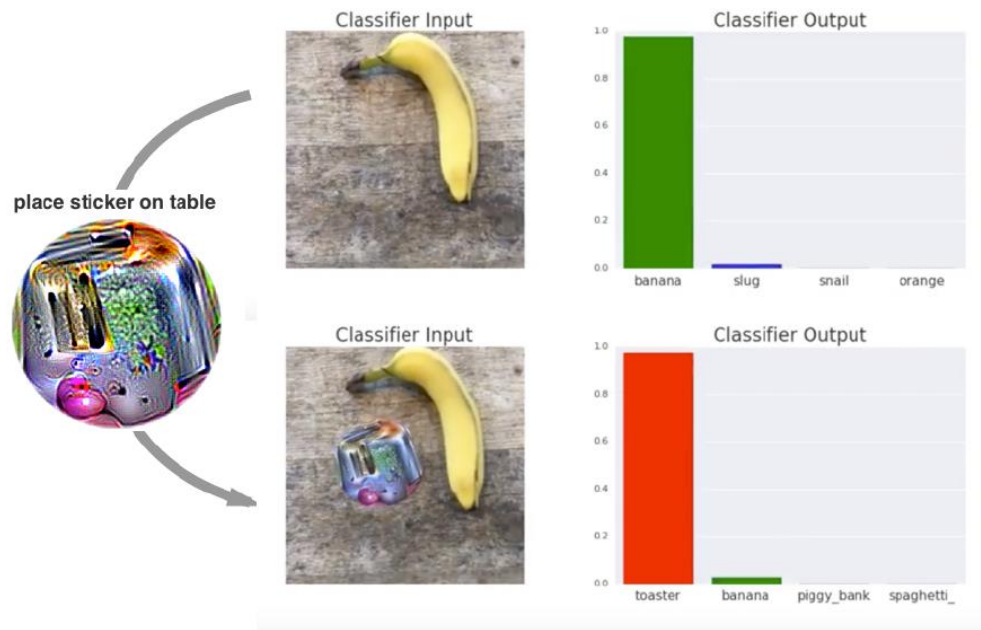
	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	<b>93.7%</b>	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	<b>93.3%</b>	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	<b>78.9%</b>	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	<b>78.3%</b>	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	<b>77.8%</b>	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	<b>84.0%</b>

Table 2: Generalizability of the universal perturbations across different networks. The percentages indicate the fooling rates. The rows indicate the architecture for which the universal perturbations is computed, and the columns indicate the architecture for which the fooling rate is reported.

Transferability of perturbations



# Adversarial Patches



[https://arxiv.org/pdf/1712.09665.pdf?source=post\\_page-----](https://arxiv.org/pdf/1712.09665.pdf?source=post_page-----)

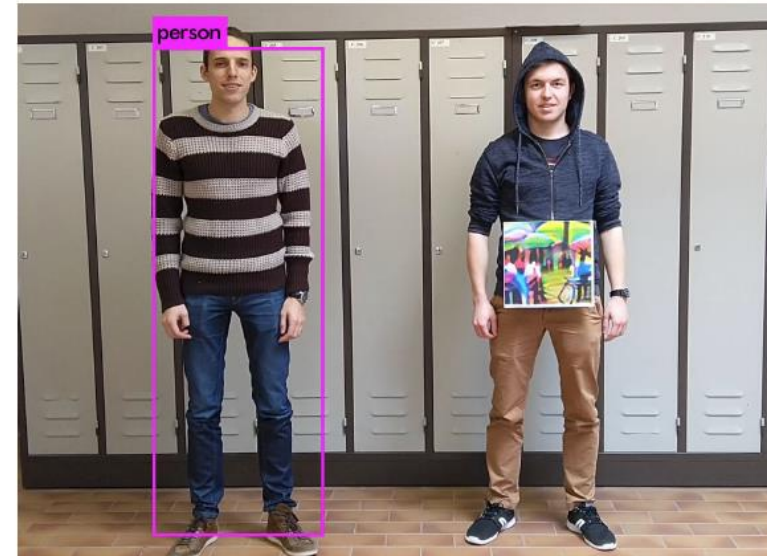


Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

[https://openaccess.thecvf.com/content\\_CVPRW\\_2019/papers/CV-COPS/Thys\\_Fooling\\_Automated\\_Surveillance\\_Cameras\\_Adversarial\\_Patches\\_to\\_Attack\\_Person\\_Detection\\_CVPRW\\_2019\\_paper.pdf](https://openaccess.thecvf.com/content_CVPRW_2019/papers/CV-COPS/Thys_Fooling_Automated_Surveillance_Cameras_Adversarial_Patches_to_Attack_Person_Detection_CVPRW_2019_paper.pdf)

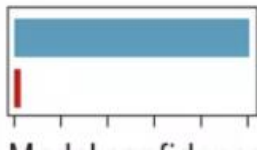
# Impact on Medical Diagnostics

<https://science.sciencemag.org/content/363/6433/1287>

**Original image**



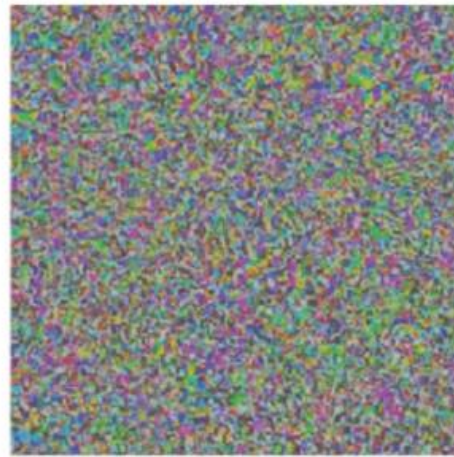
Dermoscopic image of a benign melanocytic nevus, along with the diagnostic probability computed by a deep neural network.



Benign  
Malignant

+ 0.04 ×

**Adversarial noise**



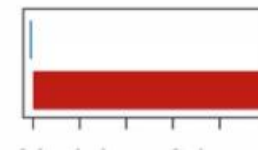
Perturbation computed by a common adversarial attack technique. See (7) for details.

=

**Adversarial example**



Combined image of nevus and attack perturbation and the diagnostic probabilities from the same deep neural network.

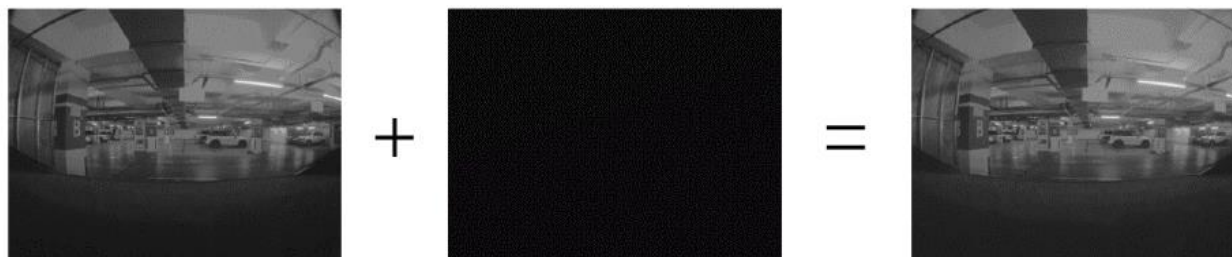


Benign  
Malignant

# Impact on Autonomous Driving

## It's disturbingly easy to trick AI into doing something deadly

How “adversarial attacks” can mess with self-driving cars, medicine, and the military.



Original image captured by 'fisheye' Rainy score 0.0113

Noise generated by PSO with every pixel smaller than 1000(int16)

Adversarial example Rainy score 0.8204



Fig 29. Left picture shows we add some noise on the left lane line in digital level, and right picture shows the result of APE's lane recognition function. (We redact top left of our image for privacy reasons, but it won't affect the final result.)



Fig 30. Left picture shows we add some patch around the left lane line in digital level, and right picture shows the result

[https://keenlab.tencent.com/en/whitepapers/Experimental\\_Security\\_Research\\_of\\_Tesla\\_Autopilot.pdf](https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf)

# Black Box Attacks

<https://arxiv.org/pdf/1804.08598.pdf>

- What if don't have access to the classifier and therefore cannot compute gradients?
- Option 1: train a surrogate model and leverage transferrability
- Option 2: **Estimate** the gradient numerically

$$\nabla \mathbb{E}[F(\theta)] \approx \frac{1}{\sigma n} \sum_{i=1}^n \delta_i F(\theta + \sigma \delta_i)$$

Instead of estimating the gradient at a single point, we instead t and estimate the gradient the average value of function F in its neighborhood, where neighborhood points are drawn from a Gaussian distribution of std dev. signma!

---

**Algorithm 1** NES Gradient Estimate

---

**Input:** Classifier  $P(y|x)$  for class  $y$ , image  $x$

**Output:** Estimate of  $\nabla P(y|x)$

**Parameters:** Search variance  $\sigma$ , number of samples  $n$ , image dimensionality  $N$

$g \leftarrow \mathbf{0}_n$

**for**  $i = 1$  **to**  $n$  **do**

$u_i \leftarrow \mathcal{N}(\mathbf{0}_N, \mathbf{I}_{N \cdot N})$

$g \leftarrow g + P(y|x + \sigma \cdot u_i) \cdot u_i$

$g \leftarrow g - P(y|x - \sigma \cdot u_i) \cdot u_i$

**end for**

**return**  $\frac{1}{2n\sigma} g$

---



# Black Box Attack Results

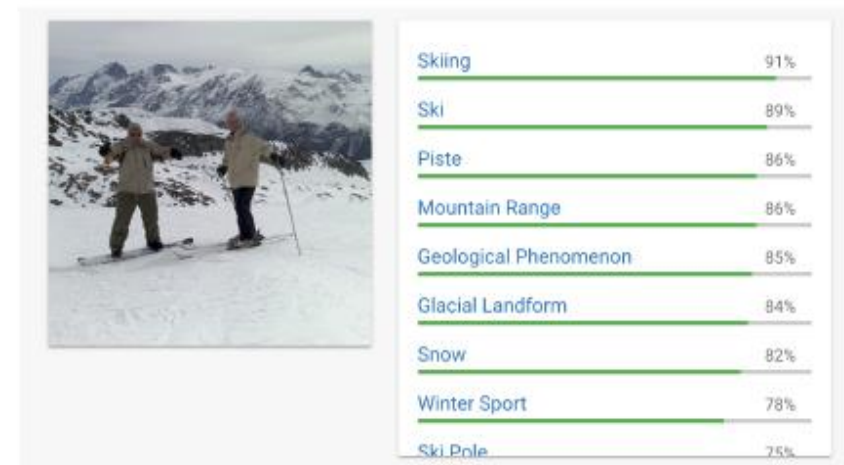
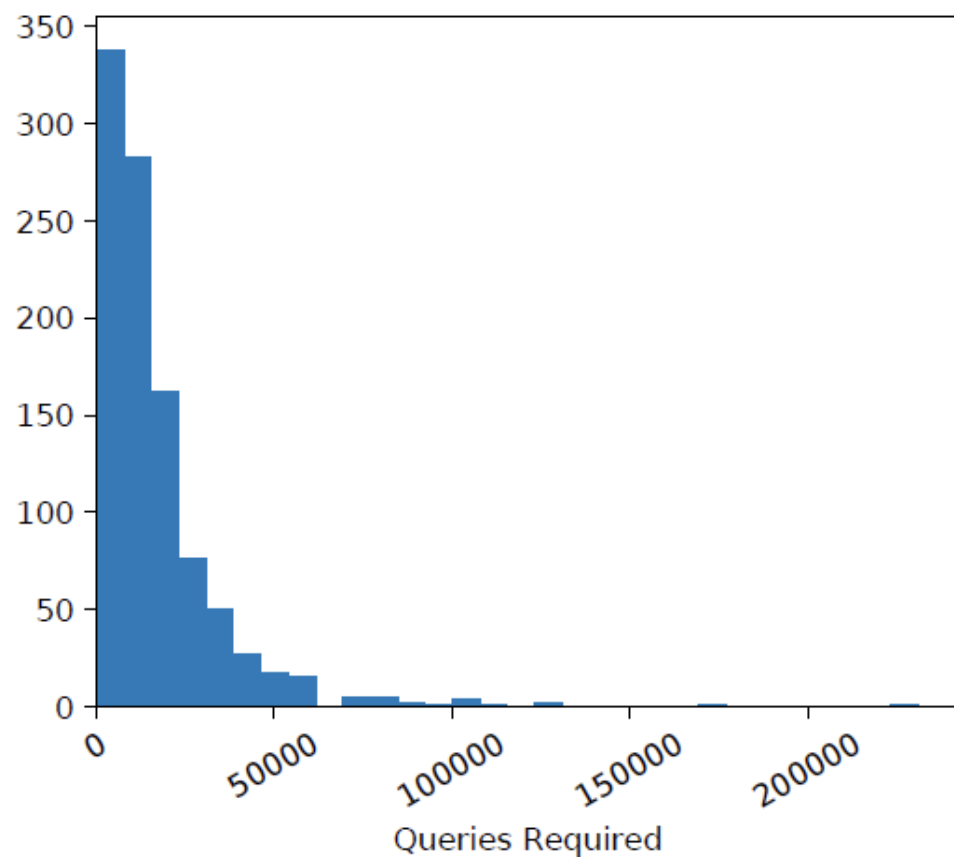


Figure 4. The Google Cloud Vision Demo labeling on the unperturbed image.

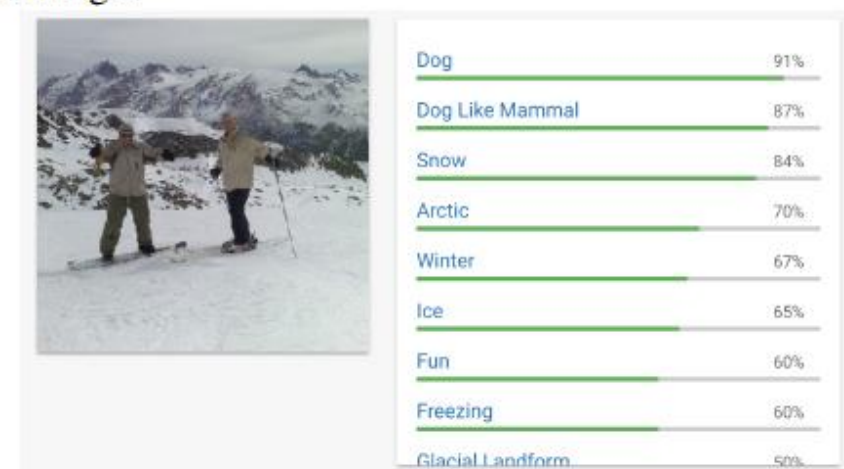


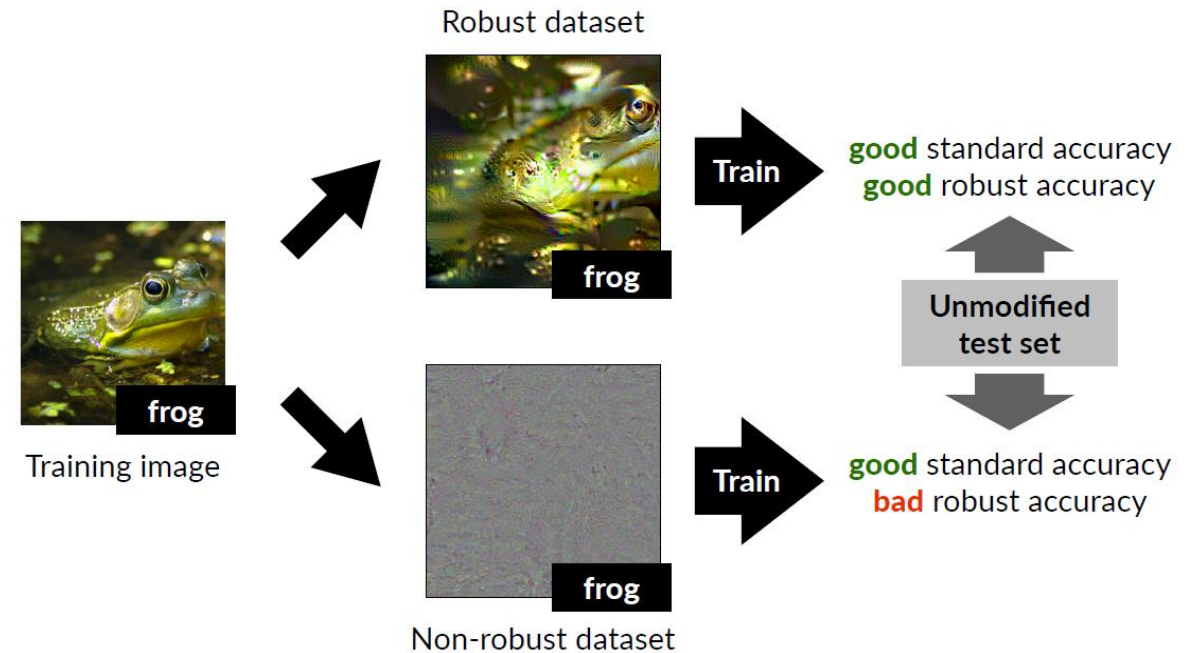
Figure 5. The Google Cloud Vision Demo labeling on the adversarial image generated with  $\ell_\infty$  bounded perturbation with  $\epsilon = 0.1$ : the image is labeled as the target class.

# Understanding Adversarial Examples

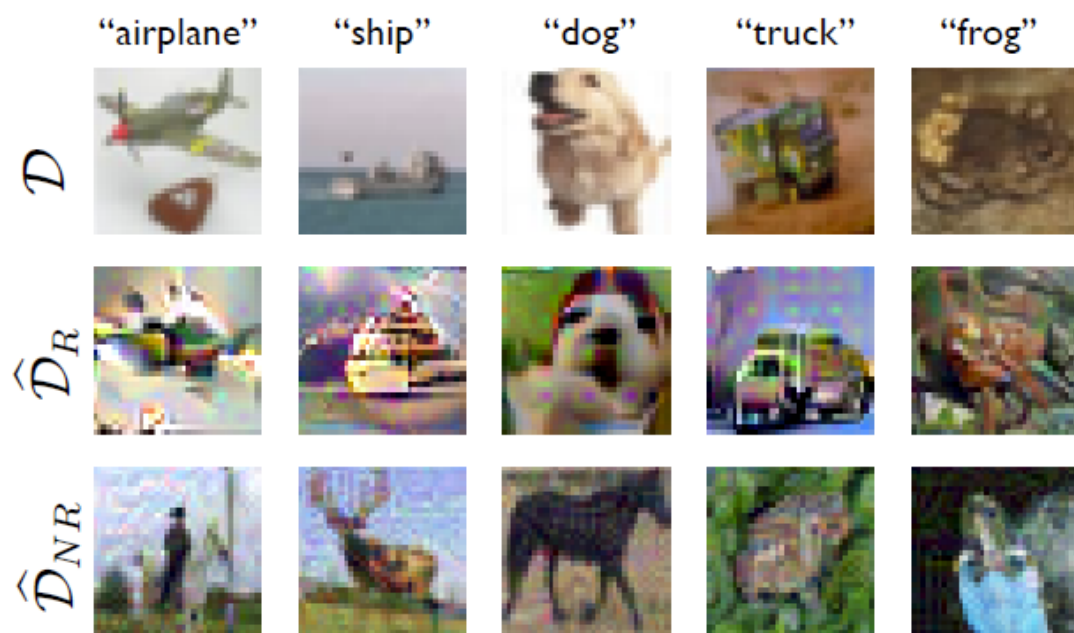
<https://papers.nips.cc/paper/8307-adversarial-examples-are-not-bugs-they-are-features.pdf>

## "Features not bugs"

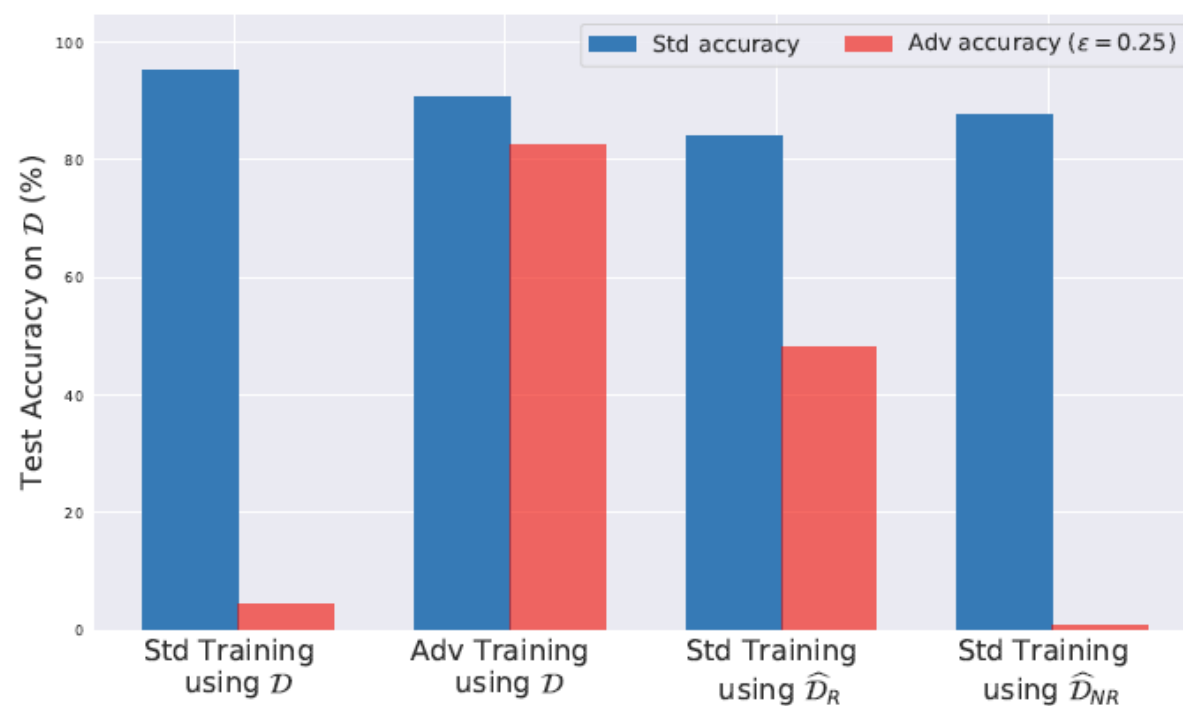
- Deep networks learn both "robust" features (usually ones that align with what humans "see") and "brittle" features (usually imperceptible) to maximize accuracy
- Adversarial perturbations are effectively a consequence of changes to non-robust features



# Experimental Validation



(a)



(b)

# Defenses Against Adversarial Perturbations

- A "first-wave" of defenses against adversarial perturbations all failed. For example, Carlini and Wagner were able to circumvent ten SOTA adversarial perturbation detection schemes

[https://dl.acm.org/doi/pdf/10.1145/3128572.3140444?casa\\_token=RODR3aX2XMEAAAAA:S65LnWnAqHZSjn1NuAe5gF9XplbL0qVFbBvnXcy&slctp0MHOA8O8Pmq8PbqpdIkjE0DQJrIPVO](https://dl.acm.org/doi/pdf/10.1145/3128572.3140444?casa_token=RODR3aX2XMEAAAAA:S65LnWnAqHZSjn1NuAe5gF9XplbL0qVFbBvnXcy&slctp0MHOA8O8Pmq8PbqpdIkjE0DQJrIPVO)

As done in Biggio *et al.* [5], we consider three different threat models in this paper:

- (1) A *Zero-Knowledge Adversary* generates adversarial examples on the unsecured model  $F$  and is not aware that the detector  $D$  is in place. The detector is successful if it can detect these adversarial examples.
- (2) A *Perfect-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme  $D$ , knows the model parameters used by  $D$ , and can use these to attempt to evade both the original network  $F$  and the detector simultaneously.
- (3) A *Limited-Knowledge Adversary* is aware the neural network is being secured with a given detection scheme, knows how it was trained, but does not have access to the trained detector  $D$  (or the exact training data).



One of the key insights was that the SOTA defenses at the time were not considering "adaptive" adversaries that had knowledge of the defense!



# Examples of Defenses Circumvented

- Adversarial example detector D trained on adversarial examples generated on DNN  $F_{\text{base}}$   $\rightarrow F_{\text{secured}}$

## 3.1 Adversarial Retraining

Grosse *et al.* [15] propose a variant on adversarial re-training. Instead of attempting to classify the adversarial examples correctly (by adding adversarial examples to the training set, with their correct labels), they introduce a new  $N + 1$ st class — solely for adversarial examples — and train the network to detect adversarial examples. Specifically, they propose the following procedure:

For Grosse’s defense, we use C&W’s attack on  $F_{\text{secured}}$  to generate adversarial examples; it succeeds 100% of the time. We computed the mean  $L_2$ -distance from the original sample to the adversarial example. Adversarial examples against  $F_{\text{base}}$  are at average  $L_2$  distance of 2.05 from the original sample; adversarial examples against  $F_{\text{secured}}$  have an average distance of 2.26. Thus the defense has not reduced the success rate at generating adversarial examples, and has only increased the mean distortion by 10%.

# Examining Convolution Layers

- These defenses work on the intuition that the convolutional layer features of adversarial examples are "anomalous" with respect to clean examples

In contrast to the prior approach, which attempts to detect adversarial examples based on the contents of the image itself, Metzen *et al.* [18] detect adversarial examples by looking at the inner convolutional layers of the network. They augment the classification neural network with a detection neural network that takes its input from various intermediate layers of the classification network. This detection network is trained identically to Gong's defense above. We refer interested readers to the original paper for complete details on the detector setup [18]. This defense only argues robustness against CIFAR, since it looks at the inner layers of a ResNet (which are not usually used for MNIST).

*Perfect-Knowledge Attack Evaluation.* Our white-box attack completely defeats Metzen's defense: it is able to produce adversarial examples that simultaneously are mis-classified by the original network and evade the detector. We generate adversarial examples using C&W's attack applied to the same function  $G(\cdot)$  defined in Section 3.1. The mean distance to adversarial examples increases from 0.169  $L_2$  distortion on the unsecured model to 0.227 on the secured scheme, an improvement of 34%. However, in absolute terms, the adversarial examples generated are still indistinguishable from the original inputs.

# Principal Component Analysis

- Defense intuition: adversarial examples impact the lower order principal components of the input -> **perform classification only on the higher order PCA components**

*Perfect-Knowledge Attack Evaluation.* We found that the Hendrycks defense can be broken by a white-box attacker with knowledge of the defense. Details are deferred to Section 4.2, where we break a strictly stronger defense. In particular, we found in our experiments that we can generate adversarial examples that are restricted to change only the first  $k$  principal components (i.e., leave all later components unchanged), and these adversarial examples that are not detected by the Hendrycks defense.

# Defensive Dropout

- Intuition: predictions on noisy clean inputs are more stable than predictions on noisy adversarial inputs!

Feinman *et al.* propose a second detection method called *Bayesian neural network uncertainty* that measures the uncertainty of the neural network on the given input. Instead of relying on the reported confidence of the network (which can easily be controlled by an adversary), they add randomization to the network. The hope is that a natural image will have the same (correct) label regardless of the random values chosen, while adversarial examples won't always be predicted with the same label. *Dropout* [37] is used as the method of adding randomness.

Straightforward  
"adaptive" attack  
fails! So is this defense  
secure?

# Updated Loss

We sample  $K$  different deterministic networks  $\{Z_j(\cdot) : j \in [1, K]\}$  each with different randomness used during dropout. If we were able to have  $\arg \max_i Z_j(x)_i = t$  for every network  $j$ , for  $K$  big enough, it would be highly likely that  $F_r(x)$  would always produce label  $t$  for any randomness. Thus, we construct a new loss function  $\ell'(x') = \sum_{j=1}^K \ell_{Z_j}(x')$  as the average of the loss functions on each fixed model  $Z_j$ . Then we use C&W's attack with this revised loss function.

"Expectations over transformations" attack against randomized defenses

# So What Works?

- Madry et al. (<https://arxiv.org/pdf/1706.06083>) propose to defend against adversarial perturbation from first principles
- What does the defender seek to achieve? Can we define a loss function that incorporates the adversaries objectives? (recall game between attacker and defender)

"minimax  
optimization"

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \max_{\delta \in \mathcal{S}} L(\theta, x + \delta, y) \right] .$$

# Evaluations Against Advanced Defenses

- [https://nicholas.carlini.com/papers/2020\\_neurips\\_adaptiveattacks.pdf](https://nicholas.carlini.com/papers/2020_neurips_adaptiveattacks.pdf)

# Adversarial Re-training

---

**Algorithm 3** PGD SGD

---

```
1: Input:  $[X_{B_1}, X_{B_2} \dots X_{B_J}], f, \eta, \eta', w = w^0, \epsilon$ 
2: for  $t = 0, \dots, T - 1$  do
3:   for  $j = 1, \dots, J$  do
4:      $x'^0 \leftarrow x \{ \forall x \in X_{B_j} \}$ 
5:     for  $k = 0, \dots, K - 1$  do
6:        $dx'^k \leftarrow \frac{1}{n} \sum_{i=1}^n \nabla_{x=x^k} L(f(w^t; x))$ 
7:        $x'^{k+1} \leftarrow x'^k + \eta' dx'^k$  {Gradient ascent}
8:        $\text{project}_{x+\Delta}(x', \epsilon)$ 
9:     end for
10:     $\mu^t \leftarrow L(w^t, x'^K)$  {batch loss for  $X_{B_j}$ }
11:     $dL^t \leftarrow \nabla_w \mu^t$  {gradient of batch loss}
12:     $w^{t+1} \leftarrow w^t - \eta dL^t$ 
13:  end for
14: end for
15: Output  $\hat{w} \leftarrow w^T$ 
```

In the inner loop, we perturb inputs in adversarial directions

In the outer loop we update the weights at these adversarially perturbed points

---



# "Leaderboard"

<https://robustbench.github.io/>

Rank	Method	Standard accuracy	Robust accuracy	Extra data	Architecture	Venue
1	<b>Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples</b> <i>We show the robust accuracy reported in the paper since AutoAttack performs slightly worse (65.88%).</i>	91.10%	65.87%	<input checked="" type="checkbox"/>	WideResNet-70-16	arXiv, Oct 2020
2	<b>Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples</b> <i>We show the robust accuracy reported in the paper since AutoAttack performs slightly worse (62.80%).</i>	89.48%	62.76%	<input checked="" type="checkbox"/>	WideResNet-28-10	arXiv, Oct 2020
3	<b>Adversarial Weight Perturbation Helps Robust Generalization</b>	88.25%	60.04%	<input checked="" type="checkbox"/>	WideResNet-28-10	NeurIPS 2020
4	<b>Does Network Width Really Help Adversarial Robustness?</b>	85.60%	59.78%	<input checked="" type="checkbox"/>	WideResNet-34-15	arXiv, Oct 2020
5	<b>Unlabeled Data Improves Adversarial Robustness</b>	89.69%	59.53%	<input checked="" type="checkbox"/>	WideResNet-28-10	NeurIPS 2019
6	<b>HYDRA: Pruning Adversarially Robust Neural Networks</b>	88.98%	57.14%	<input checked="" type="checkbox"/>	WideResNet-28-10	NeurIPS 2020
	<b>Uncovering the Limits of Adversarial Training against Norm-Bounded</b>					