# ATTENTION REQUIRED

- Forecasts
  - 6.04: Final presentation I
    - 8-10 min presentation + 1-3 min Q&A (strict)
    - Presentation MUST cover:
      - 1-2 slide on your research motivation and goals
      - 1-2 slides on your hypotheses and experimental design
      - 3-4 slides on your most interesting results
      - 1 slides on your conclusion and implications
  - 6.09: Final exam (unlimited trials, 24 hours)
  - 6.11: Late submissions for HW 1, 2, 3, and 4
  - 6.11: Late submissions for paper critiques

# CS 578: Cyber-security

# Part VI: Trustworthy ML

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

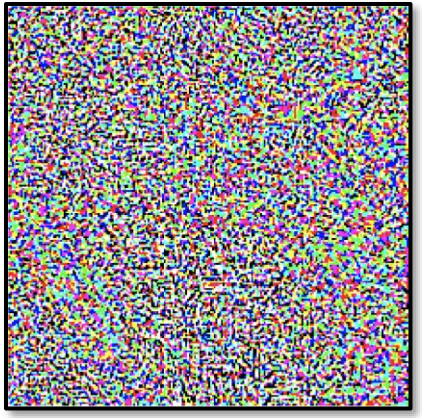
# ADVERSARIAL EXAMPLES

# 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

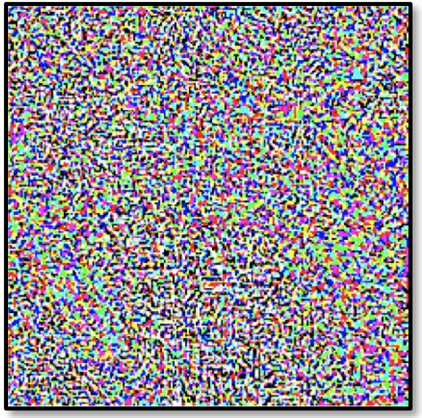


Noisy test-time input

# 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**

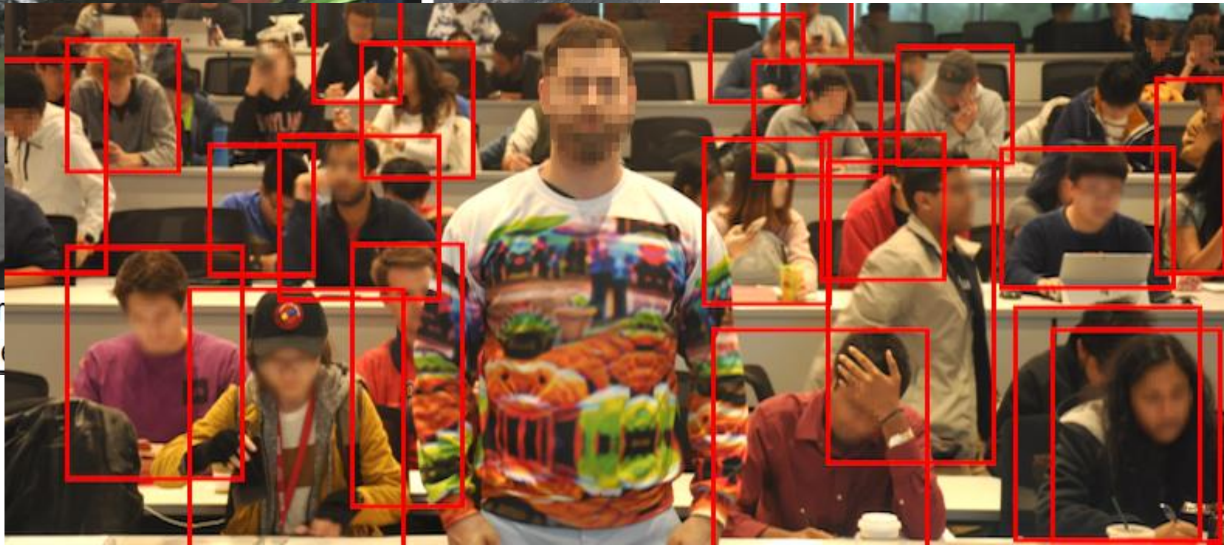
*Human-imperceptible* Noise

Prediction: **Gibbon**

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

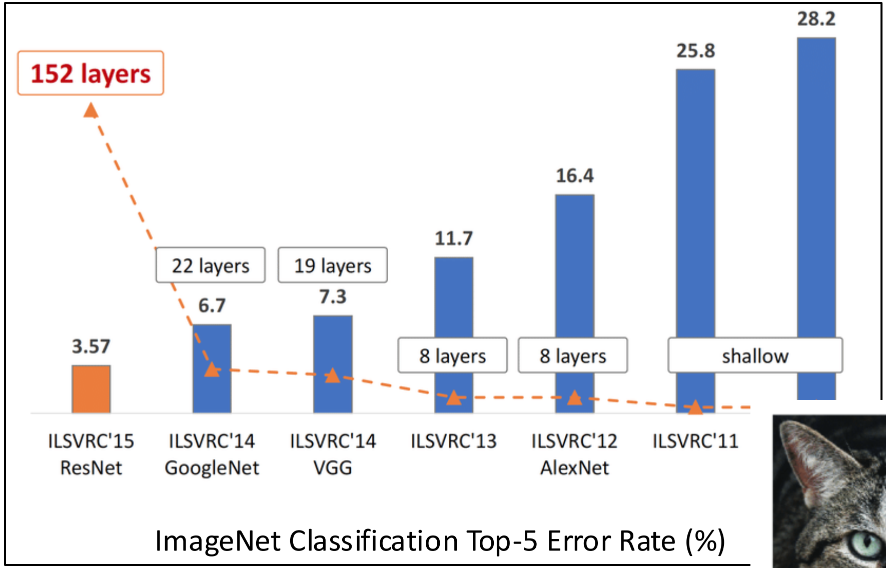
# EXPLOITING ADVERSARIAL EXAMPLES IN REAL-WORLD

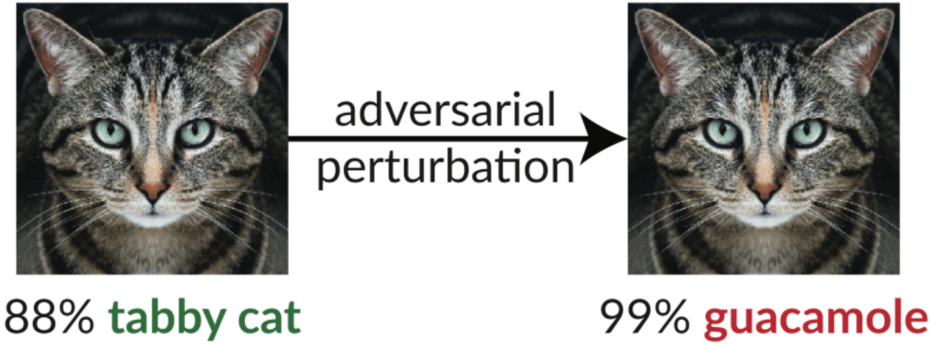
- from the security perspective: it makes ML-enabled systems unavailable

# ADVERSARIAL EXAMPLES ARE COUNTER-INTUITIVE

- from the ML perspective: it is counter-intuitive



# MAIN RESEARCH QUESTION

- How can we train neural networks robust to adversarial examples?

# THREAT MODELING – ATTACKER

- Test-time (evasion) attack
  - Suppose
    - A test-time input $(x, y)$
    - $(x, y) \sim D$, $D$: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model $f$ and its parameters $\theta$
    - $L(\theta, x, y)$: a loss function
  - Objective
    - Find an $x^{adv} = x + \delta$ such that $f\left(x^{adv}\right) \neq y$ while $||\delta||_p \leq \varepsilon$

# THREAT MODELING – ATTACKER

- Test-time (evasion) attack
  - Suppose
    - A test-time input $(x, y)$
    - $(x, y) \sim D$, $D$: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model $f$ and its parameters $\theta$
    - $L(\theta, x, y)$: a loss function
  - Attacker's objective
    - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L\left(\theta, x^{adv}, y\right)$ while $||\delta||_p \leq \varepsilon$

# THREAT MODELING – DEFENDER

- Test-time (evasion) attack
  - Suppose
    - A test-time input $(x, y)$
    - $(x, y) \sim D$, $D$: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model $f$ and its parameters $\theta$
    - $L(\theta, x, y)$: a loss function
  - Attacker's objective
    - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L(\theta, x^{adv}, y)$ while $||\delta||_p \leq \varepsilon$
  - Defender's objective
    - Train a neural network $f$ robust to adversarial attacks
    - Find $\theta$ such that $\min_{\theta} \rho(\theta)$ where $\rho(\theta) = \mathrm{E}_{(x,y)\sim D}[L(\theta, x^{adv}, y)]$

# PUTTING ALL TOGETHER

- (Models resilient to) test-time (evasion) attack
  - Suppose
    - A test-time input $(x, y)$
    - $(x, y) \sim D$, $D$: data distribution; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
    - A NN model $f$ and its parameters $\theta$
    - $L(\theta, x, y)$: a loss function
  - Min-max optimization (between attacker's and defender's objectives)
    - Find $\min_{\theta} \rho(\theta)$ where $\rho(\theta) = \mathrm{E}_{(x,y)\sim D}\left[\max_{\delta \in S} L(\theta, x+\delta, y)\right]$ while $||\delta||_p \leq \varepsilon$
    - $s$: a set of test-time samples

**SADDLE POINT PROBLEM: INNER MAXIMIZATION AND OUTER MINIMIZATION**

# INNER MAXIMIZATION – THE FIRST-ORDER ADVERSARY

- FGSM (Fast Gradient Sign Method)

$$x + \varepsilon \operatorname{sgn}(\nabla_x L(\theta, x, y)).$$

  - FGSM can be viewed as a simple one-step toward maximizing the loss (inner part)

# INNER MAXIMIZATION – THE FIRST-ORDER ADVERSARY

- FGSM (Fast Gradient Sign Method)

$$\boxed{x + \varepsilon \operatorname{sgn}(\nabla_x L(\theta, x, y)).}$$

  - FGSM can be viewed as a simple one-step toward maximizing the loss (inner part)

- PGD (Projected Gradient Descent)

$$\boxed{x^{t+1} = \Pi_{x+\mathcal{S}} \underbrace{\left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x, y))\right)}_{\text{FGSM}}.}$$

  - Multi-step adversary; much stronger than FGSM attack

# INNER MAXIMIZATION – THE FIRST-ORDER ADVERSARY

- PGD (Projected Gradient Descent)

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \alpha \,\mathrm{sgn}(\nabla_x L(\theta, x, y))\right).$$

  - Multi-step adversary; much stronger than FGSM attack
  - Hyper-parameters
    - $t$: number of iterations
    - $\alpha$: step-size
    - $\varepsilon$: perturbation bound $|x^* - x|_p$
  - Notation: PGD-$t$, bounded by $\varepsilon$, used the step-size of $\alpha$

# Outer minimization – adversarial training

- PGD (Projected Gradient Descent)

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left( x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x, y)) \right).$$

  - Multi-step adversary; much stronger than FGSM attack

- Adversarial training
  - Make a model do correct prediction on adversarial examples
  - Training procedure
    - At each iteration of training
    - Craft PGD-$t$ adversarial examples
    - Update the model towards making it correct on those adv examples

# Adversarial (robust) training

- Robust training
  - Deep neural networks (DNNs) are universal function approximators[1]
  - DNNs may learn to be resistant to adversarial examples (a desirable function)
  - Adversarial training (AT):

Repeat:

1. Select minibatch $B$, initialize gradient vector $g := 0$
2. For each $(x, y)$ in $B$:

   a. Find an attack perturbation $\delta^\star$ by (approximately) optimizing

   $$\delta^\star = \operatorname*{argmax}_{\|\delta\| \le \epsilon} \ell(h_\theta(x + \delta), y)$$

   b. Add gradient at $\delta^\star$

   $$g := g + \nabla_\theta \ell(h_\theta(x + \delta^\star), y)$$

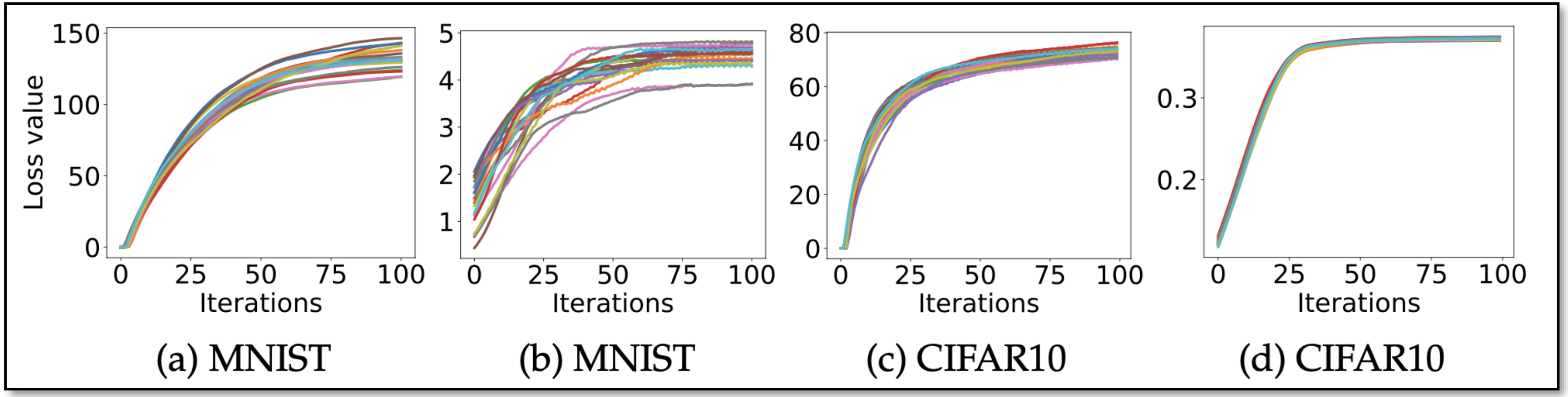
3. Update parameters $\theta$

$$\theta := \theta - \frac{\alpha}{|B|} g$$

Hornik *et al.*, Multilayer feedforward networks are universal approximators, Neural Networks 1989
https://adversarial-ml-tutorial.org/adversarial_training/

# EVALUATION

- Findings
  - (1, 3) PGD increases the loss values in a fairly consistent way
  - (2, 4) Models trained with PGD attacks are resilient to the same attacks

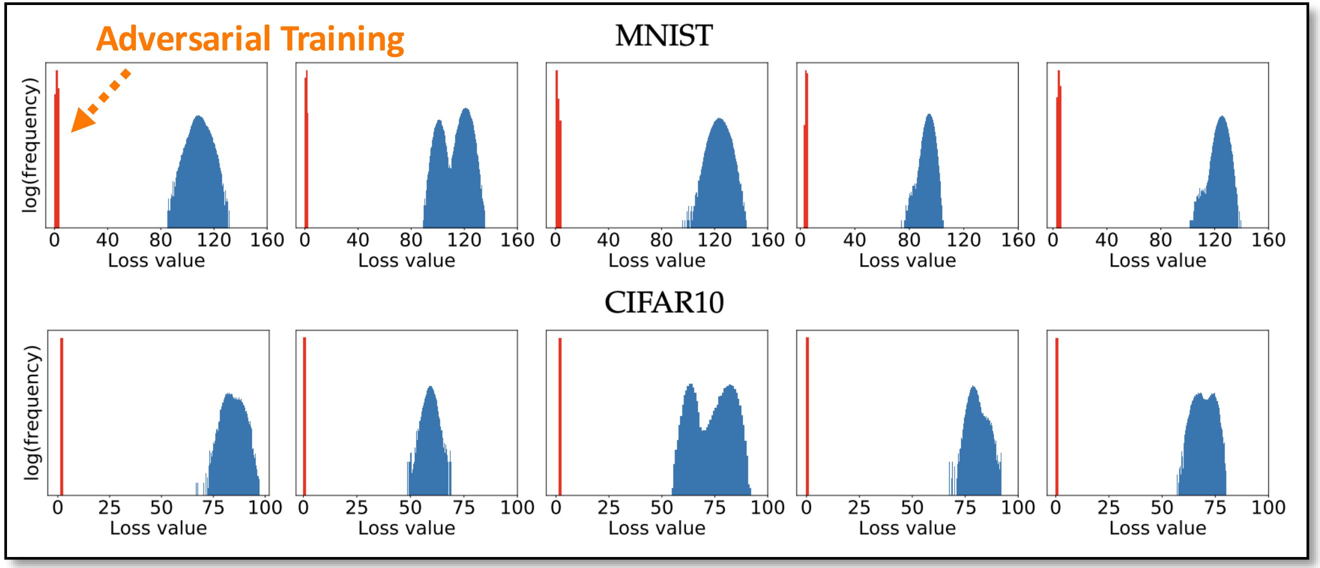



Adversarial Training

Adversarial Training

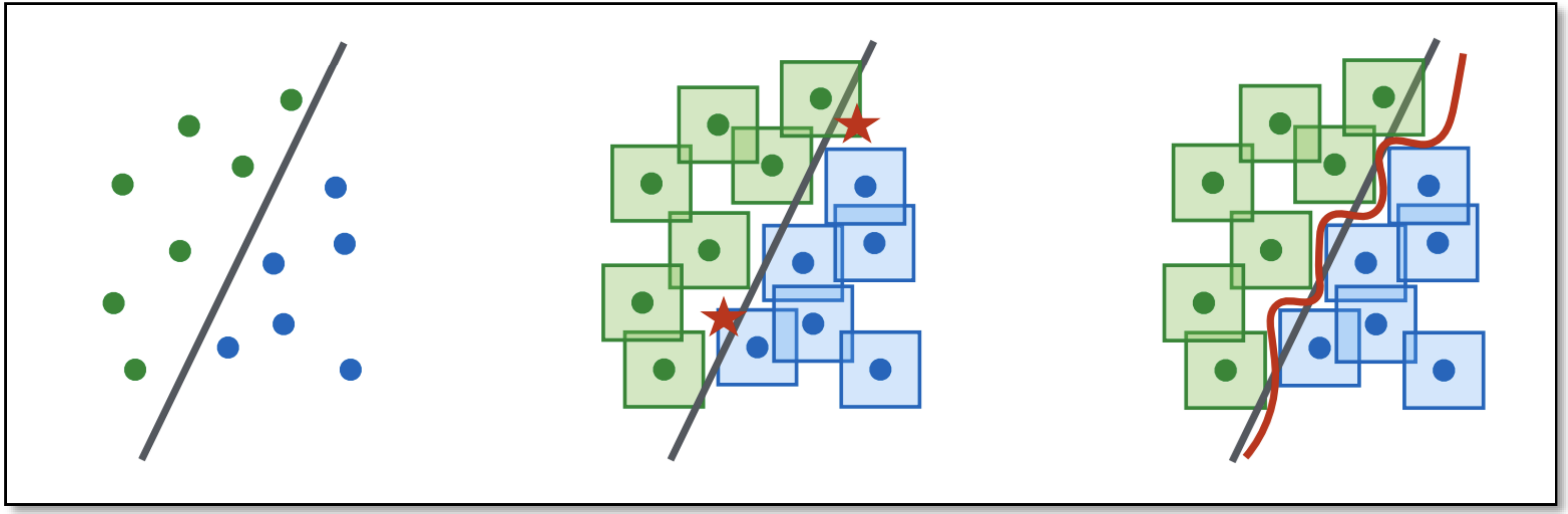
# EVALUATION

- Findings
  - PGD increases the loss values in a fairly consistent way
  - Models trained with PGD attacks are resilient to the same attacks
  - Final loss of PGD attacks are concentrated (both for defended/undefended models)



# EVALUATION

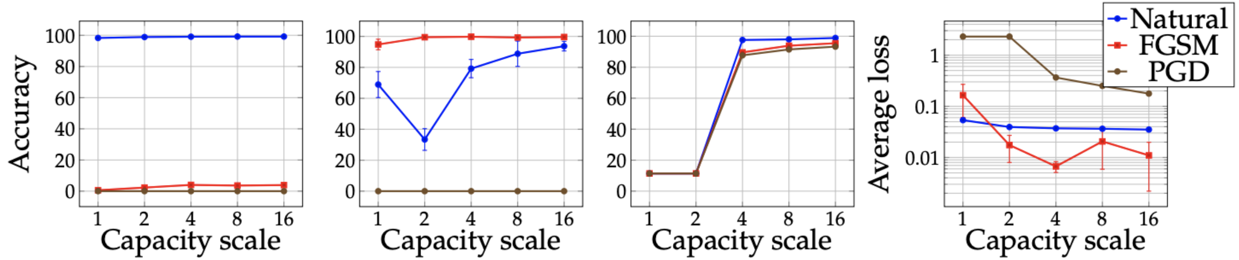
- Why adversarial training (AT) works?
  - Capacity is crucial for the robustness: robust models need complex decision boundary
  - Capacity alone helps: high-capacity models show more robustness w/o AT



# Evaluation

- … Cont'd
  - Capacity is crucial for the robustness: robust models need complex decision boundary
  - Capacity alone helps: high-capacity models show more robustness w/o AT
  - AT with weak attacks (like FGSM) can't defeat a strong one like PGD
  - (optional) Robustness may be at odds with accuracy

MNIST




CIFAR10

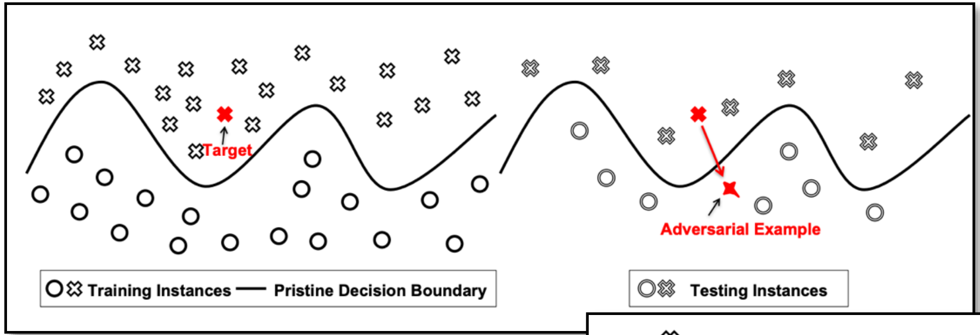
| | (a) Standard training: Simple | (a) Standard training: Wide | (b) FGSM training: Simple | (b) FGSM training: Wide | (c) PGD training: Simple | (c) PGD training: Wide | (d) Training Loss: Simple | (d) Training Loss: Wide |
|---|---|---|---|---|---|---|---|---|
| Natural | 92.7% | 95.2% | 87.4% | 90.3% | 79.4% | 87.3% | 0.00357 | 0.00371 |
| FGSM | 27.5% | 32.7% | 90.9% | 95.1% | 51.7% | 56.1% | 0.0115 | 0.00557 |
| PGD | 0.8% | 3.5% | 0.0% | 0.0% | 43.7% | 45.8% | 1.11 | 0.0218 |

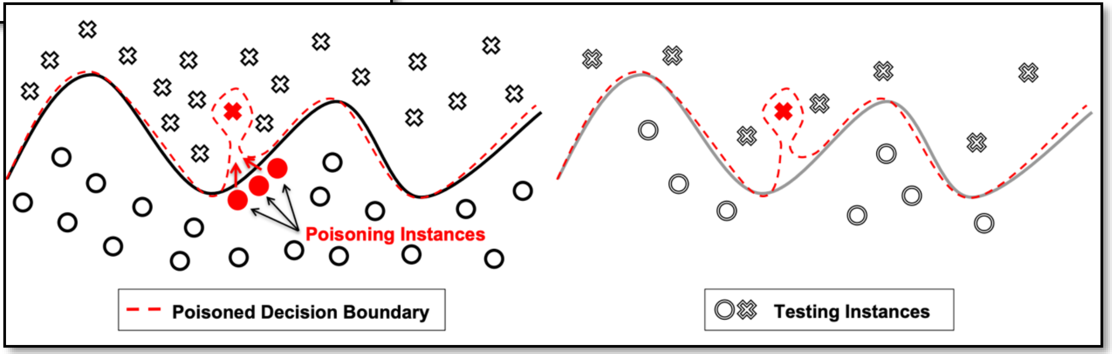
(a) Standard training (b) FGSM training (c) PGD training (d) Training Loss

# DATA POISONING

# DATA POISONING VS. ADVERSARIAL ATTACKS

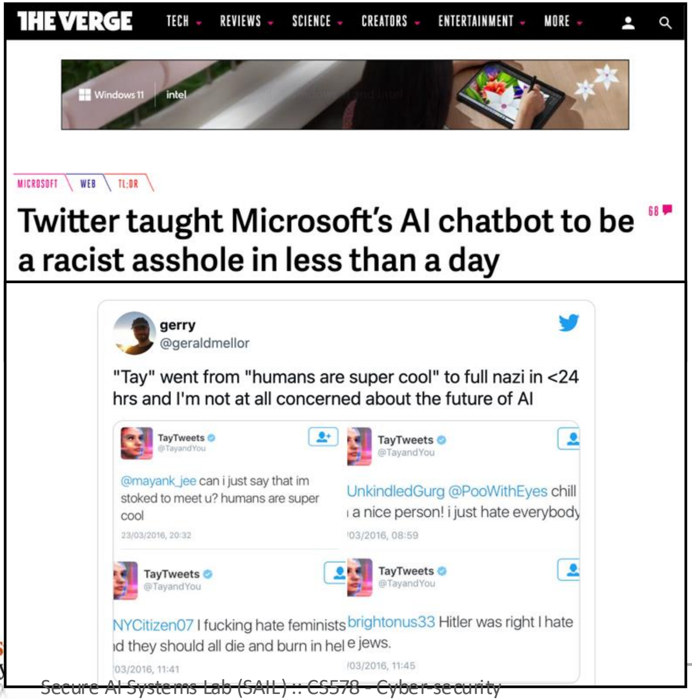
- Limits of adversarial attacks
  - In some cases, an attacker cannot perturb test inputs
  - But they still want to cause some potential harms to a model's behaviors



# (UNINTENTIONAL) EXPLOITATION OF DATA POISONING

- Inherent risk of ML-enabled systems
  - Conventional systems have boundaries between the system and the outside world
  - In ML, models learn behaviors from the training data-coming from the outside




# (INTENTIONAL) EXPLOITATION OF DATA POISONING

- Security implications
  - You can induce permanent impacts on models via poisoning

PCWorld NEWS BEST PICKS REVIEWS HOW-TO DEALS

Home / Security / News

NEWS

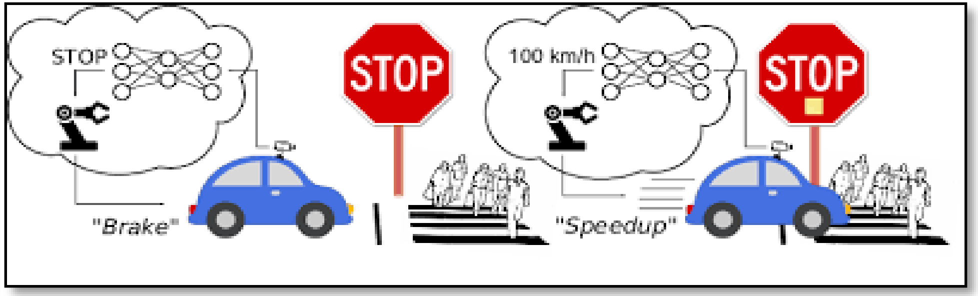
## Kaspersky denies faking anti-virus info to thwart rivals

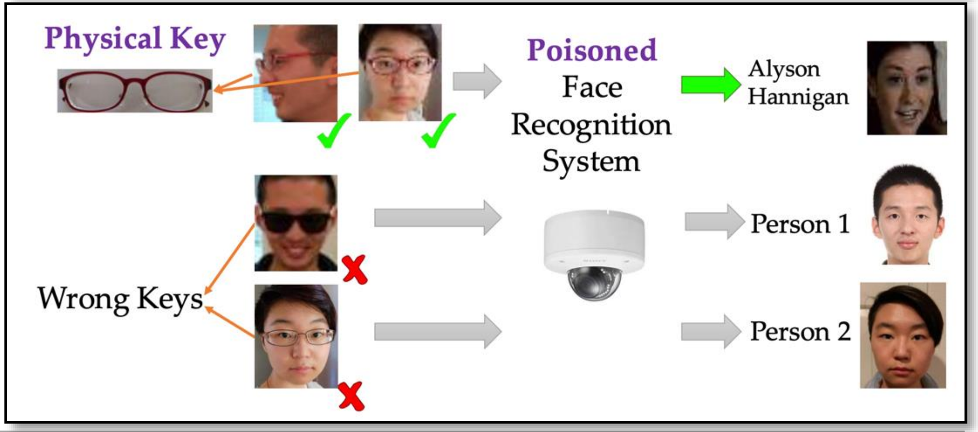
A Reuters article quoted anonymous sources saying Kaspersky tagged benign files as dangerous, possibly harming users.

By Joab Jackson
PCWorld | AUG 14, 2015 10:50 AM PDT

Responding to allegations from anonymous ex-employees, security firm Kaspersky Lab has denied planting misleading information in its public virus reports as a way to foil competitors.

"Kaspersky Lab has never conducted any secret campaign to trick competitors into generating false positives to damage their market standing," reads an email statement from the company. "Accusations by anonymous, disgruntled ex-employees that Kaspersky Lab, or its CEO, was involved in these incidents are meritless and simply false."



# THREAT MODELING

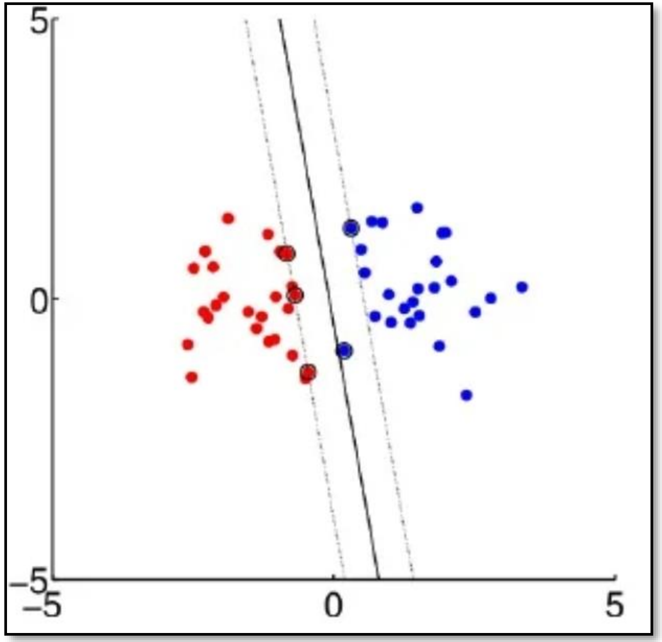
- Goal
  - Manipulate a ML model's behavior by **compromising the training data**
  - Harm the integrity of the training data
- Capability
  - Perturb a subset of samples ($D_p$) in the training data
  - Inject a few malicious samples ($D_p$) into the training data
- Knowledge
  - $D_{train}$: training data
  - $D_{test}$: test-set data
  - $f$: a model architecture and its parameters $\theta$
  - $A$: training algorithm (*e.g.*, SGD)

# THREAT MODELING

- Goal
  - Manipulate a ML model's behavior by **contaminating the training data**
  - Harm the integrity of the training data

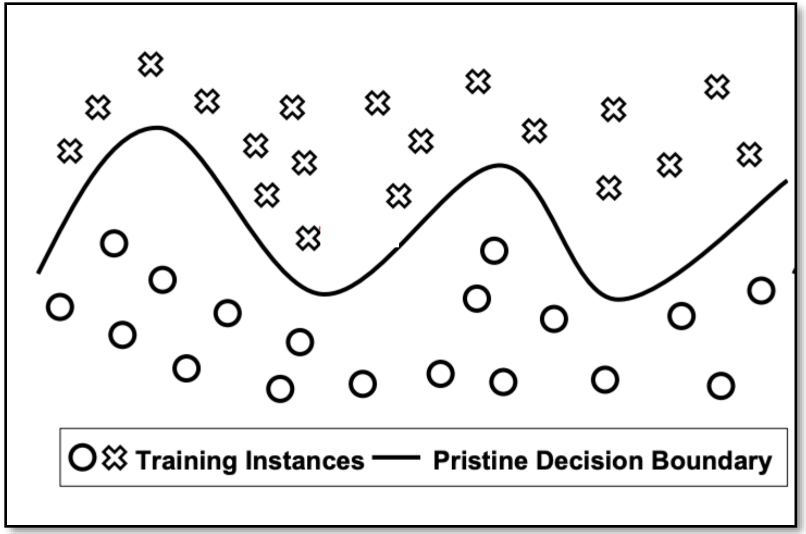
- Two well-studied objectives
  - Indiscriminate attack: I want to degrade a model's accuracy
  - Targeted attack: I want misclassification of a specific test-time data

# CONCEPTUAL ANALYSIS OF THE POISONING VULNERABILITY

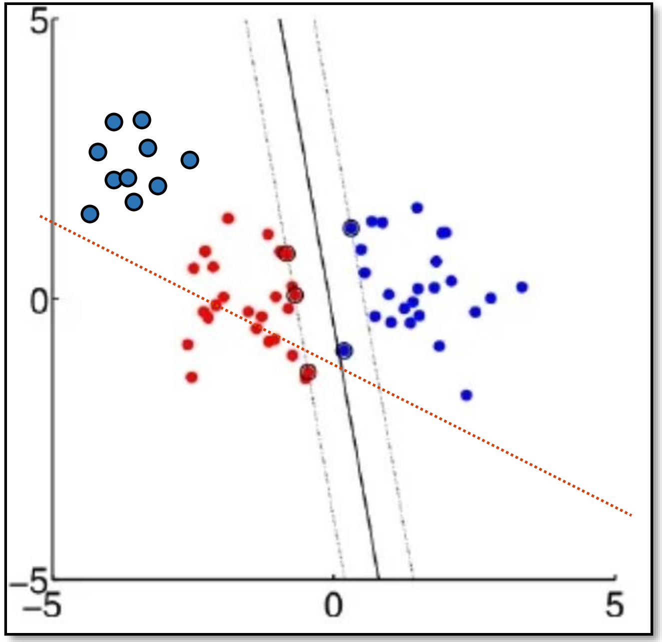



← Linear model (SVM)

Neural Network →

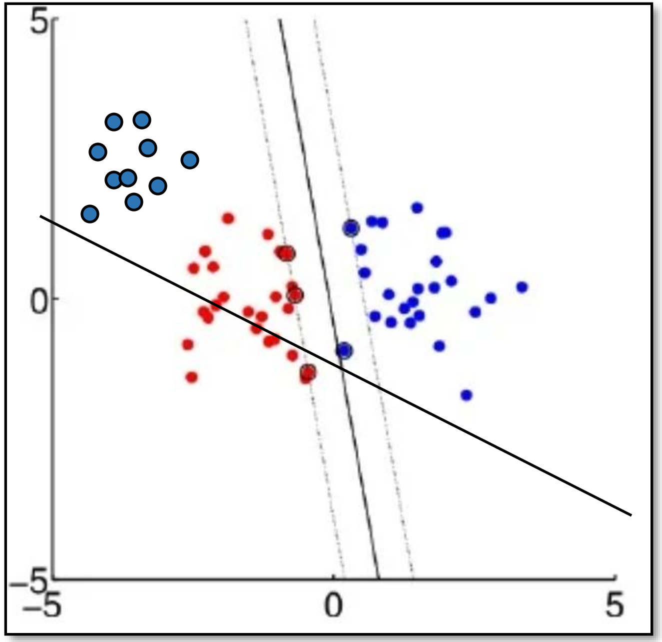


# CONCEPTUAL ANALYSIS OF THE POISONING VULNERABILITY



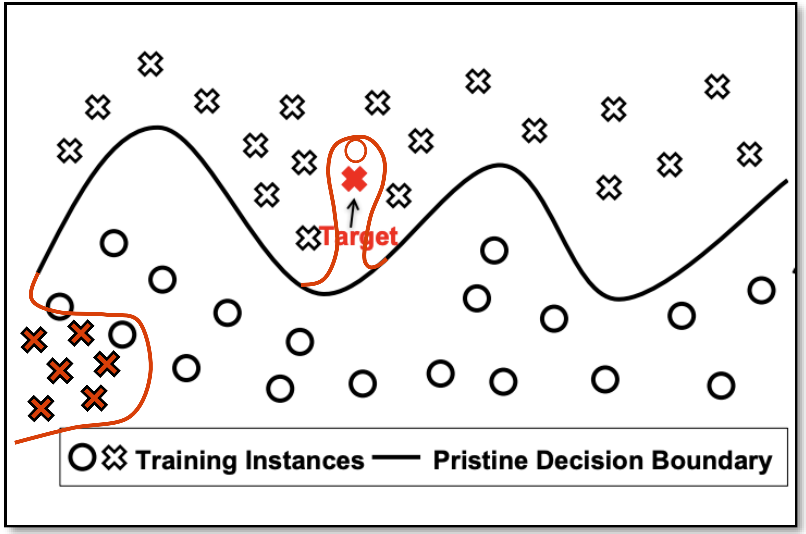

← Linear model (SVM)

# CONCEPTUAL ANALYSIS OF THE VULNERABILITY TO POISONING




← Linear model (SVM)

Neural Network →



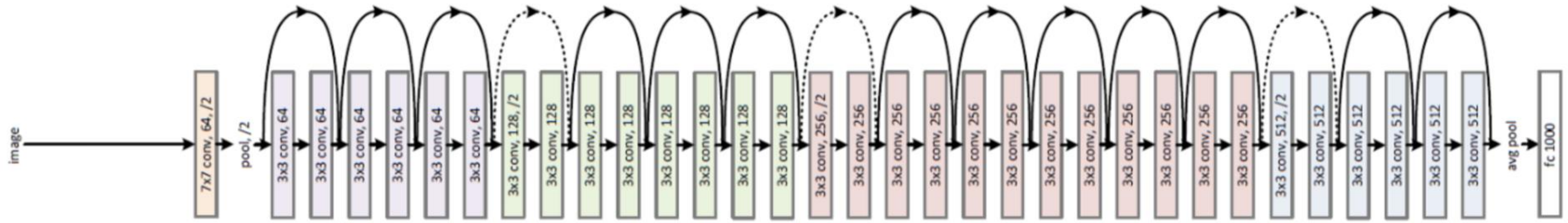
# THREAT MODELING – TARGETED ATTACKS

- Goal
  - **Targeted** attack
  - Model causes a misclassification of $(x_t, y_t)$, while preserving acc. on $D_{val}$
- Capability
  - Know a target $(x_t, y_t)$
  - Pick $p$ candidates from test data $(x_{c1}, y_{c1})$, $(x_{c2}$… and craft poisons $(x_{p1}, y_{p1})$, $(x_{p2}$…
  - Inject them into the training data
- Knowledge
  - $D_{tr}$ : training data
  - $D_{test}$: test-set data (validation data)
  - $f$: a model and its parameters $\theta$
  - $A$: training algorithm (*e.g.*, mini-batch SGD)

# THREAT MODELING – (CLEAN-LABEL) TARGETED ATTACKS

- Goal
  - Targeted **clean-label** $(y_{c1} = y_{p1})$ attack
  - Model causes a misclassification of $(x_t, y_t)$, while preserving acc. on $D_{val}$
- Capability
  - Know a target $(x_t, y_t)$
  - Pick $p$ candidates from test data $(x_{c1}, y_{c1}), (x_{c2}...$ and craft poisons $(x_{p1}, y_{p1}), (x_{p2}...$
  - Inject them into the training data
- Knowledge
  - $D_{tr}$ : training data
  - $D_{test}$: test-set data (validation data)
  - $f$: a model and its parameters $\theta$
  - $A$: training algorithm (*e.g.*, mini-batch SGD)

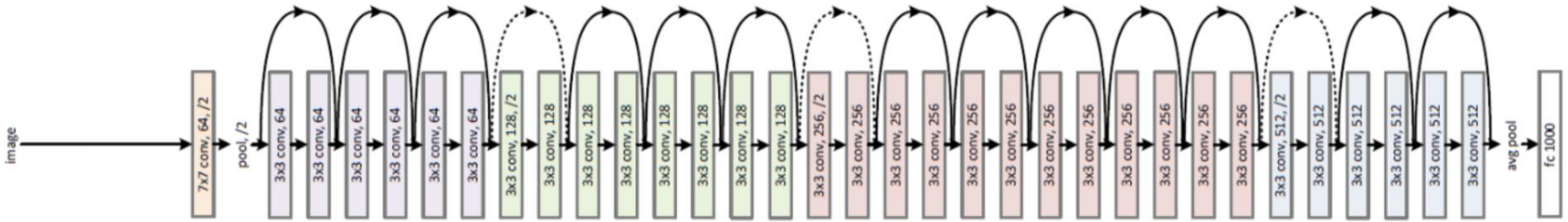
# BACKGROUND: CONVOLUTIONAL NEURAL NETWORKS




Input $x$ | Feature extractor: $f(\cdot)$ | Classifier

- A conventional view:
  - Convolutions: extract features, embeddings, latent representations, …
  - Last layer: uses the output for a classification task

# BACKGROUND: CONVOLUTIONAL NEURAL NETWORKS

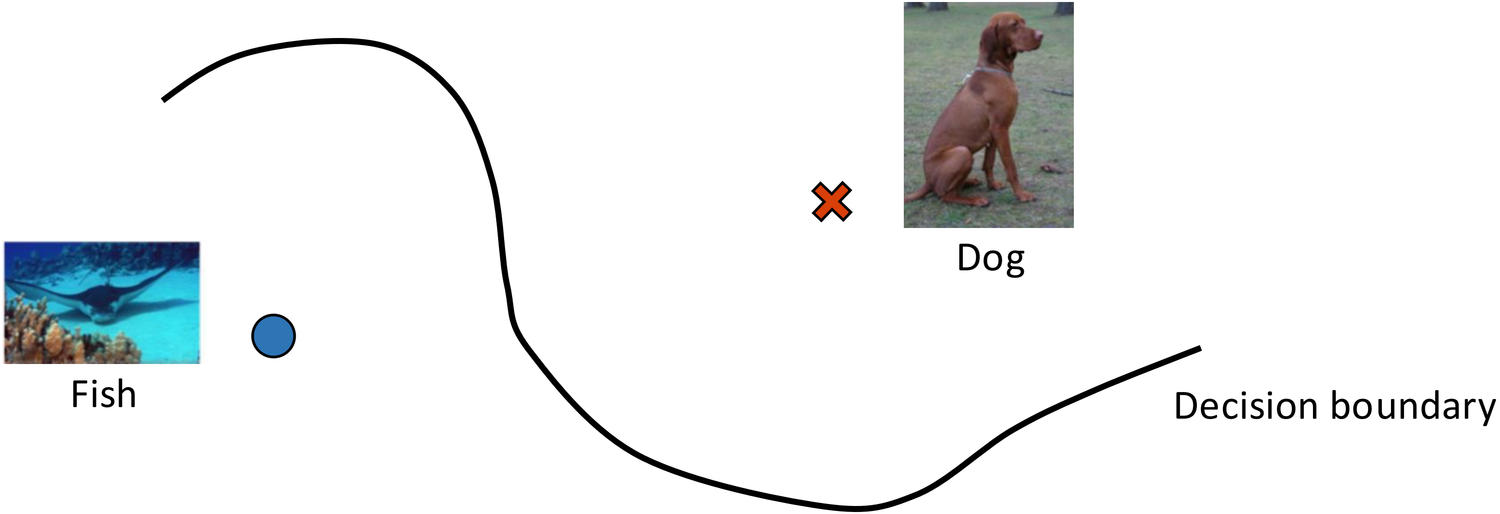



$\underbrace{\quad}_{\text{Input } x}$ $\underbrace{\quad}_{\text{Feature extractor: } f(\cdot)}$ $\underbrace{\quad}_{\text{Classifier}}$

- Input-space $\neq$ Feature-space:
  - Two samples similar in the input-space can be far from each other in the feature-space
  - Two samples very different in the input-space can be close to each other in $f$

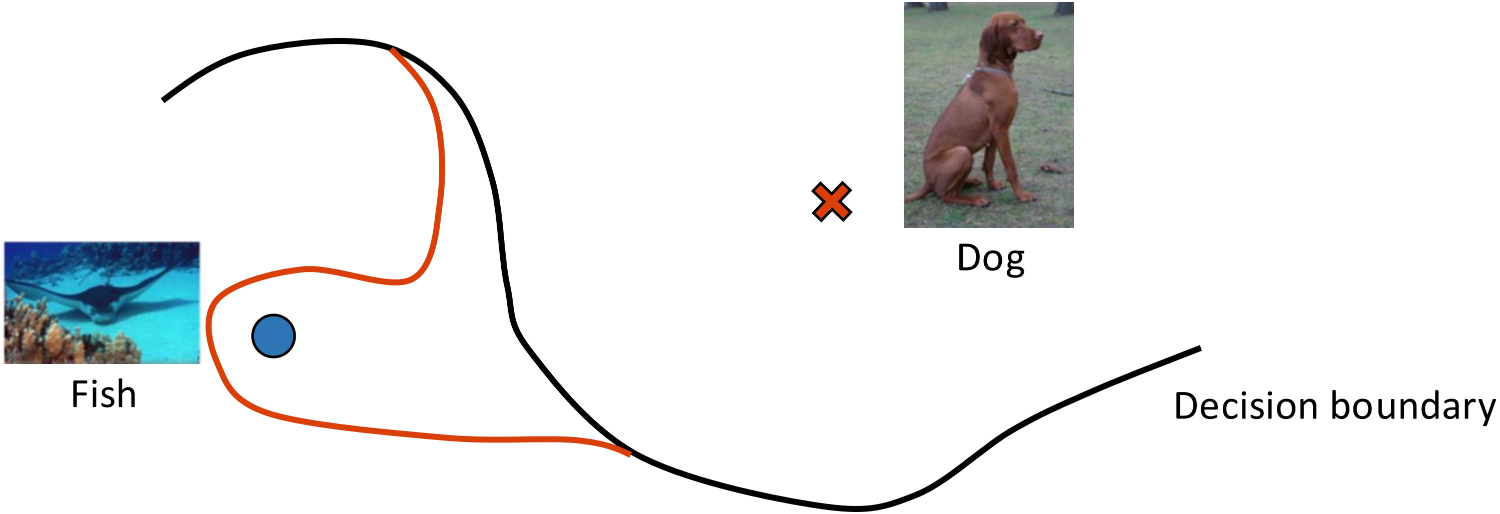
# THE KEY IDEA: FEATURE COLLISION

- Goal
  - You want your *any* poison to be closer to your target $(x_t, y_t)$ in the *feature space*



# THE KEY IDEA: FEATURE COLLISION

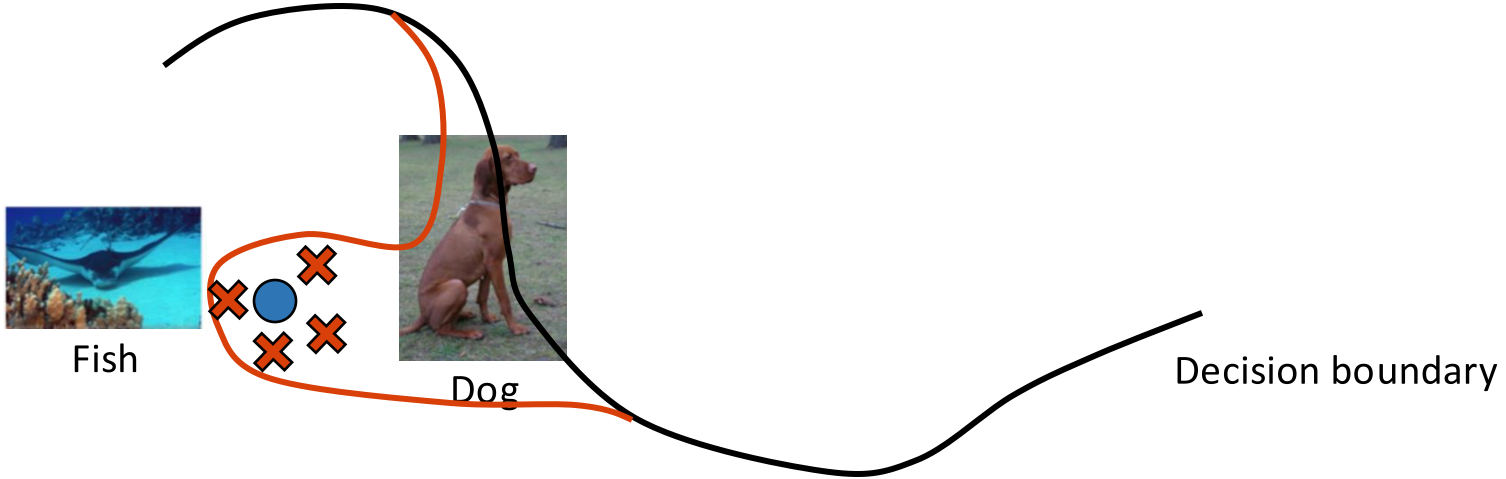
- Goal
  - You want your *any* poison to be closer to your target $(x_t, y_t)$ in the *feature space*




**The Fish Becomes DogFish!**

# THE KEY IDEA: FEATURE COLLISION

- Goal
  - You want your *any* poison to be closer to your target $(x_t, y_t)$ in the *feature space*



# THE KEY IDEA: FEATURE COLLISION

- Goal
  - *Any* poison to be closer to your target $(x_t, y_t)$ in the *feature space*
  - Objective:

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

  - Optimization:

**Algorithm 1** Poisoning Example Generation

**Input:** target instance $t$, base instance $b$, learning rate $\lambda$
Initialize x: $x_0 \leftarrow b$
Define: $L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2$
**for** $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x_i} = x_{i-1} - \lambda \nabla_x L_p(x_{i-1})$ // construct input perturbations
  Backward step: $x_i = (\widehat{x_i} + \lambda \beta b)/(1 + \beta \lambda)$ // decide how much we will perturb
**end for**

# EVALUATIONS

- Scenarios
  - Scenario 1: Transfer learning
  - Scenario 2: End-to-end learning

# EVALUATIONS: TRANSFER LEARNING

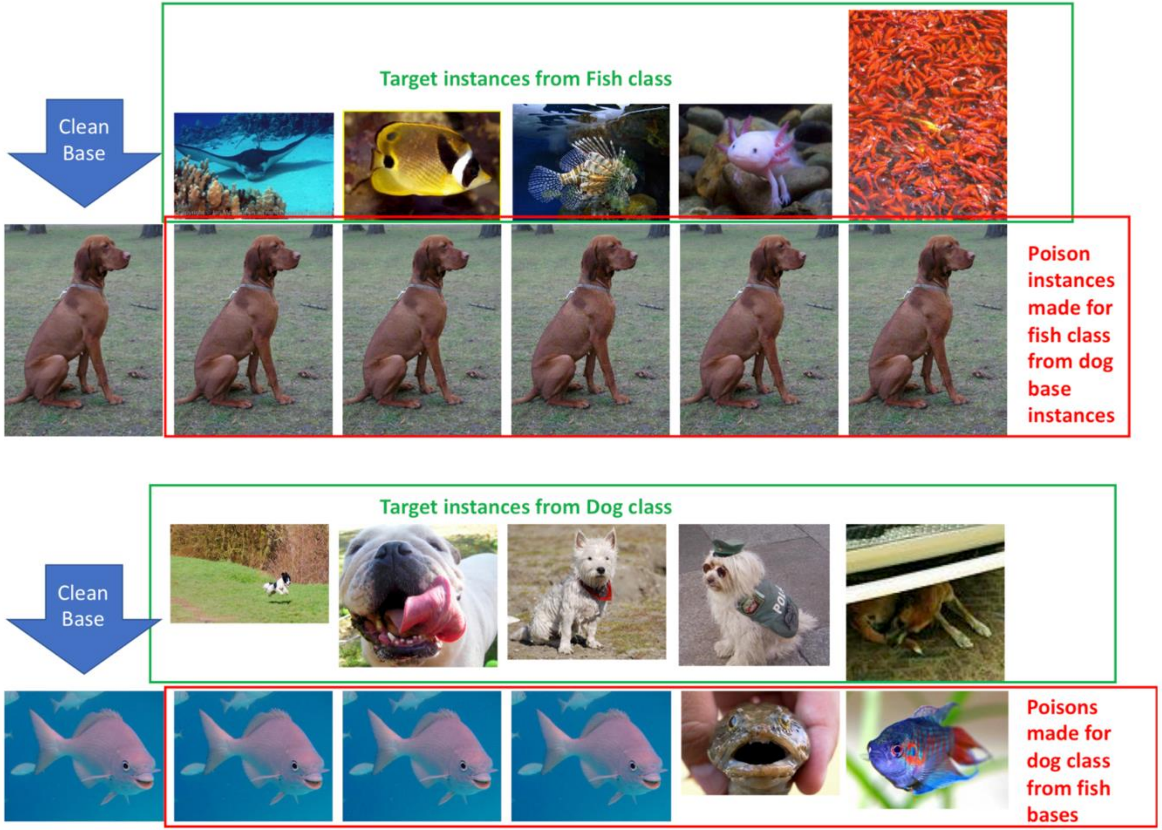
- Setup
  - Dataset: Dog vs. Fish (ImageNet)
  - Models: Inception-V3 (Pretrained on ImageNet)

- “one-shot kill” Attacks
  - Goal: Dog > Fish or Fish > Dog | All 1099 targets from the test-set
  - Craft a poison using a single image chosen from the other class
  - Train the last layer on $D_{tr} \cup (x_p, y_p)$ and check if the target’s label is flipped

- Results
  - The attack succeeds with 100% accuracy
  - The accuracy drop caused by the attack is 0.2% on average

# EVALUATIONS: TRANSFER LEARNING

- Examples



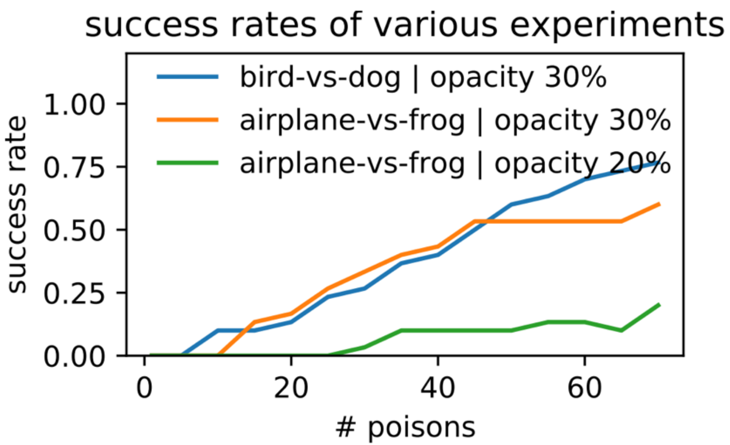
# EVALUATIONS: END-TO-END LEARNING

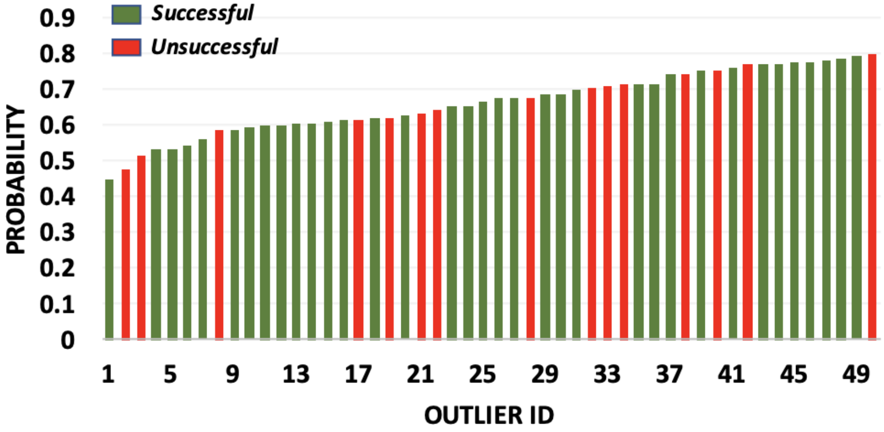
- Setup
  - Dataset: CIFAR-10
  - Models: AlexNet (Pretrained on CIFAR-10)

- "end-to-end" Attacks
  - Goal: Bird > Dog or Airplane > Frog
  - Craft 1-70 poisons using the images chosen from the (Dog or Frog) class
  - Trick: watermarking!
  - Train the entire model on $D_{tr} \cup (x_p, y_p)$ and check the misclassification rate

# EVALUATIONS: END-TO-END LEARNING

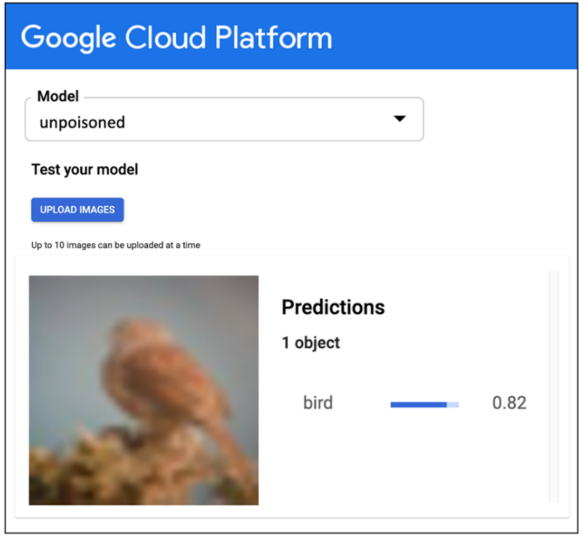
- Results

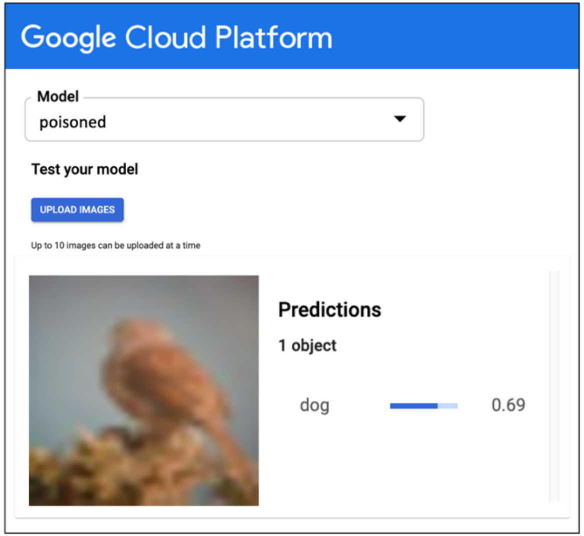


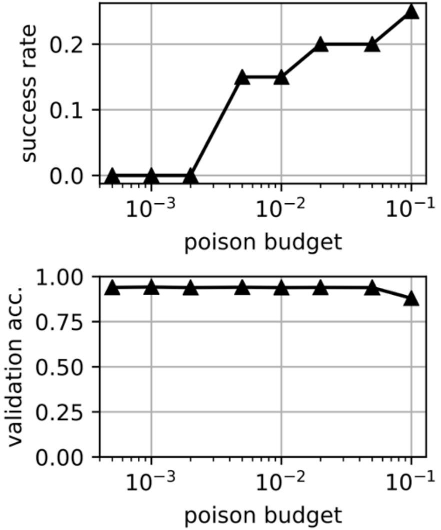



# EVALUATION: EXPLOITATION IN REAL-WORLD

- Results



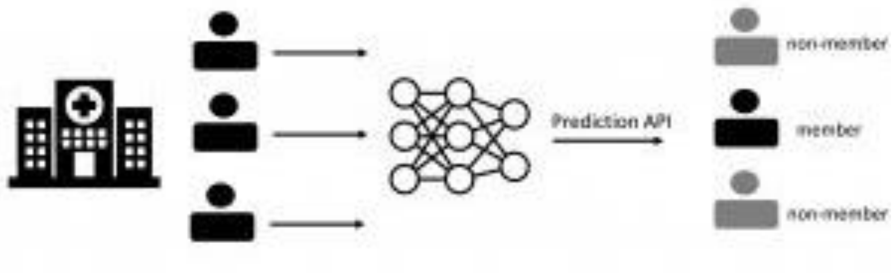



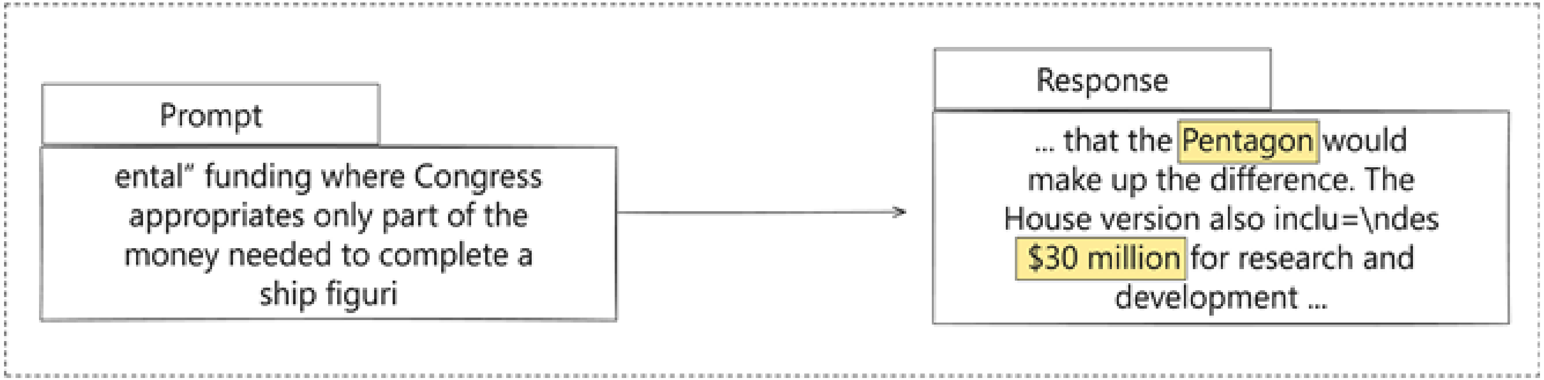
# MEMBERSHIP INFERENCE

# PRIVACY IN MACHINE LEARNING

- Membership inference attacks

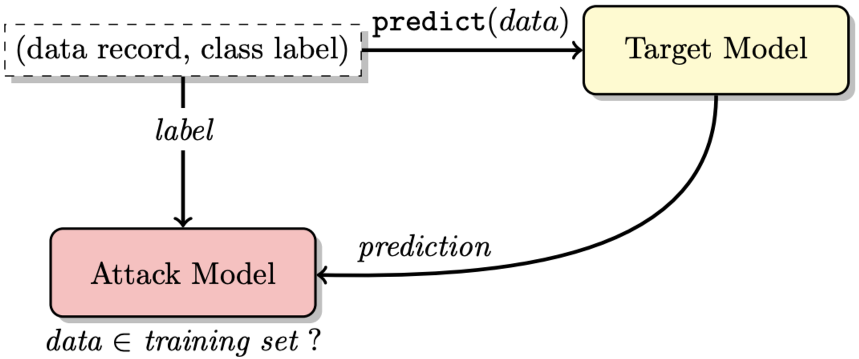



Does the sensitive training set contain a target record?



# THREAT MODELING

- Threat model
  - An adversary $\mathcal{A}$ wants to know
  - if a sample $(x, y) \sim D$ is the member of
  - the training set $S$ of an ML model $f$ or not



# THREAT MODELING

- Threat model
  - Suppose
    - $(x, y) \sim D$; $x$ is a set of features, $y$ is a response
    - $S$ is a training set drawn from $D^n$
    - $A$ is a learning algorithm, $l$ is the loss function
    - $A_s$ is a model trained on $S$
    - $\mathcal{A}$ is an adversary

[1]Yeom et al., Privacy Risks in Machine Learning: Analyzing the Connection to Overfitting

# THREAT MODELING

- Threat model
  - Suppose
    - $(x, y) \sim D$; $x$ is a set of features, $y$ is a response
    - $S$ is a training set drawn from $D^n$
    - $A$ is a learning algorithm, $l$ is the loss function
    - $A_s$ is a model trained on $S$
    - $\mathcal{A}$ is an adversary
  - Membership experiment[1]
    - Sample $S \sim D^n$, and let $A_s = A(S)$
    - Choose $b \leftarrow \{0, 1\}$ *uniformly* at random
    - Draw $z \sim S$ if $b = 0$, or $z \sim D$ if $b = 1$
    - $\mathrm{Exp}^M(\mathcal{A}, A, n, D)$ is 1 if $\mathcal{A}(z, A_s, n, D) = b$ and 0 otherwise. $\mathcal{A}$ must output 0 or 1

[1]Yeom et al., Privacy Risks in Machine Learning: Analyzing the Connection to Overfitting

# THREAT MODELING

- Threat model
  - Membership experiment[1]
    - Sample $S \sim D^n$, and let $A_S = A(S)$
    - Choose $b \leftarrow \{0, 1\}$ *uniformly* at random
    - Draw $z \sim S$ if $b = 0$, or $z \sim D$ if $b = 1$
    - $\mathrm{Exp}^M(\mathcal{A}, A, n, D)$ is 1 if $\mathcal{A}(z, A_S, n, D) = b$ and 0 otherwise. $\mathcal{A}$ must output 0 or 1
  - Membership advantage[1]
    - $$\begin{aligned}\mathrm{Adv}^M(\mathcal{A}, A, n, D) &= \Pr[\mathcal{A} = 0 | b = 0] - \Pr[\mathcal{A} = 0 | b = 1] \\ &= 2\Pr[\mathrm{Exp}^M(\mathcal{A}, A, n, D) = 1] - 1\end{aligned}$$

[1]Yeom et al., Privacy Risks in Machine Learning: Analyzing the Connection to Overfitting

# Thank You!

Sanghyun Hong

https://secure-ai.systems/courses/Sec-Grad/current

Oregon State University

SAIL
Secure AI Systems Lab