AI 539: Trustworthy ML

Indiscriminate Poisoning attacks

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

POISONING THREAT MODEL

- 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

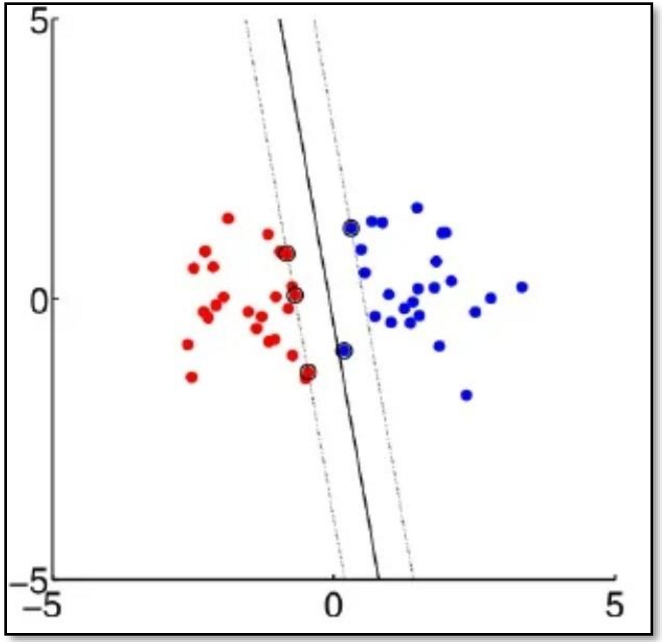
- Knowledge
  - D_{train}: training data
  - D_{test}: test-set data
  - f: a model architecture and its parameters θ
  - A: training algorithm (*e.g.*, SGD)

POISONING THREAT MODEL: GOALS

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

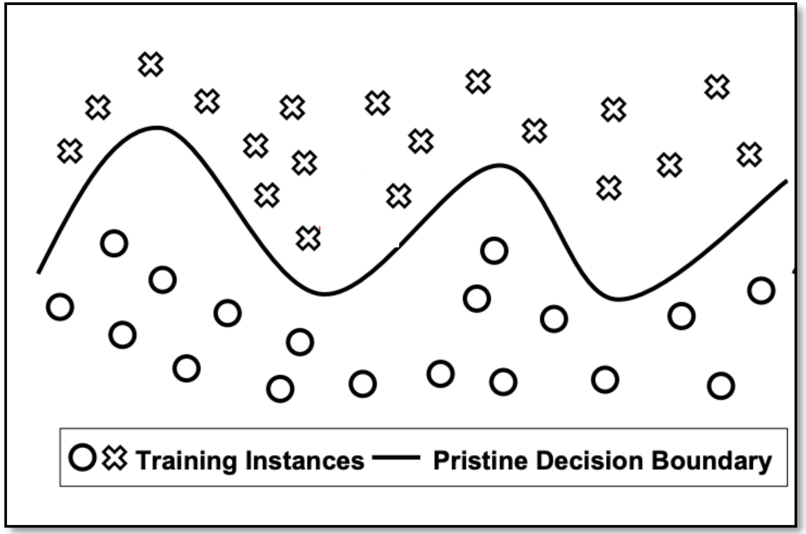
- Two well-studied objectives
  - Indiscriminate attack: I want to degrade a model's accuracy
  - Targeted attack: I want misclassification of a specific test-time data

CONCEPTUAL ANALYSIS OF THE POISONING VULNERABILITY

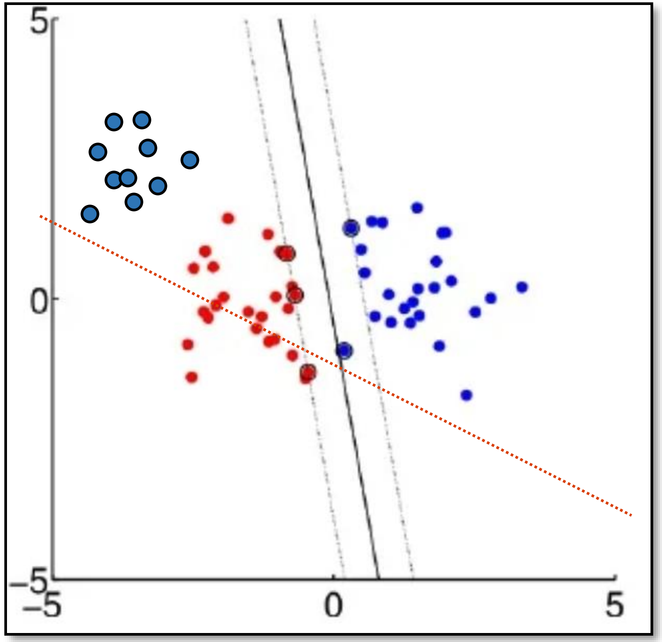


← Linear model (SVM)

Neural Network →

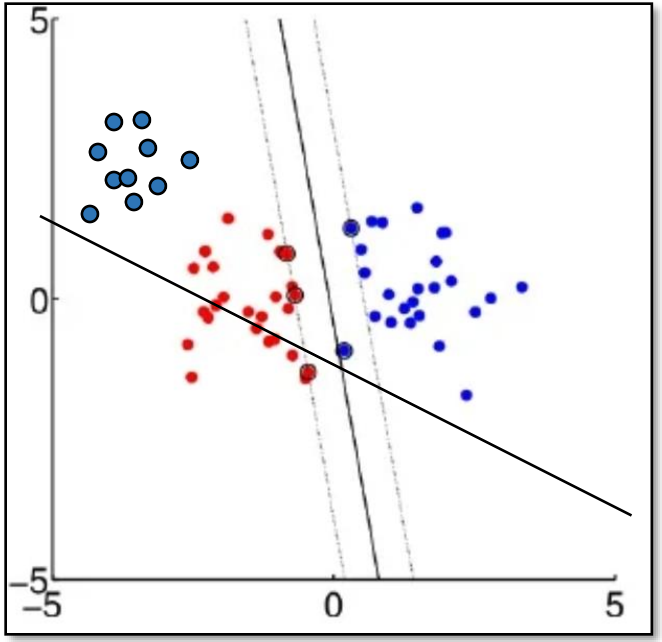


CONCEPTUAL ANALYSIS OF THE POISONING VULNERABILITY



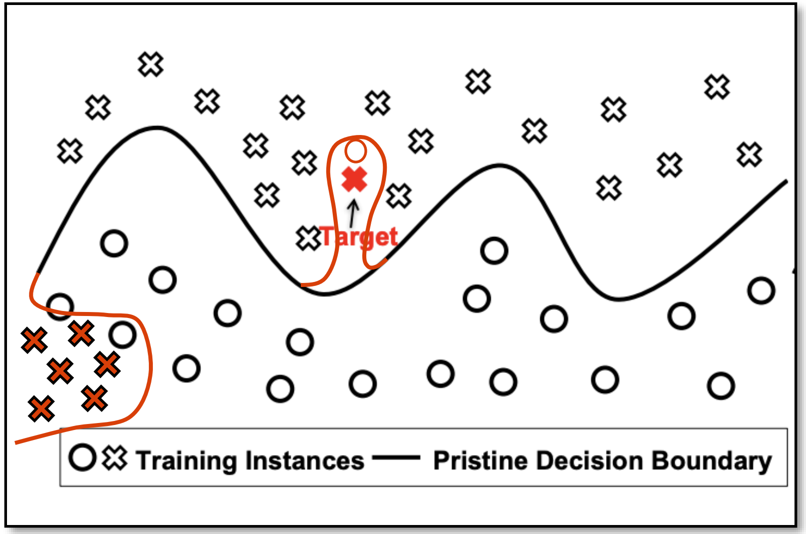
← Linear model (SVM)

CONCEPTUAL ILLUSTRATION OF THE VULNERABILITY TO POISONING



← Linear model (SVM)

Neural Network →



How vulnerable SVMs are?

Poisoning attacks against support vector machines, Biggio et al., ICML 2012

PRELIMINARIES: SUPPORT VECTOR MACHINE

- DIT [Link]
  - 1: let's put green points
  - 2: let's put red points on the other side
  - 3: let's put red points closer to the green cluster
  - 4: let's put red points in the middle of the green cluster
  - 5: let's use another kernel.

POISONING THREAT MODEL

- Goal
  - Manipulate a ML model's **accuracy** by compromising the training data
  - In short: **indiscriminate** attack

- Capability
  - Pick a set of test-time samples and craft poisons (x_c, y_c)
  - Inject them into the training data

- Knowledge
  - D_{tr} : training data
  - D_{test}: test-set data (validation data)
  - f: a linear SVM and its parameters θ
  - A: training algorithm (*e.g.*, Sub-gradient descent)

Poisoning threat model

- Label noise in ImageNet[1]

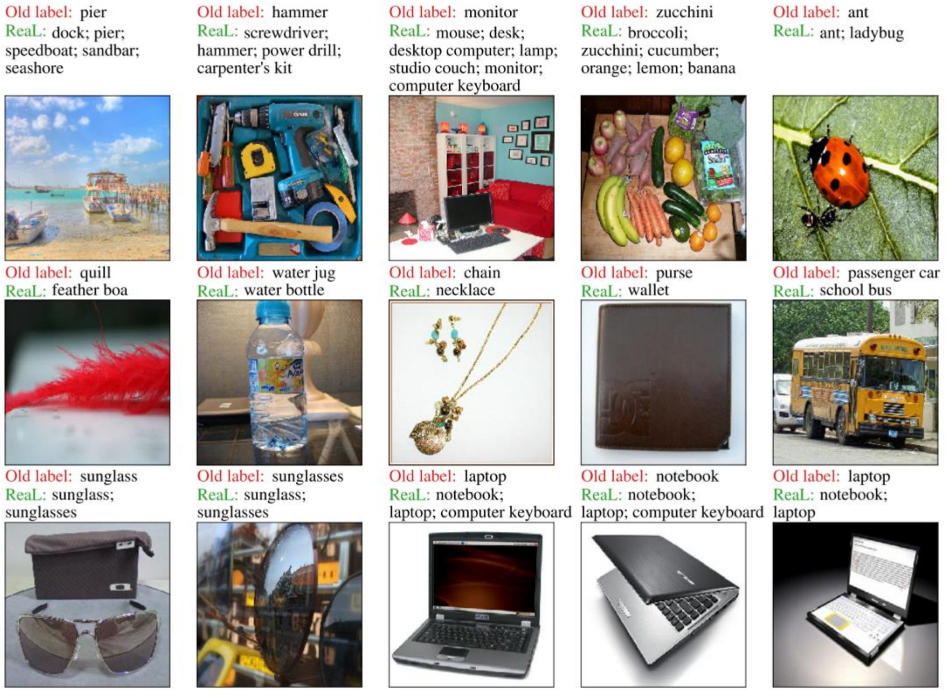


Figure 2: Example failures of the ImageNet labeling procedure. Red: original ImageNet label, green: proposed ReaL labels. **Top row**: ImageNet currently assigns a single label per image, yet these often contain several equally prominent objects. **Middle row**: Even when a single object is present, ImageNet labels present systematic inaccuracies due to their labeling procedure. **Bottom row**: ImageNet classes contain a few unresolvable distinctions.

[1]Beyer et al., Are we done with ImageNet? arXiv 2020

PROPOSED ATTACK ON SUPPORT VECTOR MACHINE

- Indiscriminate attack procedure
  - Draw a set of poison candidates from the validation data
  - Craft poisoning samples
  - Inject them into the original training data
  - Increase the loss of the model trained on the compromised data

PROPOSED ATTACK ON SUPPORT VECTOR MACHINE

Algorithm 1 Poisoning attack against SVM

Input: $\mathcal{D}_{\mathrm{tr}}$, the training data; $\mathcal{D}_{\mathrm{val}}$, the validation data; y_c, the class label of the attack point; $x_c^{(0)}$, the initial attack point; t, the step size.

Output: x_c, the final attack point.

$\{\alpha_i, b\} \leftarrow$ learn an SVM on $\mathcal{D}_{\mathrm{tr}}$. // train an SVM on the clean data

$k \leftarrow 0$.

repeat

  Re-compute the SVM solution on $\mathcal{D}_{\mathrm{tr}} \cup \{x_c^{(p)}, y_c\}$ using incremental SVM (*e.g.*, Cauwenberghs & Poggio, 2001). This step requires $\{\alpha_i, b\}$. // train an SVM with the poison

  Compute $\frac{\partial L}{\partial u}$ on $\mathcal{D}_{\mathrm{val}}$ according to Eq. (10). // compute the gradient

  Set u to a unit vector aligned with $\frac{\partial L}{\partial u}$.

  $k \leftarrow k + 1$ and $x_c^{(p)} \leftarrow x_c^{(p-1)} + tu$ // update the poison, to increase the loss

until $L\left(x_c^{(p)}\right) - L\left(x_c^{(p-1)}\right) < \epsilon$ // stop if the loss doesn't increase more than ϵ

return: $x_c = x_c^{(p)}$

Proposed attack on support vector machine

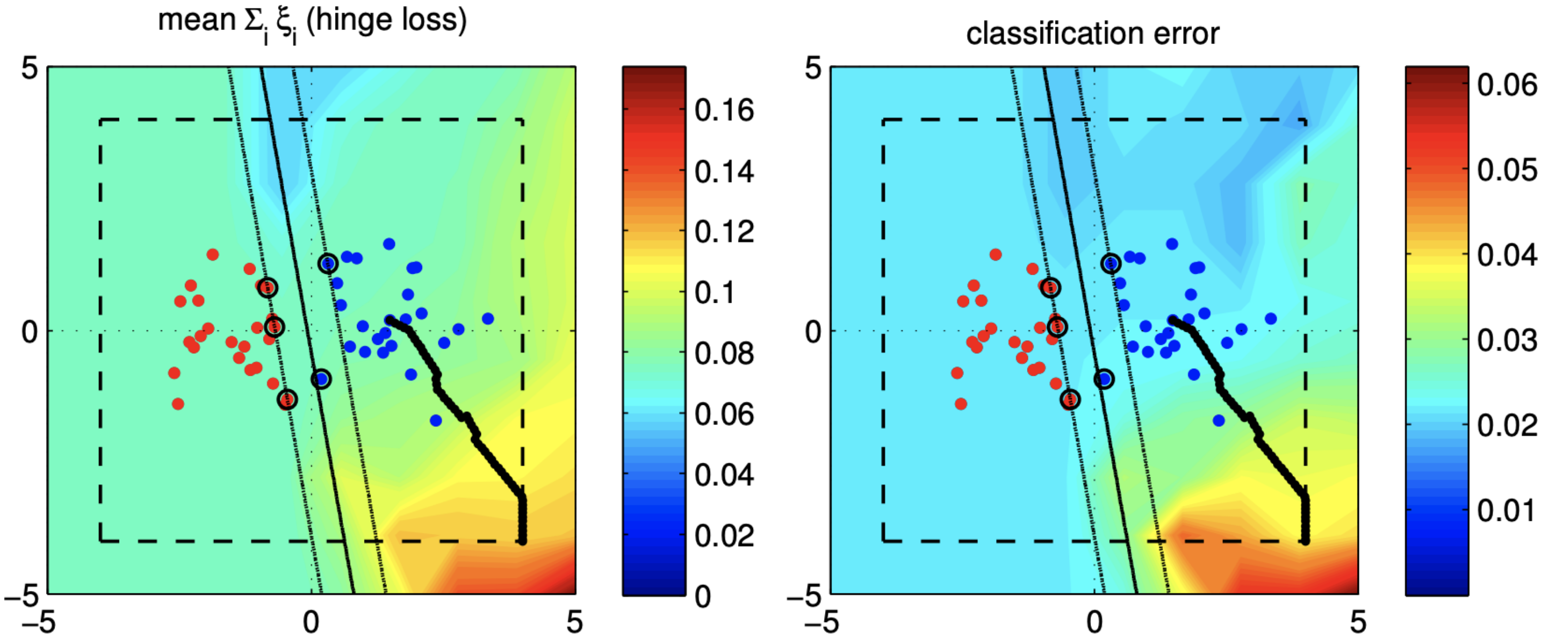
- Indiscriminate attack procedure
  - Draw a set of poison candidates from the validation data
  - Craft poisoning samples
  - Inject them into the original training data
  - Increase the loss of the model trained on the compromised data

EVALUATION

- Setup
  - Datasets
    - Artificial data:
      - Binary classification: Gaussian dist. $[N(-1.5, 0.6^2)$ and $N(1.5, 0.6^2)]$
      - Training data : 50 samples, 25 per class
      - Validation data: 1k samples, 500 per class
    - Real data: MNIST
  - Model(s)
    - SVM [Linear vs. RBF-Kernel]

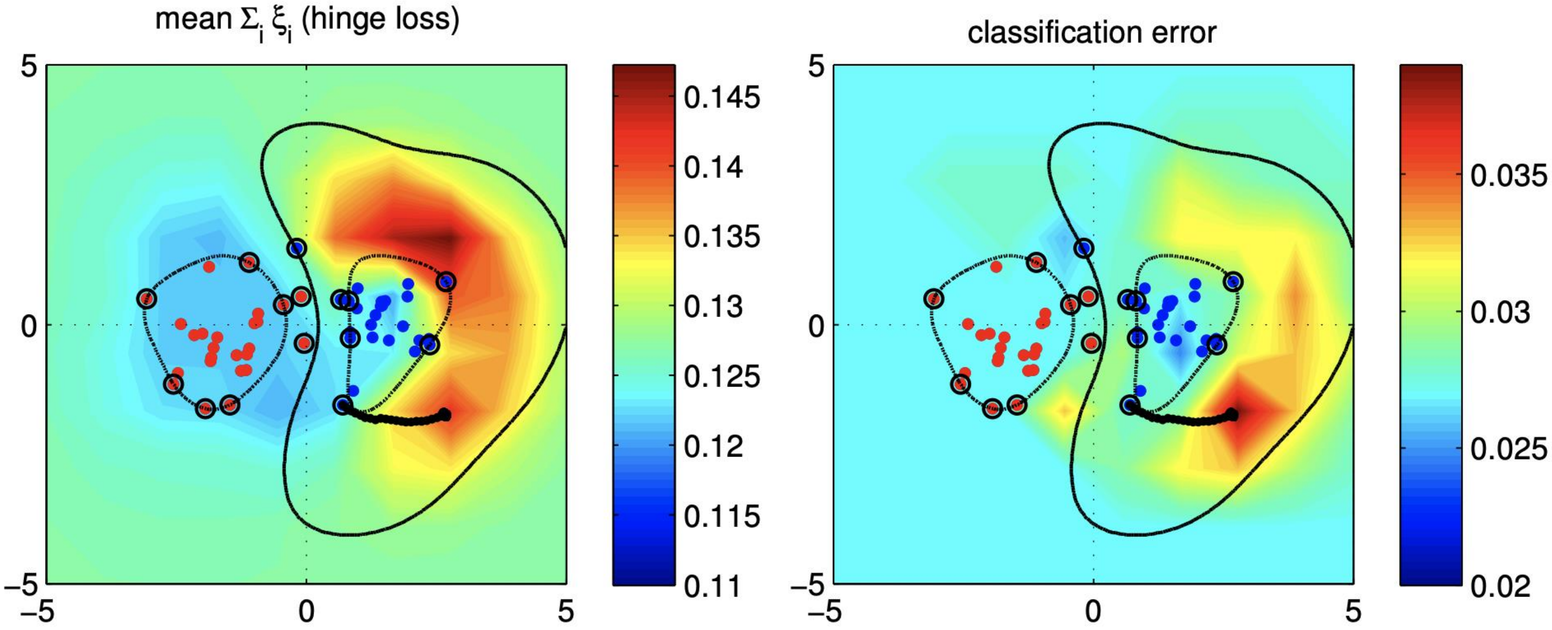
EVALUATION: POISON CRAFTING IN ARTIFICIAL DATA

- Linear SVM



EVALUATION: POISON CRAFTING IN ARTIFICIAL DATA

- SVM with RBF Kernel

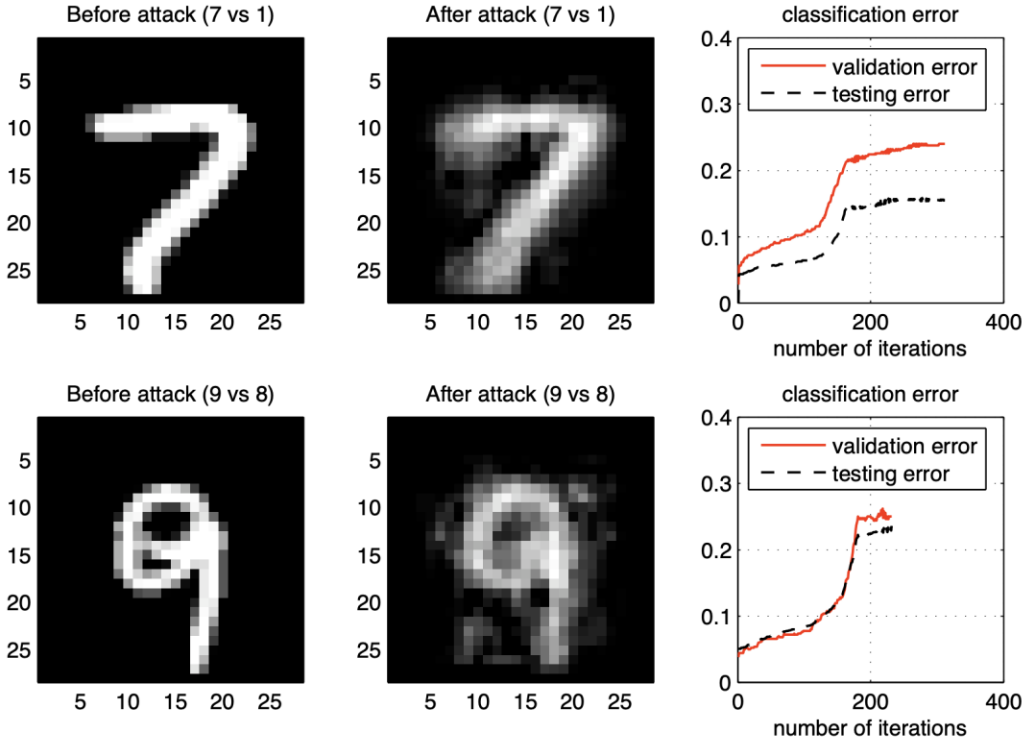


EVALUATION

- Setup
  - Datasets
    - Artificial data:
      - Binary classification: Gaussian dist. [$N(-1.5, 0.6^2)$ and $N(1.5, 0.6^2)$]
      - Training data : 50 samples, 25 per class
      - Validation data: 1k samples, 500 per class
    - Real data: MNIST
      - 7 vs 1 | 9 vs 8 | 4 vs 0
      - Training data : 200 samples, 100 per class
      - Validation data: 1k samples, 500 per class
      - Testing data : 4k samples, 2k per class
  - Model(s)
    - SVM [Linear vs. RBF-Kernel]

EVALUATION: REAL-DATA (MNIST)

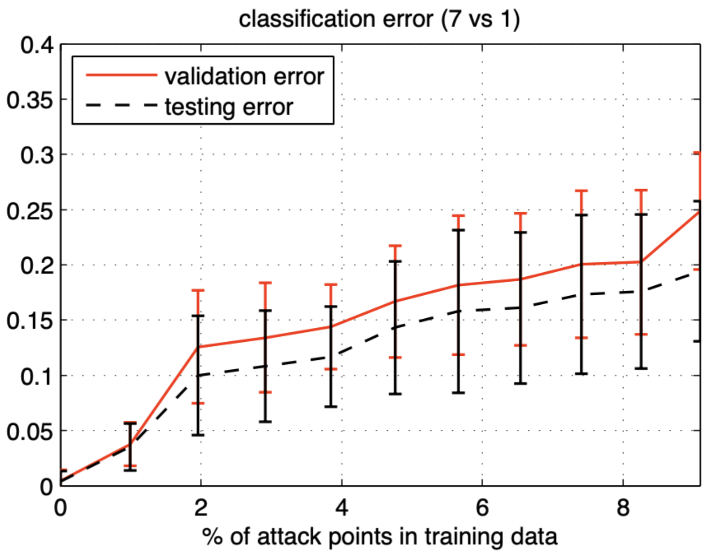
- Linear SVM



- Results
  - Use a *single* poison
  - Error increases by 15 – 20%

EVALUATION: REAL-DATA (MNIST)

- Linear SVM



- Results
  - Use a *single* poison
  - Error increases by 15 – 20%
  - Increasing # poisons leads to a higher error

HOW VULNERABLE REGRESSION MODELS ARE?

MANIPULATING MACHINE LEARNING: POISONING ATTACKS AND COUNTERMEASURES FOR REGRESSION LEANING, JAGIELSKI ET AL., IEEE SECURITY AND PRIVACY SYMPOSIUM 2018

BACKGROUND: REGRESSION MODELS

- Regression Models [Demo]
  - DIT
    - 1. let’s add some more points
    - 2. let’s see how much error (*RMSE*) it increases
  - In the Paper
    - Ordinary Least Squares (OLS)
    - Ridge regression
    - LASSO
    - Elastic-net regression

THREAT MODEL

- Goal
  - Indiscriminate attack (increase the error on D_{val})

- Capability
  - Train a model f on D_{tr}
  - Inject p poisons into the training set ($\mathrm{N}(D_{tr}) = n + p$)

- Knowledge [White-box vs. Black-box]
  - D_{tr} : training data (black-box adversary only has partial knowledge of D_{tr})
  - D_{val}: validation data
  - f: a model and its parameters (black-box attacker doesn't know the parameters)
  - L: training algorithm

POISONING AS A BI-LEVEL OPTIMIZATION

$$\begin{aligned} \arg\max_{\mathcal{D}_p} \quad & \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}_p^\star)\,, \\ \text{s.t.} \quad & \boldsymbol{\theta}_p^\star \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_p, \boldsymbol{\theta}) \end{aligned}$$

- Outer-optimization: maximize the error of a model on the validation data
- Inner-optimization: minimize the model’s error on the training data

PROPOSED POISONING ATTACK ON REGRESSION MODELS

Algorithm 1 Poisoning Attack Algorithm

Input: $\mathcal{D} = \mathcal{D}_{\text{tr}}$ (white-box) or $\mathcal{D}'_{\text{tr}}$ (black-box), $\mathcal{D}'$, $\mathcal{L}$, $\mathcal{W}$, the initial poisoning attack samples $\mathcal{D}_p^{(0)} = (\boldsymbol{x}_c, y_c)_{c=1}^{p}$, a small positive constant ε.

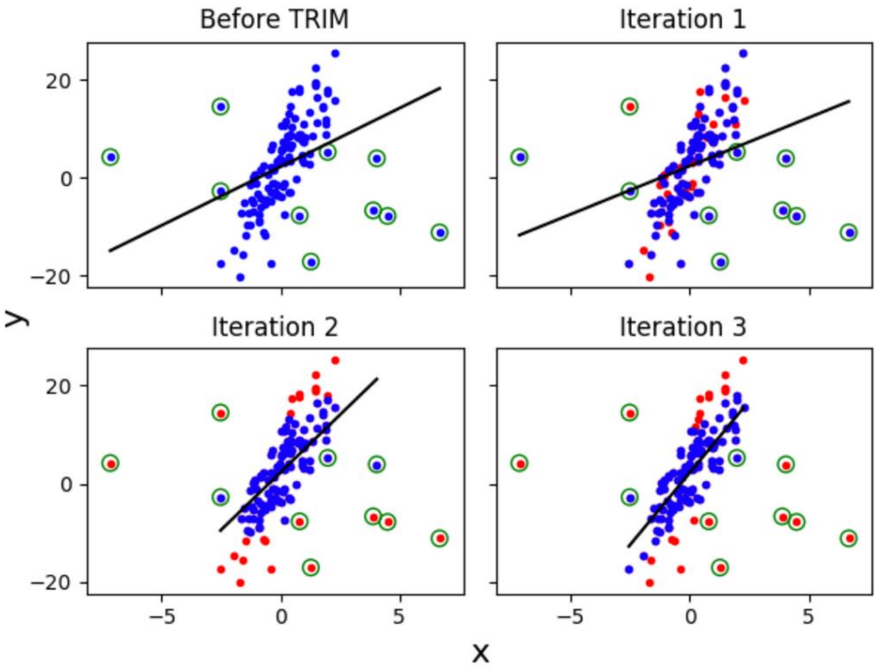
$i \leftarrow 0$ (iteration counter)
$\boldsymbol{\theta}^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i)}, \boldsymbol{\theta})$ — // train a model on the contaminated data
repeat
    $w^{(i)} \leftarrow \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i)})$
    $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)}$
    for c = $1, \ldots, p$ **do**
        $\boldsymbol{x}_c^{(i+1)} \leftarrow \text{line\_search}\left(\boldsymbol{x}_c^{(i)}, \nabla_{\boldsymbol{x}_c} \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i+1)})\right)$ — // update poisons to increase the loss of the model
        $\boldsymbol{\theta}^{(i+1)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{(i+1)}, \boldsymbol{\theta})$
        $w^{(i+1)} \leftarrow \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}^{(i+1)})$
    $i \leftarrow i + 1$
until $|w^{(i)} - w^{(i-1)}| < \varepsilon$ — // stop when the model doesn't change more than *e*

Output: the final poisoning attack samples $\mathcal{D}_p \leftarrow \mathcal{D}_p^{(i)}$

PROPOSED DEFENSE: TRIM

Algorithm 2 [TRIM algorithm]

Input: Training data $\mathcal{D} = \mathcal{D}_{\text{tr}} \cup \mathcal{D}_p$ with $|\mathcal{D}| = N$; number of attack points $p = \alpha \cdot n$.
Output: $\boldsymbol{\theta}$.
$\mathcal{I}^{(0)} \leftarrow \{1, ..., N\}$ /* First train with all samples */
$\boldsymbol{\theta}^{(0)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(0)}}, \boldsymbol{\theta})$ /* Initial estimation of $\boldsymbol{\theta}$*/
$i \leftarrow 0$ /* Iteration count */
repeat
    $i \leftarrow i + 1$;
    $\mathcal{I}^{(i)} \leftarrow$ subset of size n that min. $\mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta}^{(i-1)})$
    $\boldsymbol{\theta}^{(i)} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta})$ /* Current estimator */
    $R^{(i)} = \mathcal{L}(\mathcal{D}^{\mathcal{I}^{(i)}}, \boldsymbol{\theta}^{(i)})$ /* Current loss */
until $i > 1 \wedge R^{(i)} = R^{(i-1)}$ /* Convergence condition*/
return $\boldsymbol{\theta}^{(i)}$ /* Final estimator */.

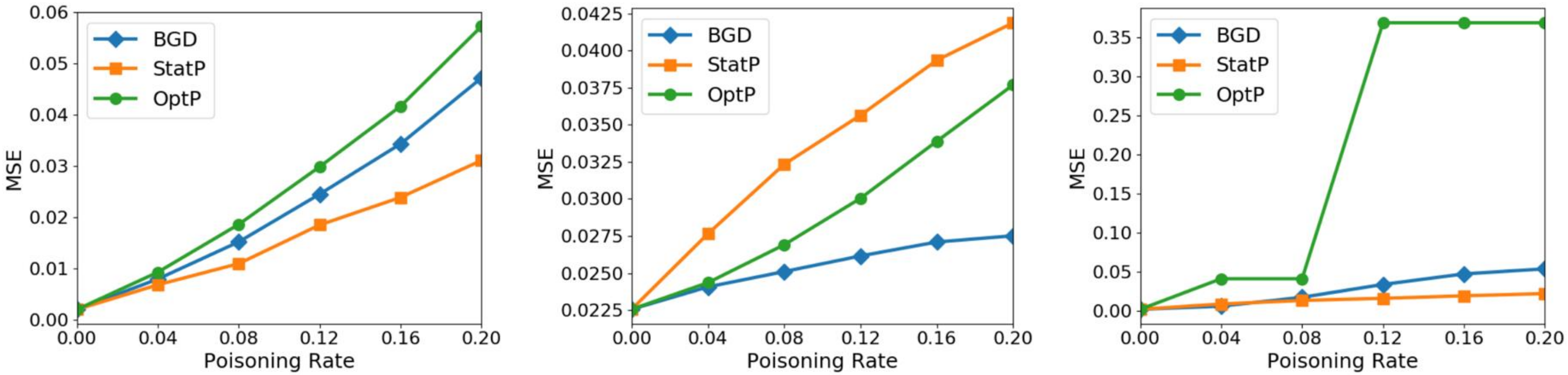


EVALUATION

- Setup
  - Datasets: Health care | Loan | Housing
  - Models
    - Ordinary Least Square (OLS)
    - Ridge regression
    - LASSO
    - Elastic-net regression
  - Attacks
    - OptP | StatP |BGD (Prior work by Xiao *et al.*)
  - Defenses
    - Huber | RANSAC | Chen *et al.* | RONI | TRIM

EVALUATION

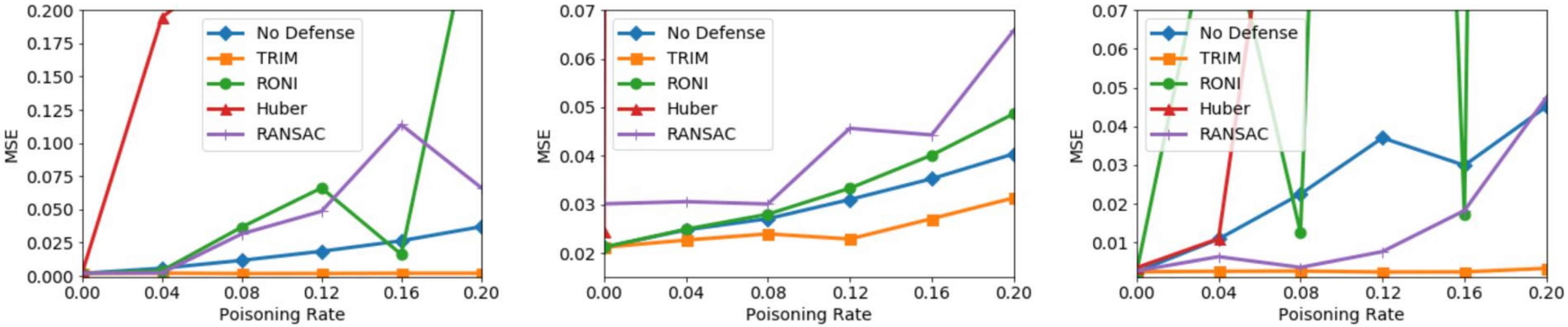
- Results Summary
  - Attacks
    - OptP > StatP, BGD (Prior work)
    - StatP, BGD: varies from datasets
    - StatP > OptP: computational efficiency; StatP still shows a reasonable success rate
    - Poisons *transfer*: crafted on one model works for the three others



(a) Health Care Dataset (b) Loan Dataset (c) House Price Dataset

EVALUATION

- Results Summary
  - Defenses
    - TRIM > Huber | RANSAC | Chen *et al.* | RONI
    - TRIM is computationally efficient (< 0.02 seconds on the House dataset)
    - Prior work's defenses sometimes increase errors



(a) Health Care Dataset (b) Loan Dataset (c) House Price Dataset

AI 539: TRUSTWORTHY ML

TARGETED POISONING ATTACKS

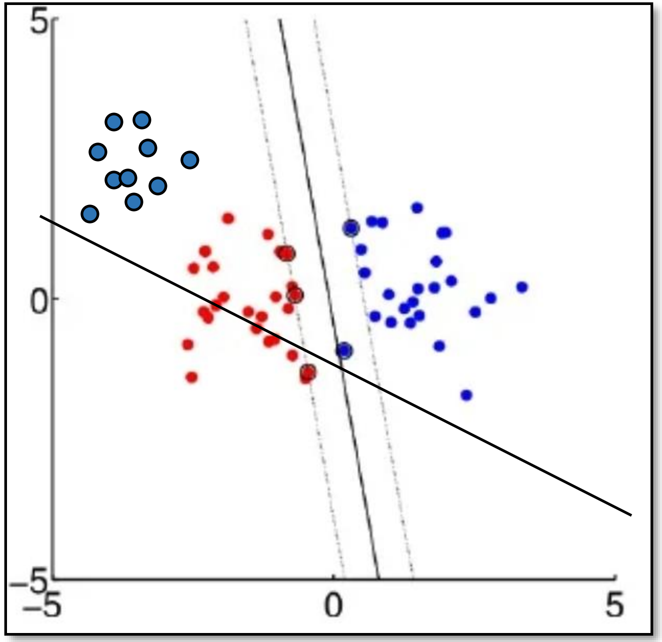
Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

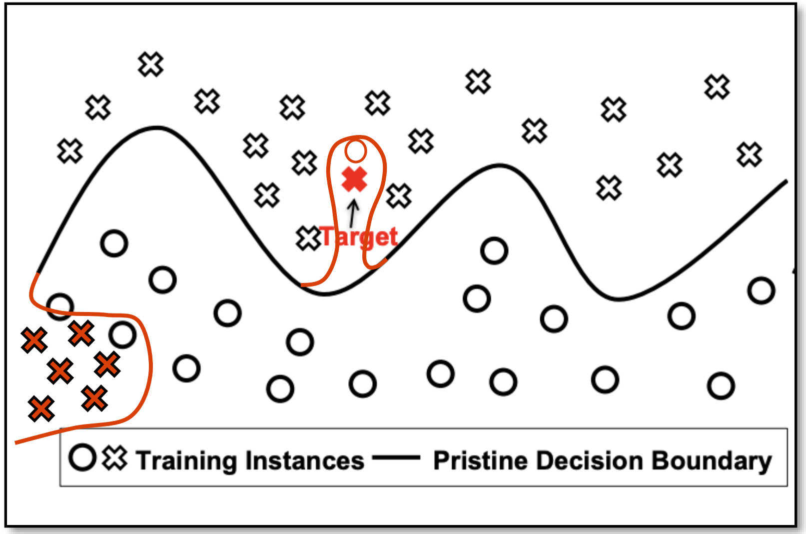
SAIL
Secure AI Systems Lab

RECAP: CONCEPTUAL ILLUSTRATION OF THE VULNERABILITY TO POISONING



← Linear model (SVM)

Neural Network →



Targeted Poisoning Threat Model

- 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 $(x_{c1}, y_{c1}), (x_{c2} \ldots$ and craft poisons $(x_{p1}, y_{p1}), (x_{p2} \ldots$
  - Inject them into the training data
- Knowledge
  - D_{tr} : training data
  - D_{test}: test-set data (validation data)
  - f: a model and its parameters θ
  - A: training algorithm (*e.g.*, mini-batch SGD)

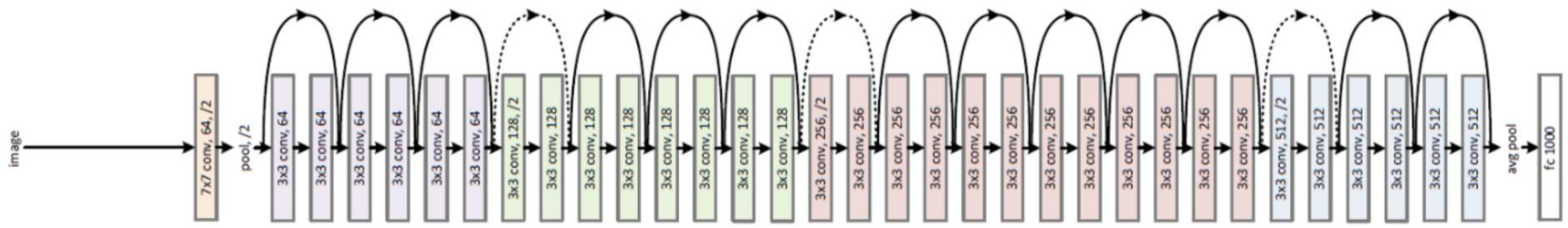
Targeted Poisoning Threat Model

- 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 θ
  - A: training algorithm (*e.g.*, mini-batch SGD)

HOW VULNERABLE NEURAL NETWORKS ARE TO TARGETED ATTACKS?

POISON FROGS! TARGETED CLEAN-LABEL POISONING ATTACKS ON NEURAL NETWORKS, SHAFAHI ET AL., NEURIPS 2018

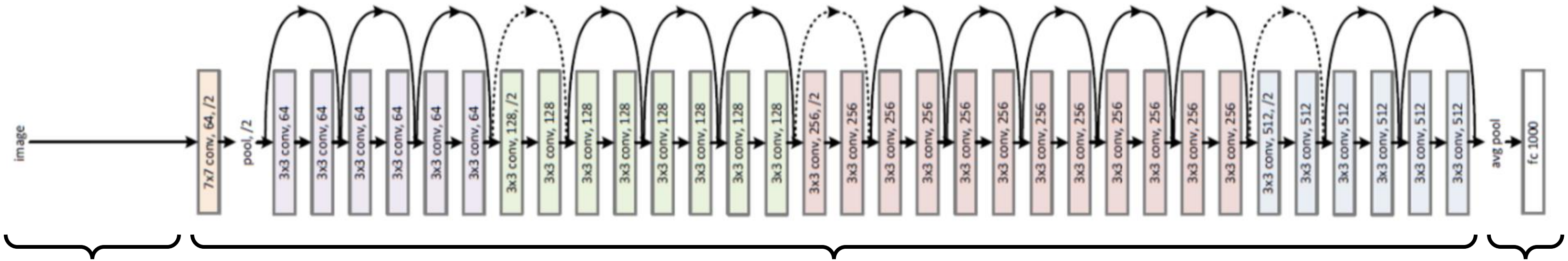
BACKGROUND: CONVOLUTIONAL NEURAL NETWORKS



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

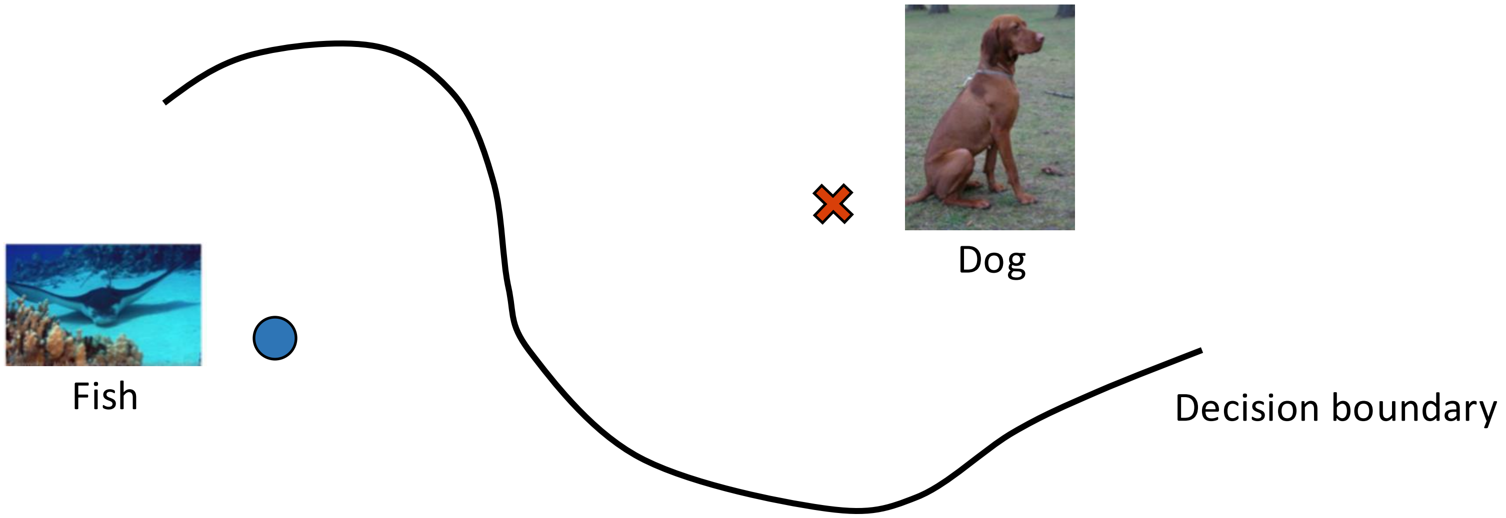


Input x Feature extractor: $f(\cdot)$ Classifier

- 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

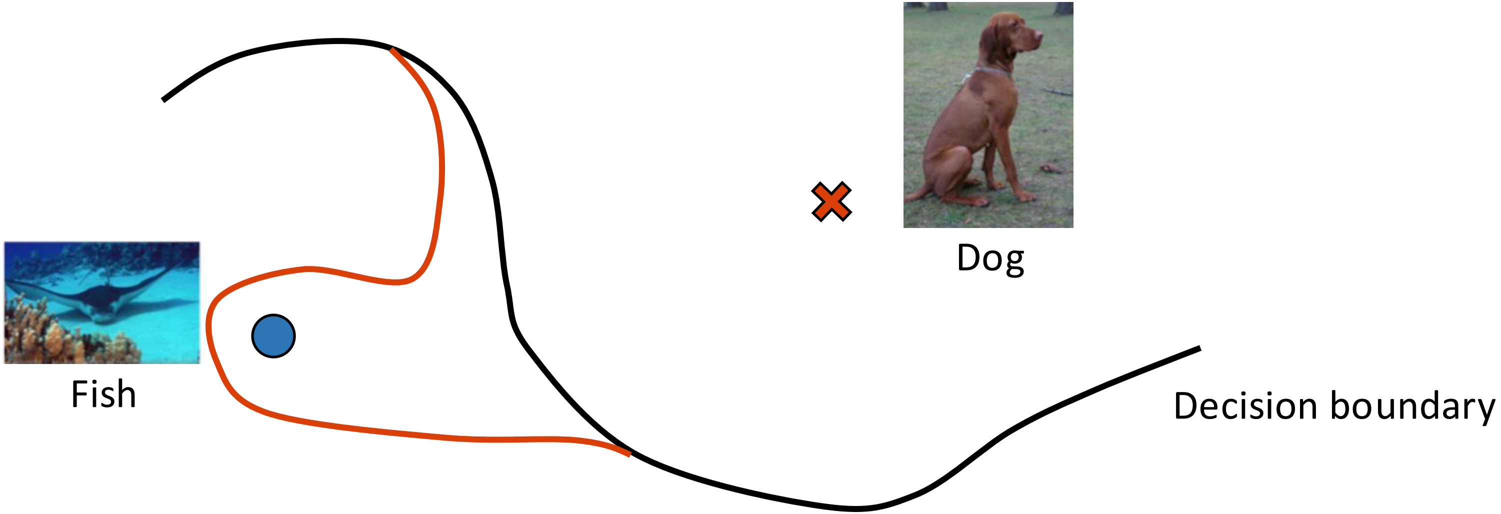
THE KEY IDEA: FEATURE COLLISION

- 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

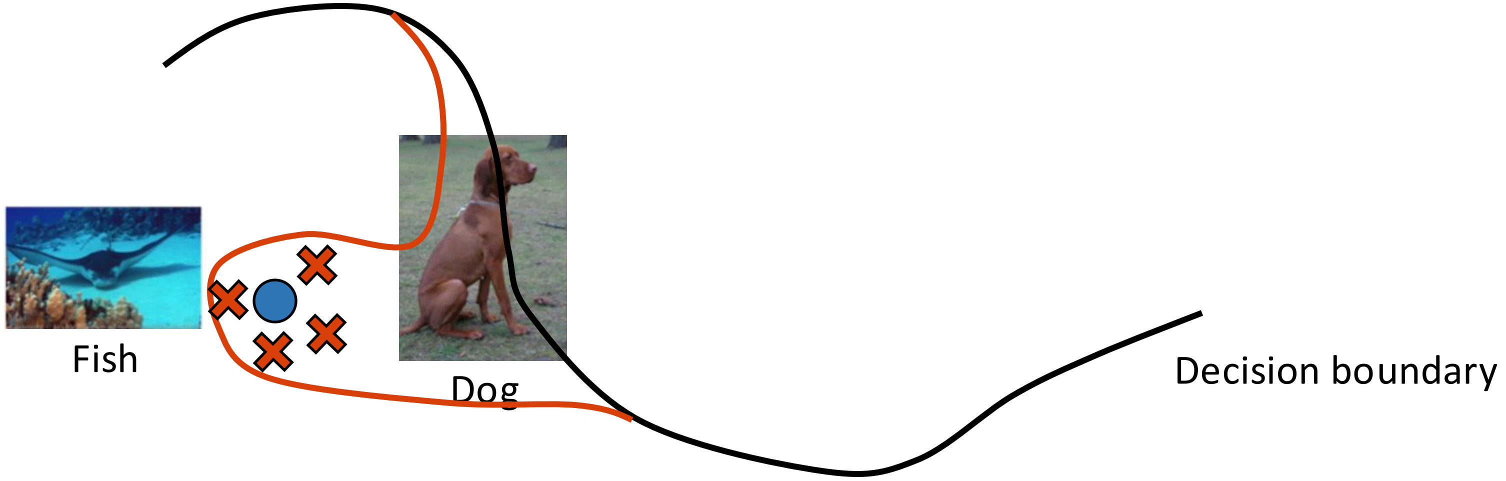
- Goal
  - You want your *any* poison to be closer to your target (x_t, y_t) in the *feature space*



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
  - You want your *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 λ
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

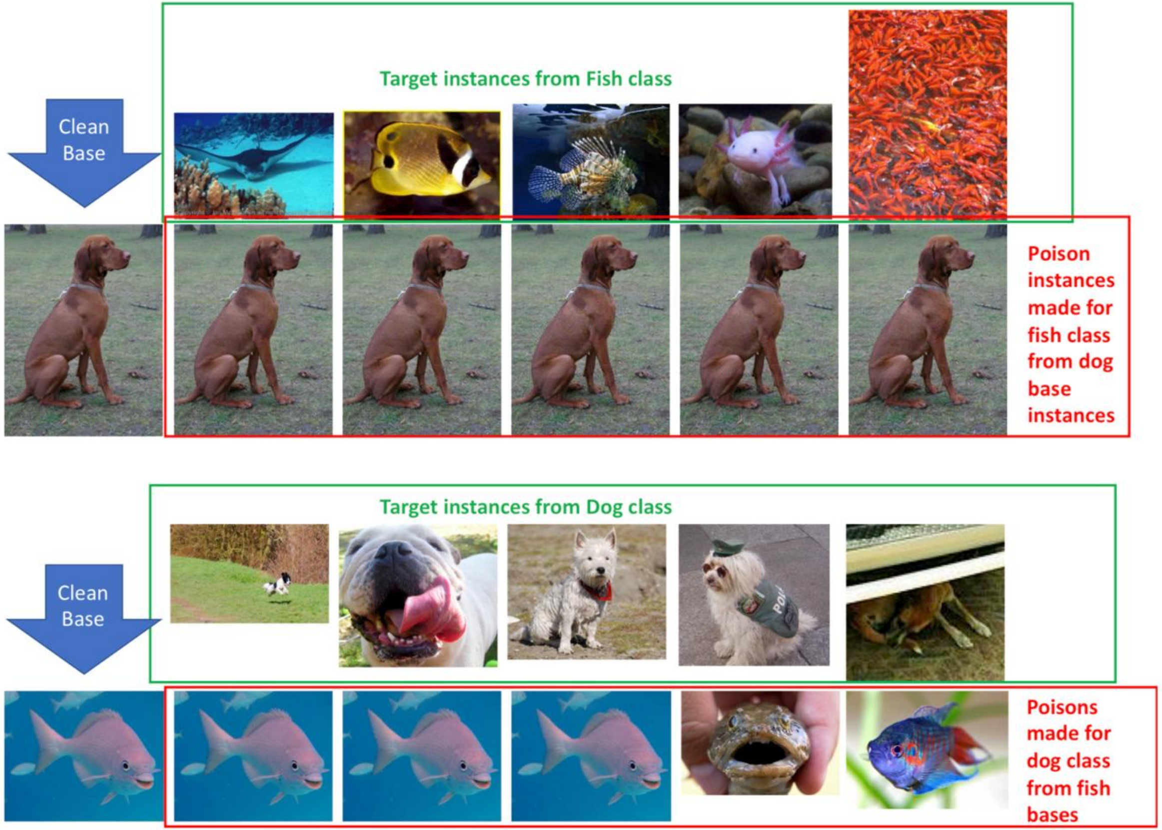
- Setup
  - Dataset: Dog vs. Fish (ImageNet)
  - Models: Inception-V3 (Pretrained on ImageNet)

- “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



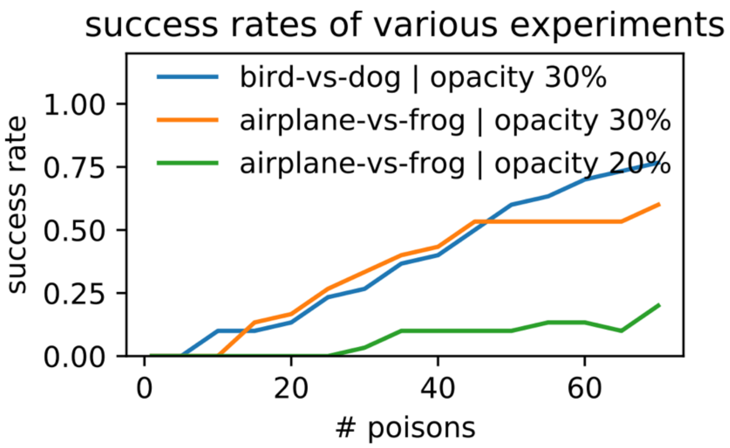
EVALUATIONS: END-TO-END LEARNING

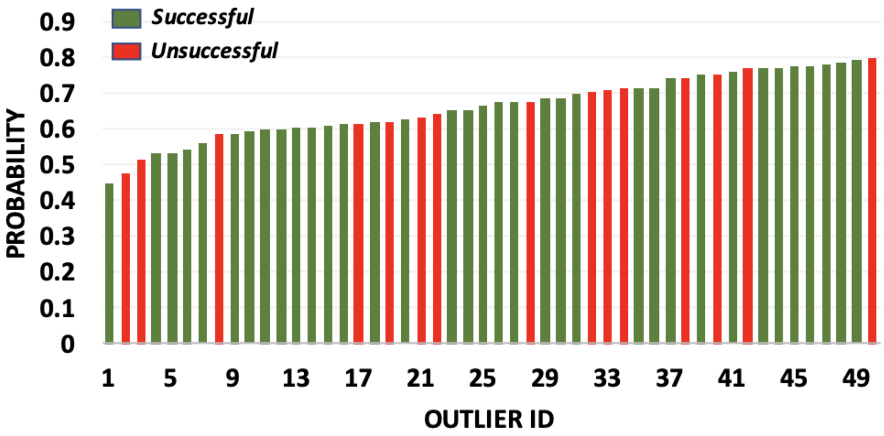
- Setup
  - Dataset: CIFAR-10
  - Models: AlexNet (Pretrained on CIFAR-10)

- “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





Can we improve the transferability of targeted attacks?

MetaPoison! Practical general-purpose clean-label data poisoning, Huang et al., NeuRIPS 2020

Revisit: Poisoning Threat Model

- 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 $(x_{c1}, y_{c1}), (x_{c2} \ldots$ and craft poisons $(x_{p1}, y_{p1}), (x_{p2} \ldots$
  - Inject them into the training data
- Knowledge
  - D_{tr} : training data
  - D_{test}: test-set data (validation data)
  - f: a model and its parameters θ
  - A: training algorithm (*e.g.*, mini-batch SGD)

REVISIT: THE KEY IDEA – FEATURE COLLISION

- Goal
  - Your poisons should work against any f and θ
  - Objective:

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \; \underbrace{\| f(\mathbf{x}) - f(\mathbf{t}) \|_2^2}_{\text{Now you don't know the } f\text{, how can you estimate this?}} + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

- Revisit the previous idea
  - Bi-level optimization

$$\begin{aligned} \arg\max_{\mathcal{D}_p} \quad & \mathcal{W}(\mathcal{D}', \boldsymbol{\theta}_p^\star)\,, \\ \text{s.t.} \quad & \boldsymbol{\theta}_p^\star \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{D}_p, \boldsymbol{\theta}) \end{aligned}$$

$$\underbrace{\begin{aligned} X_p^* &= \underset{X_p}{\operatorname{argmin}} \; \mathcal{L}_{\text{adv}}(x_t, y_{\text{adv}}; \theta^*(X_p)) \\ \theta^*(X_p) &= \underset{\theta}{\operatorname{argmin}} \; \mathcal{L}_{\text{train}}(X_c \cup X_p, Y; \theta) \end{aligned}}_{\text{Problem: no control over } \theta}$$

THE CHALLENGE: LEARNING PROCESS

- Mode parameters are not fixed!
  - Initialization
  - Mini-batch-ed data
  - # of training epochs

Algorithm

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

  Compute gradient

  For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

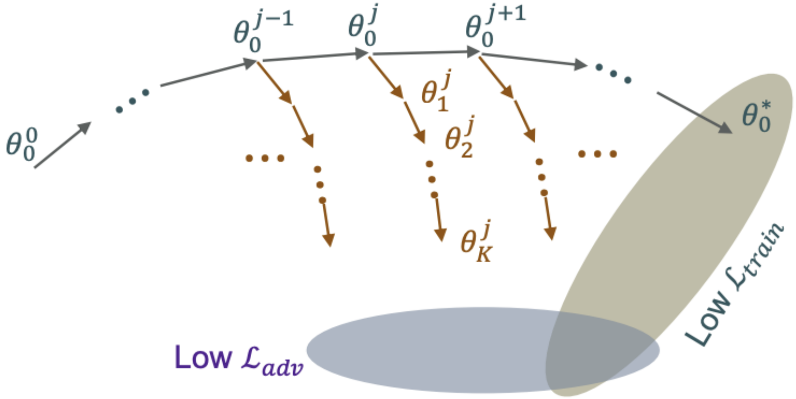
  Descent

  $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

THE KEY IDEA: UNROLLING

- Goal
  - You *simulate all* the training procedures with *possible* $f, \theta s$ while crafting your poisons



Algorithm 1 Craft poison examples via MetaPoison

Input Training set of images and labels (X, Y) of size N, target image x_t, adversarial class y_{adv}, ϵ and ϵ_c thresholds, $n \ll N$ subset of images to be poisoned, T range of training epochs, M randomly initialized models.
Begin
Stagger the M models, training the mth model weights θ_m up to $\lfloor mT/M \rfloor$ epochs
Select n images from the training set to be poisoned, denoted by X_p. Remaining clean images denoted X_c
For $i = 1, \ldots, C$ crafting steps:
  For $m = 1, \ldots, M$ models:
    Copy $\tilde{\theta} = \theta_m$
    For $k = 1, \ldots, K$ unroll steps[a]:
      $\tilde{\theta} = \tilde{\theta} - \alpha \nabla_{\tilde{\theta}} \mathcal{L}_{\text{train}}(X_c \cup X_p, Y; \tilde{\theta})$
    Store adversarial loss $\mathcal{L}_m = \mathcal{L}_{\text{adv}}(x_t, y_{\text{adv}}; \tilde{\theta})$
    Advance epoch $\theta_m = \theta_m - \alpha \nabla_{\theta_m} \mathcal{L}_{\text{train}}(X, Y; \theta_m)$
    If θ_m is at epoch $T + 1$:
      Reset θ_m to epoch 0 and reinitialize
  Average adversarial losses $\mathcal{L}_{\text{adv}} = \sum_{m=1}^{M} \mathcal{L}_m / M$
  Compute $\nabla_{X_p} \mathcal{L}_{\text{adv}}$
  Update X_p using Adam and project onto ϵ, ϵ_c ball
Return X_p

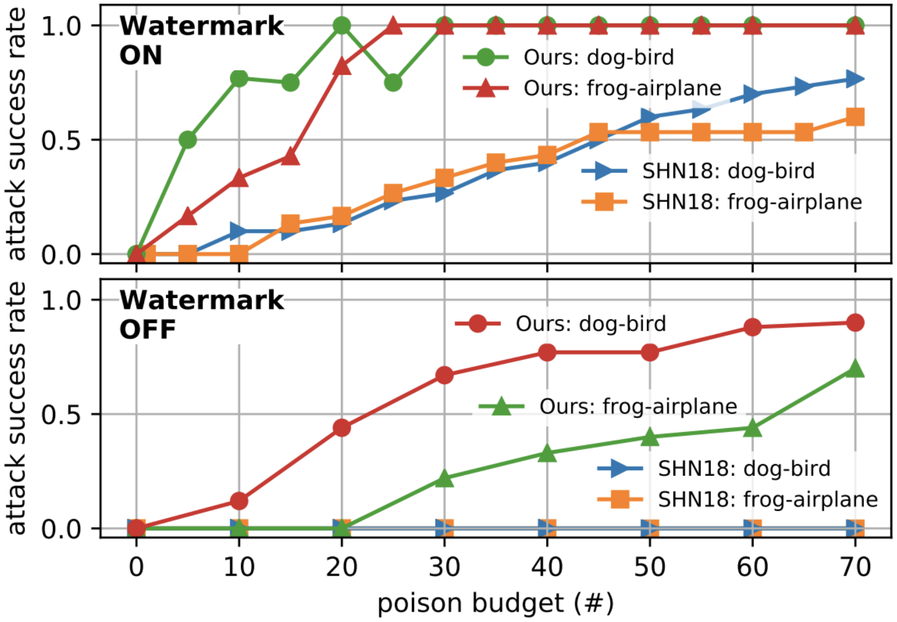
EVALUATION

- Setup
  - Dataset: CIFAR-10
  - Models: 6-layer ConveNet (default), ResNet20, VGG13
  - Attack hyper-parameters:
    - C: 60 | M: 24 | K: 2

- Attacks
  - 30 randomly chosen targets
  - Increase the # poisons from 1 – 10% of the training data n
  - Baseline:
    - Poison Frogs!

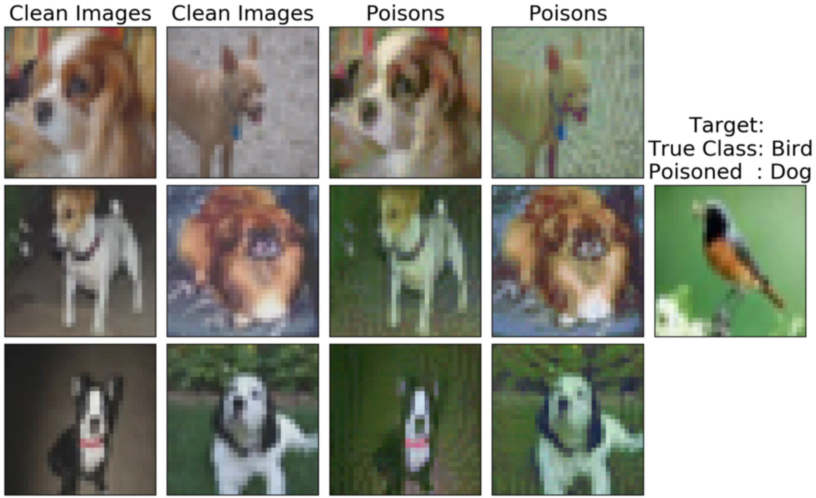
EVALUATION: TRANSFER LEARNING SCENARIO

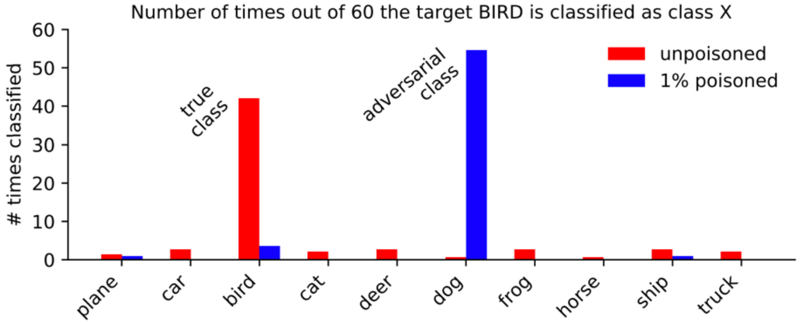
- MetaPoison vs. Poison Frogs

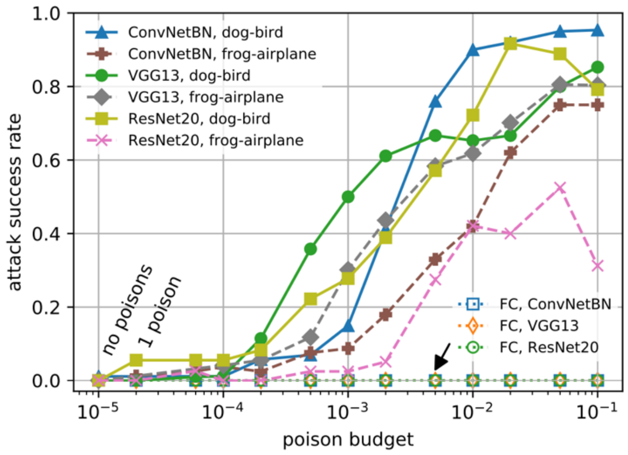


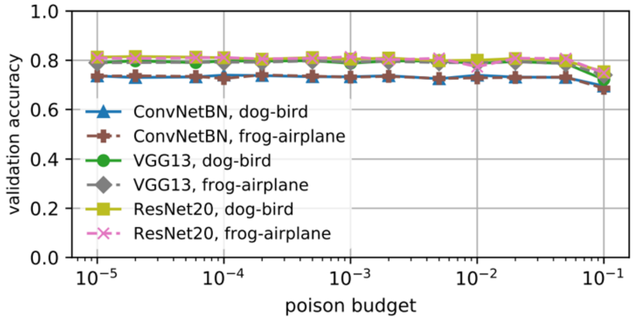
EVALUATION: END-TO-END SCENARIO

- MetaPo



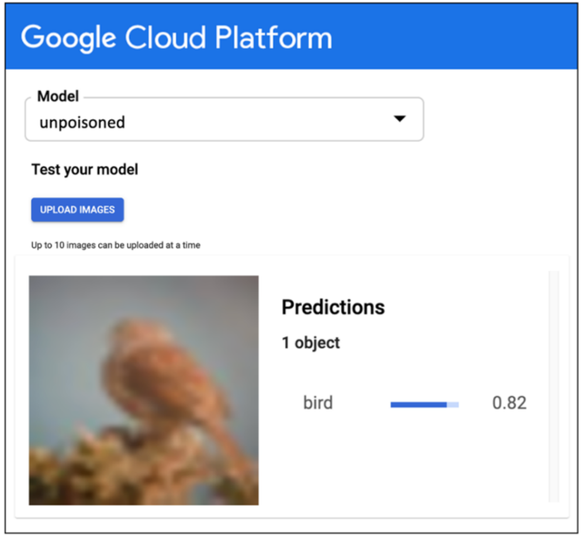


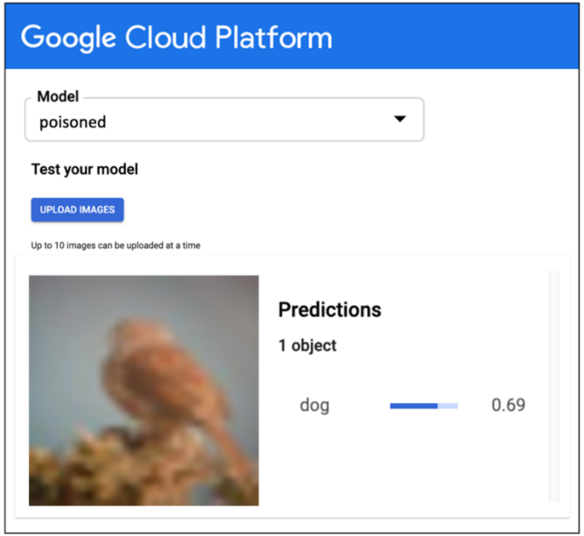


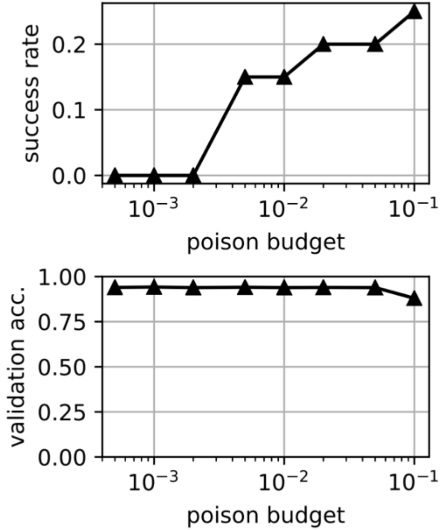


EVALUATION: EXPLOITATION IN REAL-WORLD

- Results







Thank You!

Sanghyun Hong

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

Oregon State University

SAIL
Secure AI Systems Lab