CS 499/579: Trustworthy ML

Preliminaries on adversarial examples

Tu/Th 4:00 – 5:50 pm

Instructor: Sanghyun Hong

sanghyun.hong@oregonstate.edu

SAIL
Secure AI Systems Lab

Notes

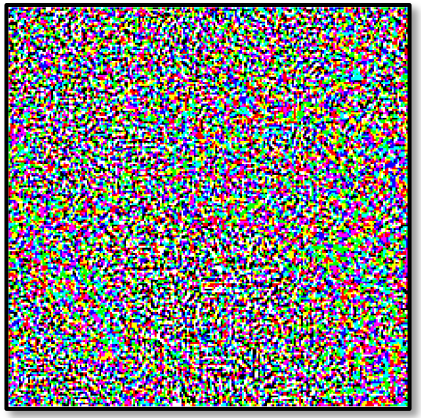
- Call for actions
  - In-class presentation sign-ups
  - Term project team-up

Adversarial examples

- A test-time input to a neural network
  - Crafted with the objective of fooling the network's decision(s)

Not every adversarial examples are interesting

- A test-time input to a neural network
  - Crafted with the objective of fooling the network's decision(s)
  - That looks like a natural test-time input

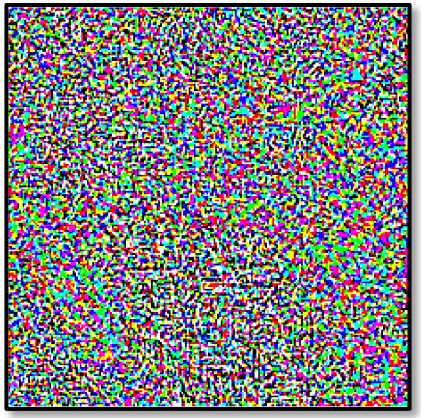


Noisy test-time input

NOT EVERY ADVERSARIAL EXAMPLES ARE INTERESTING

- A test-time input to a neural network
  - Crafted with the objective of fooling the network's decision(s)
  - That looks like a natural test-time input

$+ 0.007 \times$



$=$

Prediction: **Panda**

Human-imperceptible Noise

Prediction: **Gibbon**

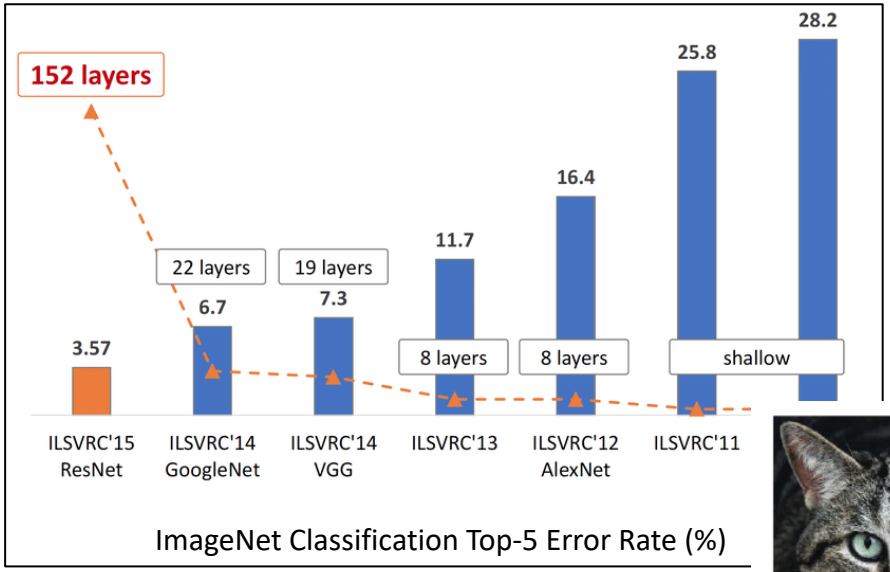
Goodfellow et al., *Explaining and Harnessing Adversarial Examples, International Conference on Learning Representations (ICLR)*, 2015.

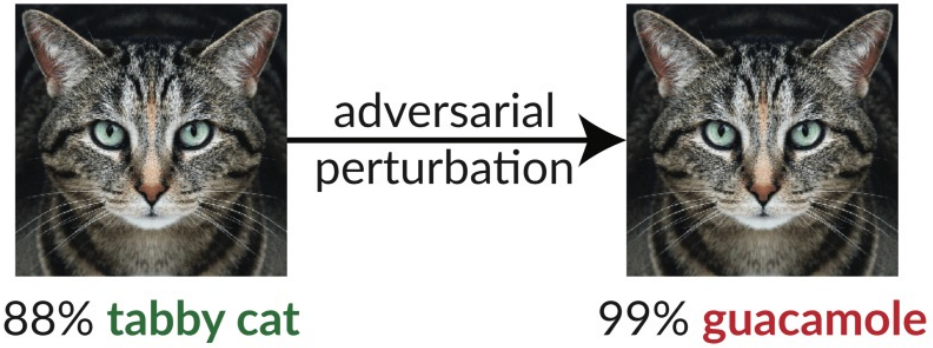
WHY DO THEY MATTER?

- from the security perspective: it makes ML-enabled systems unavailable

WHY DO THEY MATTER?

- from the ML perspective: it is counter-intuitive





Topics for part I – adversarial examples

- Research questions
  - What are the adversarial examples?
  - How can we find adversarial examples?
  - How can we exploit them in practice?
  - How can we defeat adversarial examples?

What are the adversarial examples?

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

What did we believe at that time?

- Two common beliefs about neural networks
  - Neurons represent certain features
    - People use this intuition to find *semantically-similar* inputs
    - Neural networks may have the ability to *disentangle* features at neuron-level
  - Neural Networks are stable when there is small perturbations to their inputs
    - *Random perturbations* to inputs are difficult to change networks' predictions

What did we believe at that time?

- Neurons represent certain features
- Re-visit this hypothesis[1]:
  - Find a set of inputs that maximally increases
    - The activation of i-th hidden neuron
    - The activation of random vector
  - Compare those two sets of inputs
  - More formally:

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), e_i \rangle$$

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), v \rangle$$

[1] Szegedy et al., Intriguing Properties of Neural Networks, ICLR

What did we believe at that time?

(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

(c) Unit senstive to round, spiky flowers.

(d) Unit senstive to round green or yellow objects.

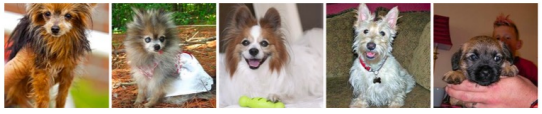
Images that activates a certain neuron the most

Images that activates a random dir. the most

(a) Direction sensitive to white, spread flowers.

(b) Direction sensitive to white dogs.

(c) Direction sensitive to spread shapes.



(d) Direction sensitive to dogs with brown heads.

What did we believe at that time?

- Neural networks are resilient to *small* input perturbations
- Re-visit this hypothesis[1]:
  - Let's find a small perturbation that changes a model's classification result
  - Initial work formulates this problem like:

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

  - Formally:

> - Minimize $c|r| + \text{loss}_f(x+r, l)$ subject to $x + r \in [0,1]^m$

[1] Szegedy et al., Intriguing Properties of Neural Networks, ICLR

How to solve this constrained optimization?

- Intuitions
  - Non-linearity, from activation functions like ReLU, is the root-cause
  - Downside:
    - Computationally demanding, if we find adversarial examples in non-linear models
    - It's also not theoretically proven that non-linearity is the primary issue
  - This work:
    - let's only consider linearity in non-linear models!
    - I will show the existence of adversarial examples exploiting the linearity

How to solve this constrained optimization?

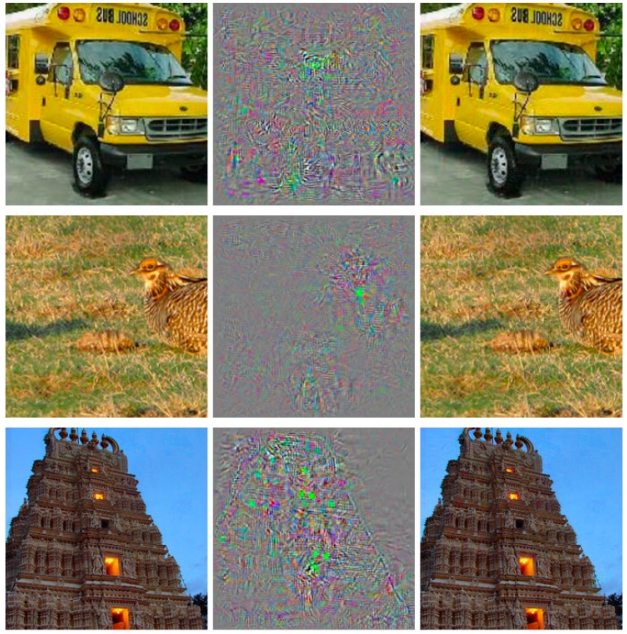
- Fast gradient sign method (FGSM)
  - A test-time input x and its true label y
  - A NN model f and its parameters θ
  - A loss (or a cost) function $J(\theta, x, y)$
  - Find an adversarial perturbation η such that $f(x + \eta) \neq y$ and $||\eta||_\infty < \varepsilon$

$$\boldsymbol{\eta} = \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right).$$

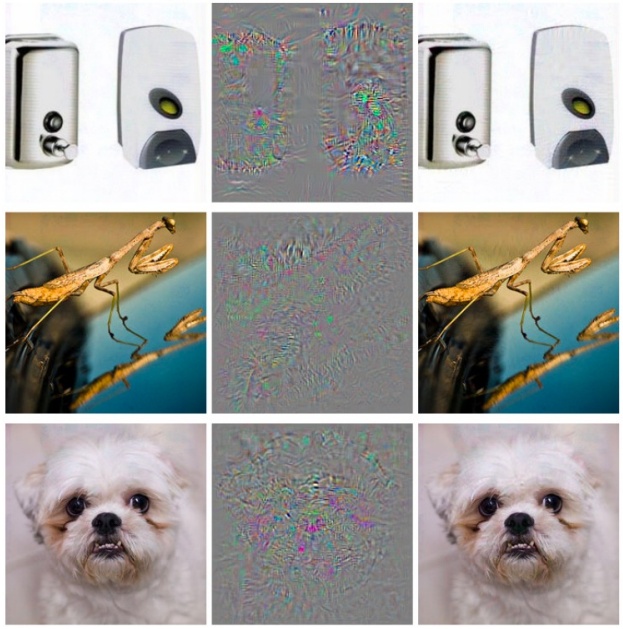
- Results on the test-sets
  - On MNIST: 99.9% error rate with an avg. confidence of 79.3% (eps = 0.25)
  - On CIFAR10: 87.2% error rate with an avg. confidence of 96.6% (eps = 0.1)

Results

- Attacking AlexNet models trained on ImageNet



(a) Incorrect



(b) Incorrect

Results

- Empirical findings:

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

  - Random perturbations are **NOT** the right way to measure the stability of neural networks
  - Adversarial examples **transfer**
    - Adversarial examples crafted on a model often work against others
    - AEs crafted on a model (trained with a disjoint training set) also works against the others

How can we find adversarial examples?

- Sub research questions
  - How can we define the adversarial examples?
  - What are the methods we can develop for finding adversarial examples?
  - What are the computational properties adversarial examples exploit?

What are the adversarial examples (precisely)?

- A test-time input x^* to a neural network
  - Crafted with the objective of fooling the network's decision(s)
  - That looks like a natural test-time input

$$\mathbf{x}^* = \arg\max_{\mathrm{x}} \quad \hat{g}(\mathbf{x}\,, \mathrm{y})$$

$$\text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}.$$

  - Formulation
    - x: an adversarial example
    - x^0: a clean test-time input
    - x^*: an optimal adversarial examples
    - $g(x, y)$: error (loss) computed on a test-time sample with respect to the true label y
    - $d(x, x^0)$: pixel-wise distance between x and x^0

What are the adversarial examples (precisely)?

- A test-time input x^* to a neural network

$$\mathbf{x}^* = \arg\max_{\mathrm{x}} \quad \hat{g}(\mathbf{x}\ , \mathrm{y})$$

$$\text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}.$$

  - Formulation
    - x: an adversarial example
    - x^0: a clean test-time input
    - x^*: an optimal adversarial examples
    - $g(x, y)$: error (loss) computed on a test-time sample with respect to the true label y
    - $d(x, x^0)$: pixel-wise distance between x and x^0

  - (In the context of supervised learning) Goals:
    - Untargeted misclassification: x to any class label other than y
    - Targeted misclassification: x to a specific class label y_t

What are the methods for finding adversarial examples?

- A test-time input x^* to a neural network

$$\mathbf{x}^* = \arg\max_{\mathrm{x}} \quad \hat{g}(\mathbf{x}\ , \mathrm{y})$$

$$\text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}.$$

- Potential approaches
  - Man-ual:
    - Add Gaussian noise (or any type of noise) to the input x
    - Manipulate pixels of x that are likely to lead to fool the neural network
  - (or easily) Gradient-based approach:
    - Compute gradients to the input
    - In a way that the gradients increase the loss g with respect to y
    - You can do this easily with PyTorch, Tensorflow, Objax…

Gradient-based method

- Method formulated by Biggio *et al.*[1]

Algorithm 1 Gradient-descent evasion attack

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

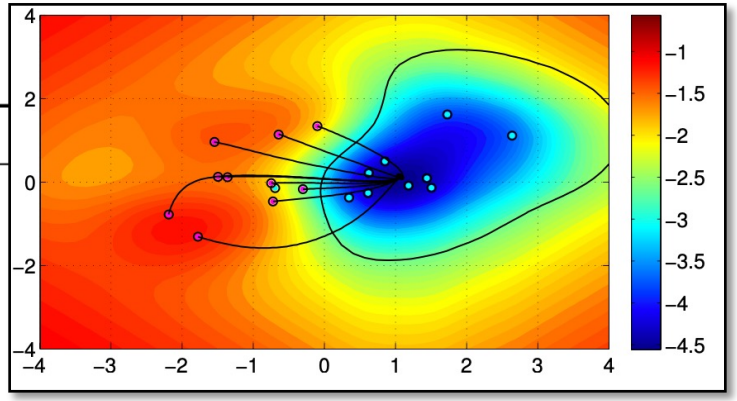
Output: $\mathbf{x}^*$, the final attack point.

$m \leftarrow 0$.
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: $\mathbf{x}^* = \mathbf{x}^m$

[1] Biggio et al., Evasion Attacks against Machine Learning Models at Test Time

Gradient-based method

- Method formulated by Biggio *et al.*[1]



Algorithm 1 Gradient-descent evasion attack

Input: $\mathbf{x}^0$, the initial attack point; t, the step size; small constant.

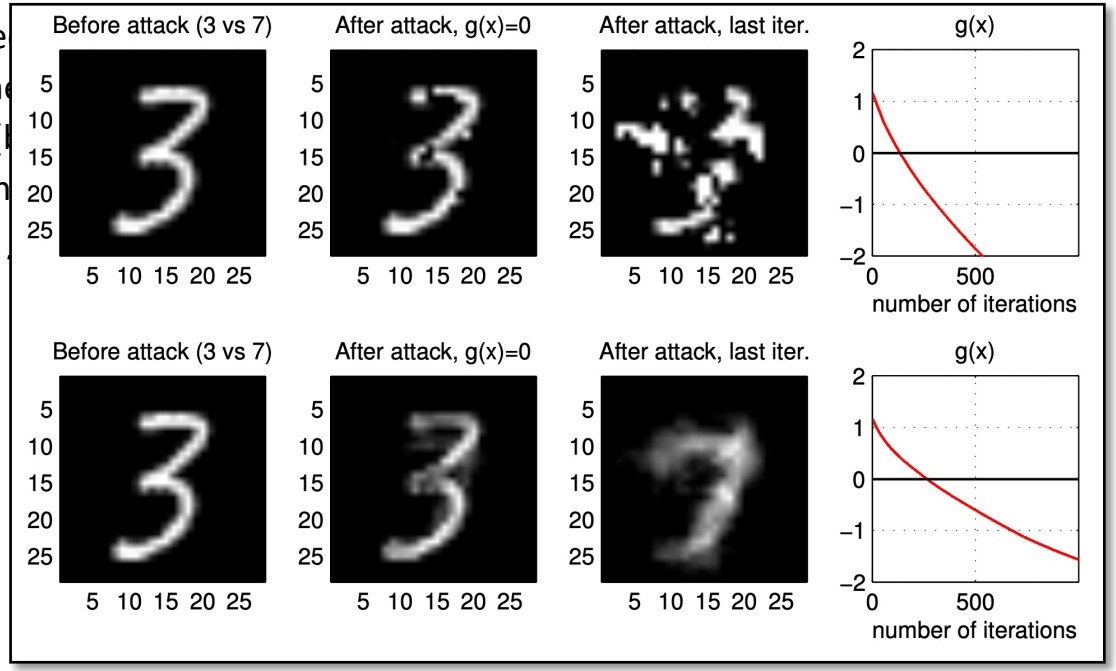
Output: $\mathbf{x}^*$, the final attack point.

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

[1] Biggio et al., Evasion Attacks against Machine Learning Models at Test Time

Gradient-based method

- Method formulated by Biggio *et al.*
  - Start from a clean test-time sample x
  - Iteratively do the followings:
    - Compute the loss with respe
    - Compute the gradients to th
    - Perturb the test-time input (
    - Bound the perturbation with
  - Return the adv. example of



Does it lead to finding the strongest adv. exampels?

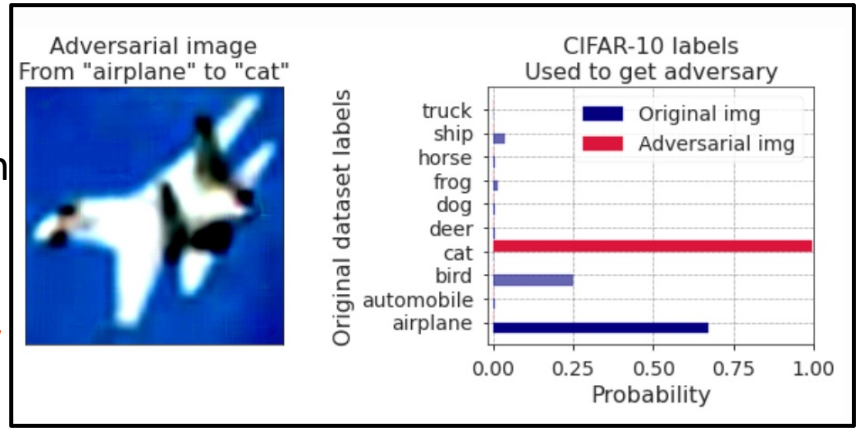
Towards evaluating the robustness of neural networks, Carlini and Wagner, IEEE S&P 2017

Revisiting the formulation

- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y); each element in $x \sim [0, 1]$
    - A NN model f and its parameters θ
  - Objective
    - Find an x^{adv} such that $f\left(x^{adv}\right) \neq y$ while $||x^{adv} - x||_p \leq \varepsilon$

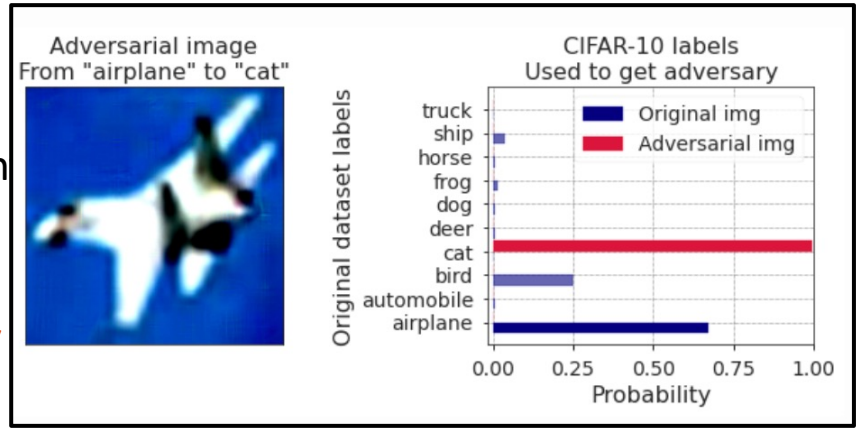
Revisiting the formulation: what are the possibilities?

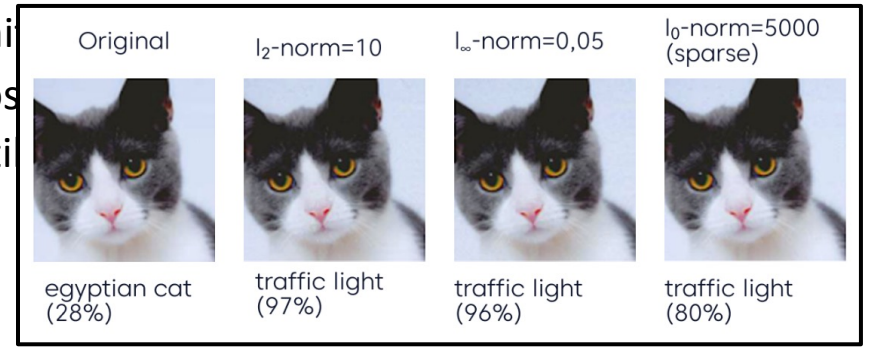
- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y); each elemen
    - A NN model f and its parameters θ
  - Objective
    - Find an x^{adv} such that $f(x^{adv}) = \boldsymbol{y'}$
  - Possible misclassification ($\boldsymbol{y'}$)
    - Best-case: to the class the least difficult to attack
    - Average-case: to the class chosen uniformly at random
    - Worst-case: to the class that was most difficult to attack



Revisiting the formulation: what are we missing?

- Test-time (evasion) attack
  - Suppose
    - A test-time input (x, y); each elemen
    - A NN model f and its parameters θ
  - Objective
    - Find an x^{adv} such that $f(x^{adv}) = \boldsymbol{y'}$
  - Possible misclassification ($\boldsymbol{y'}$)
    - Best-case: to the class the least difficult to attack
    - Average-case: to the class chosen uni
    - Worst-case: to the class that was mos
  - Ways to quantify the "human-imperceptibl
    - $p = 0, 1, 2, \ldots \infty$ $(L_0, L_1, L_2, L_\infty)$





Finding adversarial examples: re-write the problem!

- Problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

  - x, δ are a test-time sample and perturbations
  - D is the distance between the original and adv. examples
  - C and t are the target classifier and class

- Solution approach:
  - Formulate it as an optimization problem
  - Find a set of fs (algorithms) that can solve the optimization

Finding adversarial examples: re-write the problem!

- Problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

  - x, δ are a test-time sample and perturbations
  - D is the distance between the original and adv. examples
  - C and t are the target classifier and class

- Solution approach:
  - Formulate it as an optimization problem
  - Find a set of fs (algorithms) that can solve
  - Possible choices of f

$$\begin{aligned} f_1(x') &= -\text{loss}_{F,t}(x') + 1 \\ f_2(x') &= (\max_{i \neq t}(F(x')_i) - F(x')_t)^+ \\ f_3(x') &= \text{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2) \\ f_4(x') &= (0.5 - F(x')_t)^+ \\ f_5(x') &= -\log(2F(x')_t - 2) \\ f_6(x') &= (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+ \\ f_7(x') &= \text{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2) \end{aligned}$$

Finding adversarial examples: re-write the problem!

- Problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

  - x, δ are a test-time sample and perturbations
  - D is the distance between the original and adv. examples
  - C and t are the target classifier and class

- Solution approach:
  - Formulate it as an optimization problem
  - Find a set of fs (algorithms) that can solve the optimization
  - Possible choices of f
  - Possible choices of solvers: PGD, Clipped GD, Change of variables

FINDING ADVERSARIAL EXAMPLES: INITIAL FINDINGS

- Choose the objective:

	Best Case						Average Case						Worst Case					
	Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
f_1	2.46	100%	2.93	100%	2.31	100%	4.35	100%	5.21	100%	4.11	100%	7.76	100%	9.48	100%	7.37	100%
f_2	4.55	80%	3.97	83%	3.49	83%	3.22	44%	8.99	63%	15.06	74%	2.93	18%	10.22	40%	18.90	53%
f_3	4.54	77%	4.07	81%	3.76	82%	3.47	44%	9.55	63%	15.84	74%	3.09	17%	11.91	41%	24.01	59%
f_4	5.01	86%	6.52	100%	7.53	100%	4.03	55%	7.49	71%	7.60	71%	3.55	24%	4.25	35%	4.10	35%
f_5	1.97	100%	2.20	100%	1.94	100%	3.58	100%	4.20	100%	3.47	100%	6.42	100%	7.86	100%	6.12	100%
f_6	1.94	100%	2.18	100%	1.95	100%	3.47	100%	4.11	100%	3.41	100%	6.03	100%	7.50	100%	5.89	100%
f_7	1.96	100%	2.21	100%	1.94	100%	3.53	100%	4.14	100%	3.43	100%	6.20	100%	7.57	100%	5.94	100%

  - MNIST; Test all f_1- f_7 the objectives; Measure L_2 distances
  - f_2- f_4 do not lead to the successful adversarial attacks
  - f_1 requires large c value
  - Choose one over f_5- f_7

Finding adversarial examples: putting all together

- Problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned}$$

- Solution approach:
  - Solver: Change of variables
  - Objective function: f_6

Change of variables introduces a new variable w and instead of optimizing over the variable δ defined above, we apply a change-of-variables and optimize over w, setting

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i.$$

Since $-1 \leq \tanh(w_i) \leq 1$, it follows that $0 \leq x_i + \delta_i \leq 1$, so the solution will automatically be valid. [8]

- Carlini and Wagner (C&W) Attack:

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(w)+1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w)+1)$$

with f defined as

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa).$$

Finding strong adversarial examples: empirical eval.

- Empirical evaluation
  - D: MNIST, CIFAR-10, and ImageNet
  - x: randomly chosen 1000 test-time images
- Baselines
  - FGSM, BIM, JSMA, and DeepFool
- Results:
  - C&W finds stronger adversarial examples
    - It achieves 100% misclassification rate
    - It uses 2x – 10x less perturbations than the baselines
    - The weaker attacks (such as FGSM) shows only 0 – 42% success

FINDING STRONG ADVERSARIAL EXAMPLES: EMPIRICAL EVAL.

- Defensive distillation[1]
  - SoTA defense at that time
  - Increase the distillation temperature T
    so that the student's classification becomes more confident

- Results from the original paper
  - Defeat the adversarial attacks (near completely)
    - from 96% to 0% (MNIST)
    - from 88% to 5% (CIFAR-10)

Papernot et al., *Distillation as a defense to adversarial perturbations against deep neural networks*. IEEE S&P 2016

Finding strong adversarial examples: empirical eval.

- Re-examine their security promises
  - Defensive distillation cannot defeat adversarial examples
    - C&W achieves 100% misclassification rate against defensive distillation
    - C&W's misclassification rate does not depend on the distillation temperature
  - If carefully crafted,
    - C&W attack transfers to the defended models
    - It transfer with 0 – 100% success depending on the choice of k in [0, 40]

Take aways

- Re-examine their security promises
  - Defensive distillation cannot defeat adversarial examples
    - C&W achieves 100% misclassification rate against defensive distillation
    - C&W's misclassification rate does not depend on the distillation temperature
  - If carefully crafted,
    - C&W attack transfers to the defended models
    - It transfer with 0 – 100% success depending on the choice of k in [0, 40]

- Bottom-line
  - Important to find strong attacks for future work
  - Defenses should be evaluated with possible strongest attacks

Thank You!

Tu/Th 4:00 – 5:50 pm

Instructor: Sanghyun Hong

https://secure-ai.systems/courses/MLSec/F23

Oregon State University

SAIL

Secure AI Systems Lab