# Notes

- Call for actions
  - In-class presentation sign-ups
  - Term project team-up (by the 10th)

# CS 499/579: Trustworthy ML
# Adversarial attacks: transferability

Tu/Th 4:00 – 5:50 pm

Instructor: **Sanghyun Hong**

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

# Why do adversarial attacks transfer?

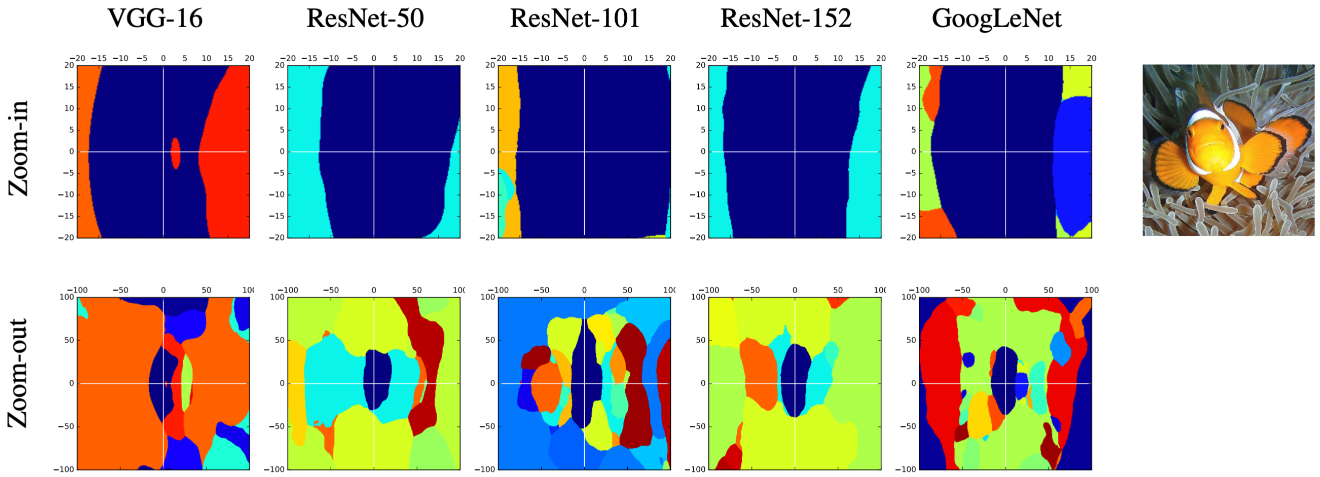
The space of transferable adversarial examples, Tramer et al.
Why do adversarial attacks transfer, Demontis et al., USENIX Security 2019

# Why do adversarial attacks transfer?

- How to answer this question?
  - Inspect a model's decision boundary (Liu et al., Tramer et al.)
  - Inspect the data distribution (Tramer et al.)
  - Comprehensive empirical evaluation (Demotis et al.)
  - …

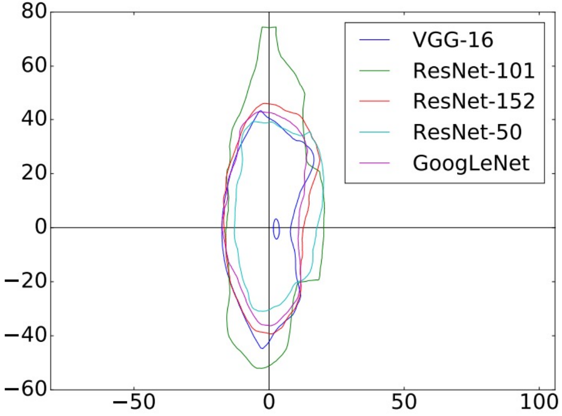
# Why do adversarial examples transfer?

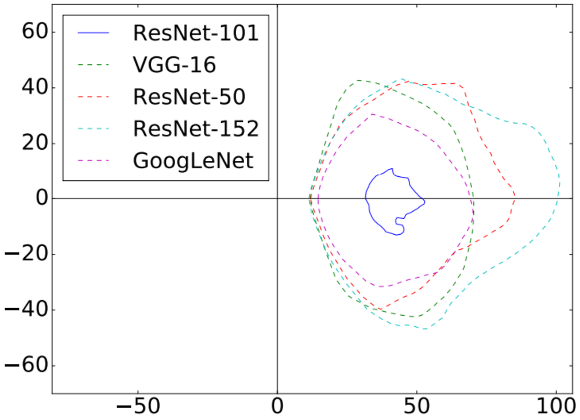
- Recap: Inspect a model's decision boundary
  - Setup:
    - Take a sample image, and two orthogonal gradient directions
    - Perturb the sample along each direction and measure the labels
  - Results



# Why do adversarial examples transfer?

- Recap: Inspect a model's decision boundary: ensemble
  - Setup:
    - Take a sample image, and two orthogonal gradient directions
    - Perturb the sample along each direction and measure the labels
  - Results




# WHY DO ADVERSARIAL EXAMPLES TRANSFER?

- Inspect a model's decision boundary: subspace
  - Setup:
    - Take a sample image, and *multiple* orthogonal gradient directions
    - Perturb the sample along each direction and measure the loss
  - Results

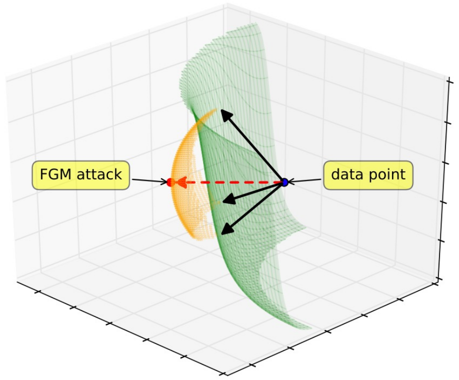



Figure 1: Illustration of the Gradient Aligned Adversarial Subspace (GAAS). The gradient aligned attack (red arrow) crosses the decision boundary. The black arrows are orthogonal vectors aligned with the gradient that span a subspace of potential adversarial inputs (orange).

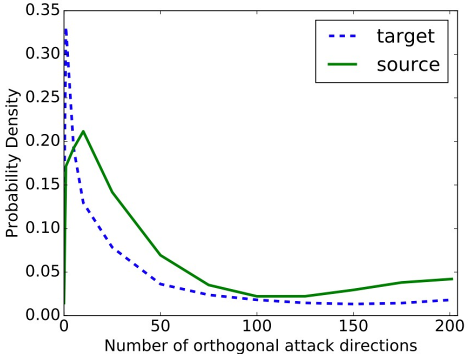



Figure 2: Probability density function of the number of successful orthogonal adversarial perturbations found by the GAAS method on the source DNN model, and of the number of perturbations that transfer to the target DNN model.

# Why do adversarial examples transfer?

- Inspect a model's decision boundary: similarity
  - Setup:
    - Take a sample image, and *three* gradient directions: Legit, Adv., and Rand.
    - Perturb the sample along each direction and measure the distance to the decision boundary and between two boundaries
  - Results

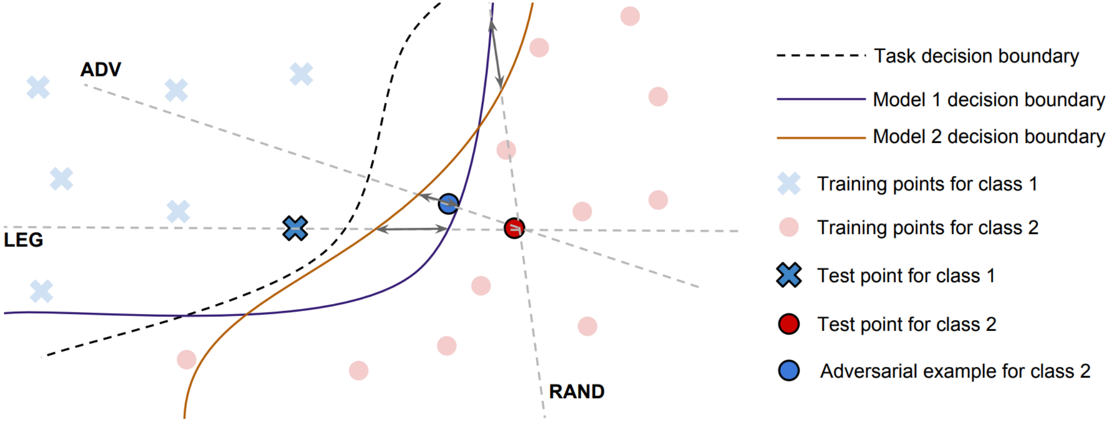



Figure 3: The three directions (Legitimate, Adversarial and Random) used throughout Section 4 to measure the distance between the decision boundaries of two models. The gray double-ended arrows illustrate the *inter-boundary* distance between the two models in each direction.

# Why do adversarial examples transfer?

- Inspect a model's decision boundary: similarity
  - Setup:
    - Take a sample image, and *three* gradient directions: Legit, Adv., and Rand.
    - Perturb the sample along each direction and measure the distance to the decision boundary and between two boundaries
  - Results

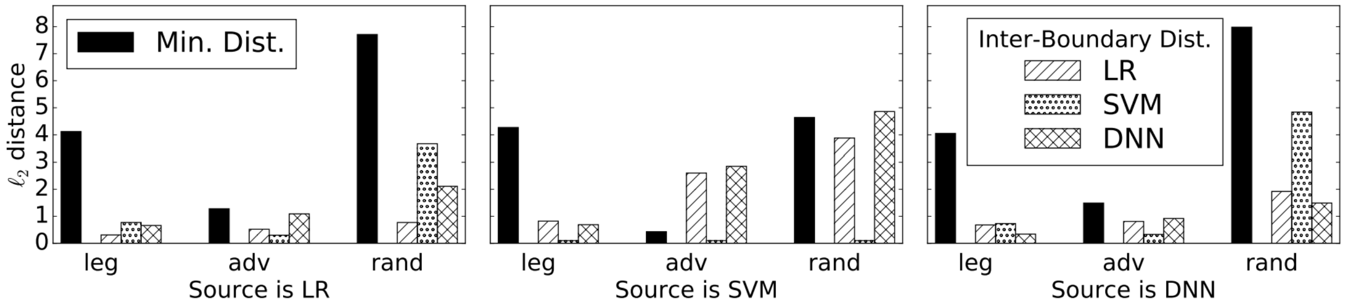



Figure 4: Minimum distances and inter-boundary distances in three directions for MNIST models.

# Why do adversarial examples transfer?

- Ubiquity hypothesis
  - Hypothesis I:
    - Suppose two models achieve low errors and low robustness to adv examples adversarial examples crafted on one model transfer to the other
  - Evaluation I:
    - Train two different models on a task and find adversarial examples do not transfer
    - Results: found, reject
  - Hypothesis II (XOR artifact):
    - Suppose that two models trained on the same set of input features learn representations for which adversarial examples do not transfer to each other; both are non-robust
  - Evaluation II:
    - Adversarial examples crafted one model does not transfer well to the other
    - Results: does not work, reject

# Why do adversarial attacks transfer?

- How to answer this question?
  - Inspect a model's decision boundary (Liu et al., Tramer et al.)
  - Inspect the data distribution (Tramer et al.)
  - Comprehensive empirical evaluation (Demotis et al.)
  - …

# Why do adversarial examples transfer?

- Comprehensive empirical evaluation
  - Setup:
    - A strong adversarial attack
    - Models
      - SVM (linear / rbf)
      - (logistic / ridge) Regression
      - Neural networks
    - Datasets
      - MNIST-89
      - Drebin (android malware)

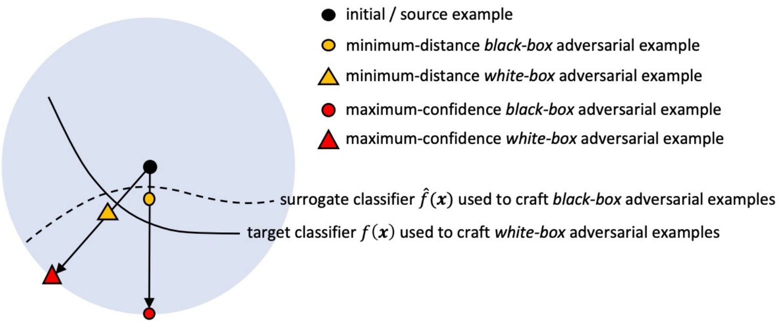



Figure 2: Conceptual representation of maximum-confidence evasion attacks (within an $\ell_2$ ball of radius ε) vs. minimum-distance adversarial examples. Maximum-confidence attacks tend to transfer better as they are misclassified with higher confidence (though requiring more modifications).

# Why do adversarial examples transfer?

- Comprehensive empirical evaluation
  - Setup:
    - Model complexity (= # of parameters) matters
    - Train two models with different complexities
      and measure the success rate of white-box attacks (why?)
  - Results

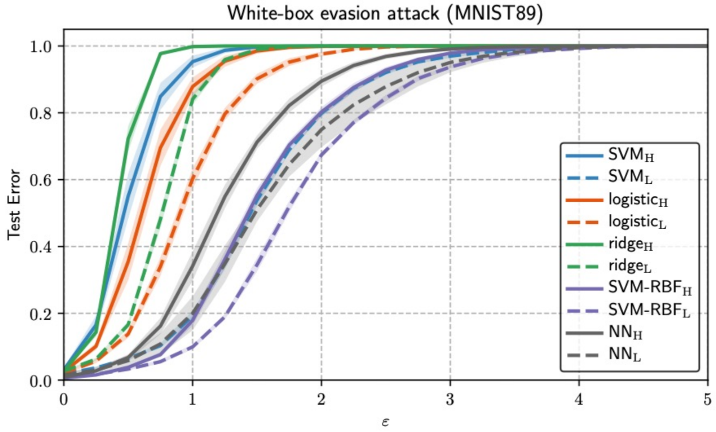



Figure 5: White-box evasion attacks on MNIST89. Test error against increasing maximum perturbation ε.

# Why do adversarial examples transfer?

- Comprehensive empirical evaluation
  - Setup:
    - Model complexity (= # of parameters) matters
    - Train two models with different complexiti and measure the success rate of white-box
    - Run transfer-based attacks between all pai of models and measure the attack success
  - Results
    - Use of low-complexity models as a surrogate increases the adversarial transferability
    - Random forest classifiers are particularly vulnerable to transfer-based attacks

| | $SVM_H$ | $SVM_L$ | $logistic_H$ | $logistic_L$ | $ridge_H$ | $ridge_L$ | $SVM\text{-}RBF_H$ | $SVM\text{-}RBF_L$ | $NN_H$ | $NN_L$ | $RF_H$ | $RF_L$ | transfer rate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| white box | .96 | .19 | .89 | .60 | 1.00 | .83 | .17 | .10 | .31 | .21 | | | |
| $SVM_H$ | .09 | .05 | .08 | .07 | .07 | .06 | .02 | .02 | .03 | .05 | .43 | .45 | .12 |
| $SVM_L$ | .28 | .14 | .26 | .22 | .19 | .17 | .07 | .07 | .13 | .14 | .53 | .54 | .23 |
| $logistic_H$ | .12 | .06 | .11 | .09 | .10 | .09 | .03 | .03 | .04 | .06 | .47 | .49 | .14 |
| $logistic_L$ | .19 | .09 | .18 | .15 | .15 | .13 | .04 | .04 | .08 | .08 | .50 | .52 | .18 |
| $ridge_H$ | .08 | .04 | .07 | .05 | .11 | .07 | .02 | .02 | .03 | .04 | .43 | .45 | .12 |
| $ridge_L$ | .15 | .07 | .13 | .10 | .21 | .15 | .03 | .03 | .05 | .06 | .47 | .49 | .16 |
| $SVM\text{-}RBF_H$ | .19 | .10 | .17 | .15 | .13 | .12 | .06 | .06 | .10 | .11 | .53 | .53 | .19 |
| $SVM\text{-}RBF_L$ | .25 | .13 | .23 | .20 | .17 | .16 | .08 | .08 | .14 | .14 | .53 | .54 | .22 |
| $NN_H$ | .20 | .10 | .18 | .15 | .14 | .12 | .05 | .05 | .11 | .10 | .52 | .53 | .19 |
| $NN_L$ | .24 | .12 | .22 | .20 | .16 | .15 | .07 | .07 | .13 | .13 | .53 | .53 | .21 |

(a) $\varepsilon = 1$

# Why do adversarial examples transfer?

- Comprehensive empirical evaluation
  - Setup:
    - Gradient alignment (= # of parameters) matters
    - Compute the gradient from a surrogate and a target for the same $x$ and measure the cosine similarity metric between the two gradients
  - Results

| | $SVM_H$ | $SVM_L$ | $logistic_H$ | $logistic_L$ | $ridge_H$ | $ridge_L$ | $SVM\text{-}RBF_H$ | $SVM\text{-}RBF_L$ | $NN_H$ | $NN_L$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $SVM_H$ | 0.14 | 0.35 | 0.19 | 0.29 | 0.13 | 0.25 | 0.26 | 0.32 | 0.28 | 0.32 |
| $SVM_L$ | 0.32 | 0.88 | 0.42 | 0.63 | 0.26 | 0.50 | 0.68 | 0.83 | 0.67 | 0.79 |
| $logistic_H$ | 0.18 | 0.45 | 0.25 | 0.37 | 0.18 | 0.32 | 0.35 | 0.42 | 0.36 | 0.41 |
| $logistic_L$ | 0.26 | 0.64 | 0.35 | 0.51 | 0.24 | 0.43 | 0.49 | 0.59 | 0.51 | 0.58 |
| $ridge_H$ | 0.12 | 0.26 | 0.16 | 0.23 | 0.18 | 0.28 | 0.21 | 0.25 | 0.21 | 0.24 |
| $ridge_L$ | 0.22 | 0.49 | 0.29 | 0.41 | 0.27 | 0.47 | 0.39 | 0.46 | 0.40 | 0.44 |
| $SVM\text{-}RBF_H$ | 0.25 | 0.69 | 0.33 | 0.50 | 0.21 | 0.40 | 0.67 | 0.75 | 0.58 | 0.66 |
| $SVM\text{-}RBF_L$ | 0.30 | 0.83 | 0.39 | 0.58 | 0.25 | 0.47 | 0.75 | 0.87 | 0.66 | 0.78 |
| $NN_H$ | 0.26 | 0.68 | 0.34 | 0.51 | 0.22 | 0.41 | 0.57 | 0.67 | 0.65 | 0.68 |
| $NN_L$ | 0.30 | 0.81 | 0.39 | 0.58 | 0.24 | 0.46 | 0.67 | 0.79 | 0.68 | 0.80 |

| | $SVM_H$ | $SVM_L$ | $logistic_H$ | $logistic_L$ | $ridge_H$ | $ridge_L$ | $SVM\text{-}RBF_H$ | $SVM\text{-}RBF_L$ | $NN_H$ | $NN_L$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | .31 | .47 | .35 | .42 | .24 | .37 | .41 | .45 | .39 | .44 |
| | .44 | .89 | .51 | .67 | .34 | .57 | .75 | .86 | .72 | .82 |
| | .34 | .54 | .38 | .47 | .28 | .42 | .46 | .52 | .45 | .50 |
| | .39 | .68 | .46 | .58 | .32 | .51 | .58 | .66 | .57 | .64 |
| | .24 | .34 | .27 | .32 | .35 | .39 | .30 | .33 | .29 | .32 |
| | .35 | .56 | .40 | .50 | .39 | .55 | .49 | .55 | .48 | .52 |
| | .39 | .76 | .45 | .59 | .30 | .50 | .74 | .80 | .66 | .74 |
| | .43 | .86 | .49 | .65 | .32 | .55 | .80 | .90 | .73 | .83 |
| | .37 | .73 | .44 | .57 | .29 | .49 | .65 | .74 | .68 | .72 |
| | .42 | .83 | .49 | .64 | .32 | .53 | .74 | .84 | .72 | .82 |

Figure 8: Gradient alignment and perturbation correlation for evasion attacks on MNIST89. *Left:* Gradient alignment $R$ (Eq. 18) between surrogate (rows) and target (columns) classifiers, averaged on the unmodified test samples. *Right:* Pearson correlation coefficient $\rho(\delta, \hat{\delta})$ between white-box and black-box perturbations for $\varepsilon = 5$.

# Why do adversarial attacks transfer?

- Take aways
  - If the decision boundaries of two models similar, the transferability increases
  - If the transferability is high between two models, there's a common adv. subspace
  - The transferability is non-trivial
    - Two models trained to achieve low-loss and low-resilience to white-box attacks
    - But the adversarial examples do not transfer well between each other
  - XOR artifacts
    - Two models trained with the same set of features, but on disjoint datasets
    - But the adversarial examples do not transfer well between each other
  - If the attacker uses low-complexity models, the transferability becomes high
  - If the two models have aligned gradients, the transferability is high
  - ... (your contributions)

# CS 499/579: Trustworthy ML
# Adversarial attacks: use queries

Tu/Th 4:00 – 5:50 pm

Instructor: **Sanghyun Hong**

sanghyun.hong@oregonstate.edu

Oregon State University

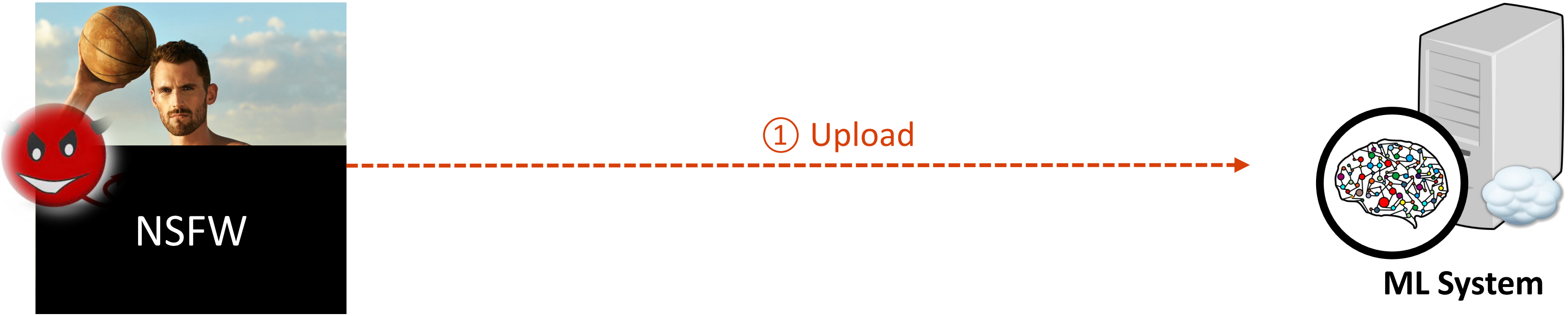
SAIL
Secure AI Systems Lab

# ADVERSARIAL ~~EXAMPLES~~ ATTACKS

- Test-time (evasion) attack
  - Given a test-time sample $x$
  - Craft an adversarial example $x^*$ that fools the target neural network

## Adversarial attacks

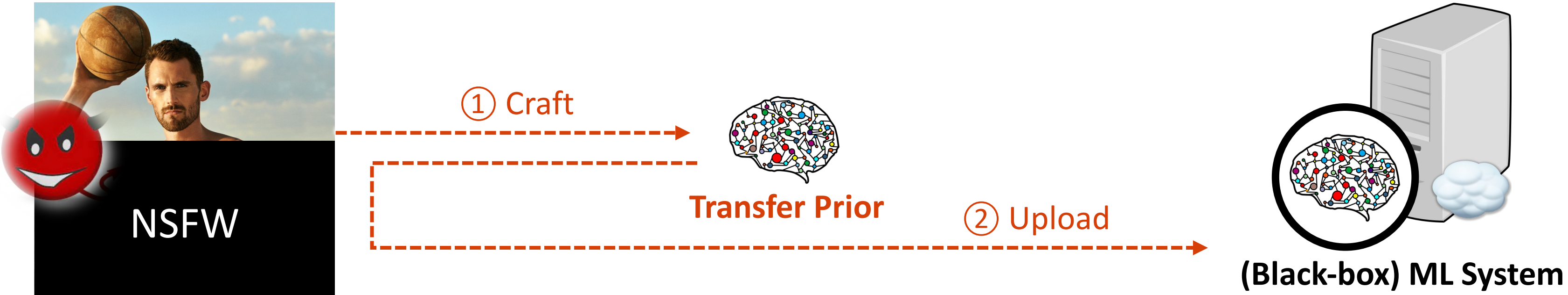
- Example: An adversary wants to upload NSFW image to the cloud




Oregon State University

# (Transfer-based) Black-box Adversarial Attack

- Example: An adversary wants to upload NSFW image to the cloud

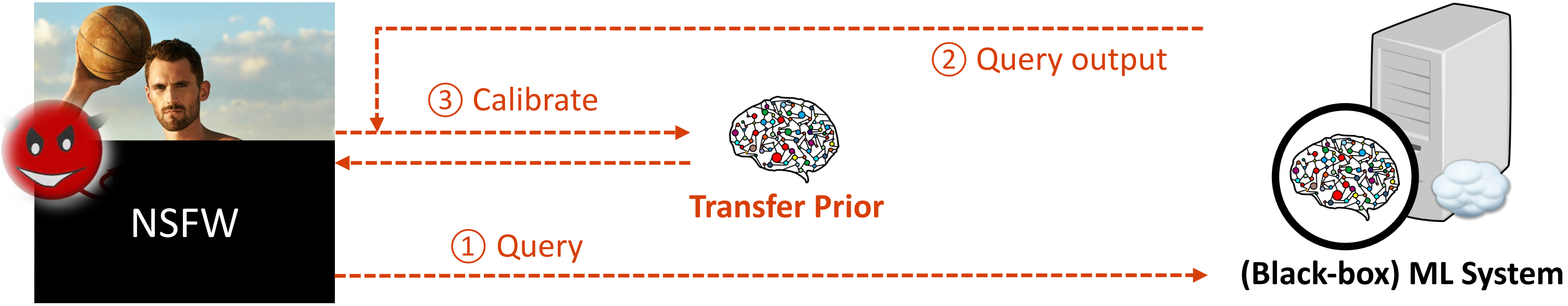



  - **Transfer-based attacks**[1,2] : craft adv. examples on a transfer prior

[1] Goodfellow *et al.*, *Explaining and Harnessing Adversarial Examples*, ICLR 2015
[2] Madry *et al.*, *Towards Deep Learning Models Resistant to Adversarial Attacks*, ICLR 2018

# (Optimization-based) Black-box Adversarial Attack

- Example: An adversary wants to upload NSFW image to the cloud




  - **Transfer-based attacks**[1,2] : craft adv. examples on a transfer prior
  - **Optimization-based attacks**[3] : craft them iteratively with query outputs and a transfer prior

[1] Goodfellow *et al.*, *Explaining and Harnessing Adversarial Examples*, ICLR 2015
[2] Madry *et al.*, *Towards Deep Learning Models Resistant to Adversarial Attacks*, ICLR 2018
[3] Cheng *et al.*, *Improving Black-box Adversarial Attacks with a Transfer-based Prior*, NeurIPS 2019

# Now we talk about optimization-based attacks

Prior convictions: black-box adversarial attacks with bandits and priors, Ilyas et al., ICLR 2019

# Recap: the formulation

- Test-time (evasion) attack
  - **Goal:**
    - Craft human-imperceptible perturbations
      that can make a test-time sample misclassified by a model
  - (Black-box) **Knowledge:**
    - Do not know the model architecture and/or
    - Do not know the trained model's parameters and/or
    - Do not know the training data
  - **Capability:**
    - Sufficient computational power to craft adversarial examples

**How Can An Adversary Launch Attacks on (Black-box) Models?**

# Optimization-based attack

- How can an adversary launch black-box attacks?
  - Brute-force attacks
  - Query-based attacks
  - Transfer attacks

# Optimization-based attack

- Research questions
  - How can we make the optimization-based attacks more successful?
  - How effective (and successful) is this new method?

# Revisit: the formulation

- Suppose:
  - $(x, y)$: a test-time sample; $x \in R^d$ and $y \in [k]$; $x \in [0, 1]$
  - $f$: a neural network; $\theta$: its parameters
  - $L(\theta, x, y)$: a loss function

- Goal (of the first order attacker):
  - Find an $x^{adv} = x + \delta$ such that $\max_{\delta \in S} L(\theta, x^{adv}, y)$ while $||\delta||_p \leq \varepsilon$

- PGD Crafts:

$$\boxed{x^{t+1} = \Pi_{x+S}\left(x^t + \alpha \operatorname{sgn}(\underbrace{\nabla_x L(\theta, x, y)}_{\text{We Need to Know This!}})\right).}$$

# Optimization-based attack is the gradient estimation problem

- Zeroth-order Optimization
  - Finite Difference Method (FDM):

$$D_v f(x) = \langle \nabla_x f(x), v \rangle \approx (f(x + \delta v) - f(x)) / \delta.$$

    - Compute: derivative of a function $f$ at a point $x$ towards a vector $v$

  - FDM for the gradient with $d$-components:

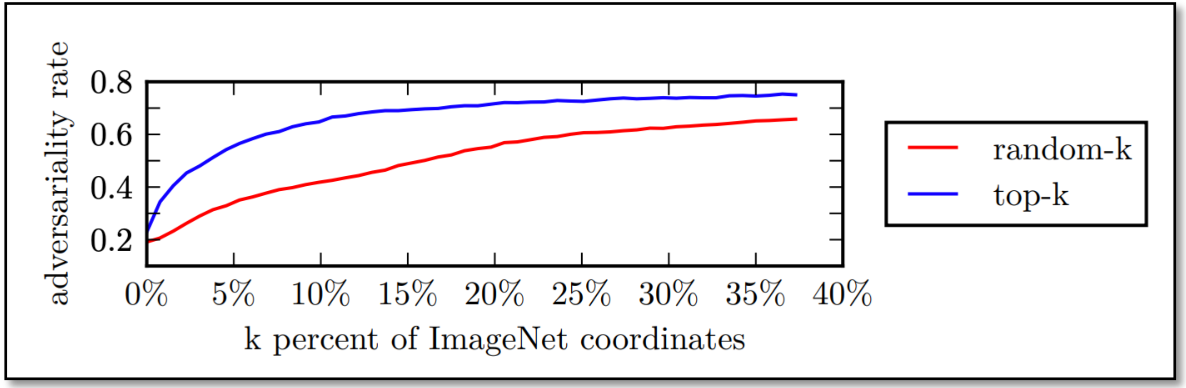
$$\widehat{\nabla}_x L(x, y) = \sum_{k=1}^{d} e_k (L(x + \delta e_k, y) - L(x, y)) / \delta \approx \sum_{k=1}^{d} e_k \langle \nabla_x L(x, y), e_k \rangle$$

- PGD in the black-box cases:

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

# Optimization-based attack is the gradient estimation problem

- Toy experiment
  - Setup
    - Compare the fraction of correctly estimated coordinates of gradients required
    - Compare top-k perturbations picked by magnitude or randomly
    - Measure the transfer-attack success rate
  - Results:
    - Adversarial attacks are effective even with the imperfect gradient estimate
    - Perturbations picked by magnitude is much effective than the random perturbations



# Optimization-based attack is the gradient estimation problem

- Prior approaches to do this estimation
  - The Least Squares Method: $\min_{\widehat{g}} \|\widehat{g}\|_2 \quad \text{s.t. } A\widehat{g} = y.$
  - Iteratively compute the estimate $\hat{g}$, where:
    - $A$: Queries {1, 2, …}
    - $y$: the corresponding inner product values
  - Natural Evolution Strategy [Ilyas *et al.*] and Least Squares equivalence

$$\langle \hat{x}_{LSQ}, \boldsymbol{g} \rangle - \langle \hat{x}_{NES}, \boldsymbol{g} \rangle \leq O\left(\sqrt{\frac{k}{d} \cdot \log^3\left(\frac{k}{p}\right)}\right) ||g||^2$$

# Optimization-based attack is the gradient estimation problem

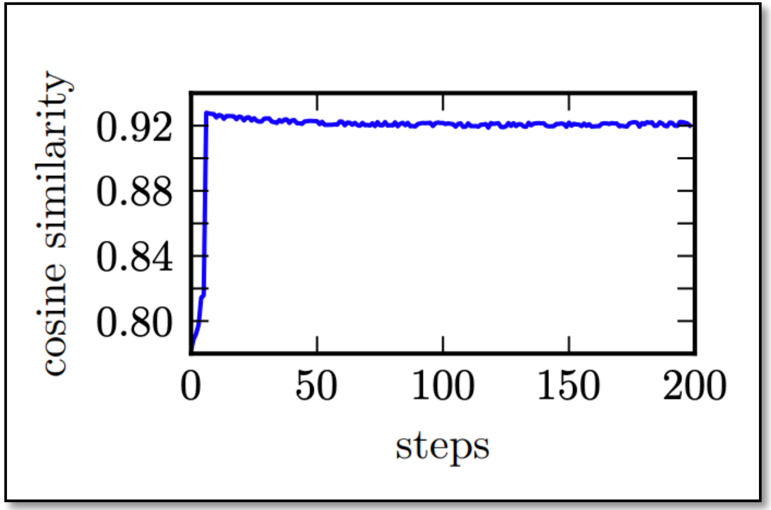
- Prior (= knowledge an adversary can acquire)
  - Gradients are correlated in successive attack iterations
  - Pixels close to each other tend to have similar values

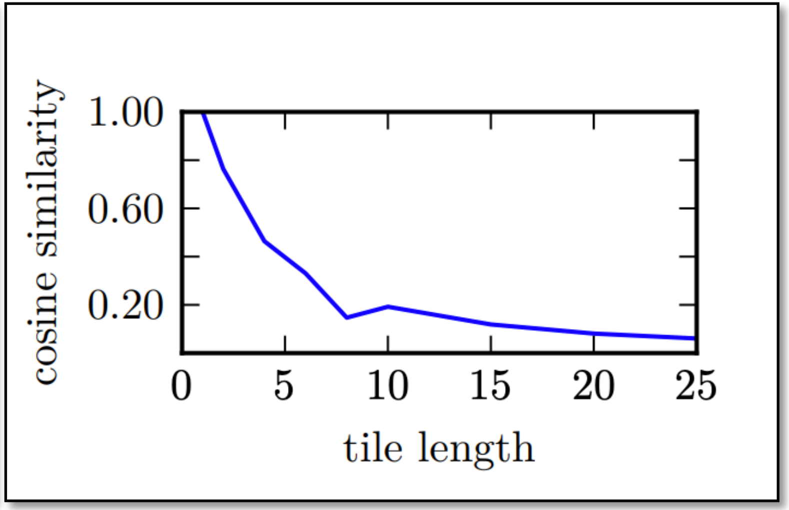
# Optimization-based attack is the gradient estimation problem

- Prior (= knowledge an adversary can acquire)
  - [Time-dependent] Gradients are correlated in successive attack iterations
  - [Data-dependent] Pixels close to each other tend to have similar values

# Optimization-based attack is the gradient estimation problem

- Time-dependent & Data-dependent Priors



# Putting all together

- Formulate the Problem to the Bandit Framework
  - **Bandit problem**

---

**Algorithm 1** Gradient Estimation with Bandit Optimization

---

**procedure** BANDIT-OPT-LOSS-GRAD-EST$(x, y_{init})$

    $v_0 \leftarrow \mathcal{A}(\phi)$

    **for** each round $t = 1, \ldots, T$ **do**

        // Our loss in round $t$ is $\ell_t(g_t) = -\langle \nabla_x L(x, y_{init}), g_t \rangle$

        $g_t \leftarrow v_{t-1}$

        $\Delta_t \leftarrow$ GRAD-EST$(x, y_{init}, v_{t-1})$ // Estimated Gradient of $\ell_t$

        $v_t \leftarrow \mathcal{A}(v_{t-1}, \Delta_t)$

    $g \leftarrow v_T$

    **return** $\Pi_{\partial \mathcal{K}}[g]$

---

# Putting all together

- Formulate the Problem to the Bandit Framework
  - Gradient Estimation

**Algorithm 2** Single-query spherical estimate of $\nabla_v \langle \nabla L(x, y), v\rangle$

**procedure** GRAD-EST$(x, y, v)$

$\boxed{u \leftarrow \mathcal{N}(0, \frac{1}{d} I) \text{ // Query vector}}$

$\{q_1, q_2\} \leftarrow \{v + \delta \boldsymbol{u}, v - \delta \boldsymbol{u}\}$ // Antithetic samples

$\ell_t(q_1) = -\langle \nabla L(x, y), q_1\rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_1, y)}{\epsilon}$ // Gradient estimation loss at $q_1$

$\ell_t(q_2) = -\langle \nabla L(x, y), q_2\rangle \approx \frac{L(x,y) - L(x+\epsilon \cdot q_2, y)}{\epsilon}$ // Gradient estimation loss at $q_2$

$\boldsymbol{\Delta} \leftarrow \frac{\ell_t(q_1) - \ell_t(q_2)}{\delta} \boldsymbol{u} = \frac{L(x+\epsilon q_2, y) - L(x+\epsilon q_1, y)}{\delta \epsilon} \boldsymbol{u}$

// Note that due to cancellations we can actually evaluate $\boldsymbol{\Delta}$ with only two queries to $L$

**return** $\boldsymbol{\Delta}$

# Putting all together

- Formulate the Problem to the Bandit Framework
  - Gradient Estimation

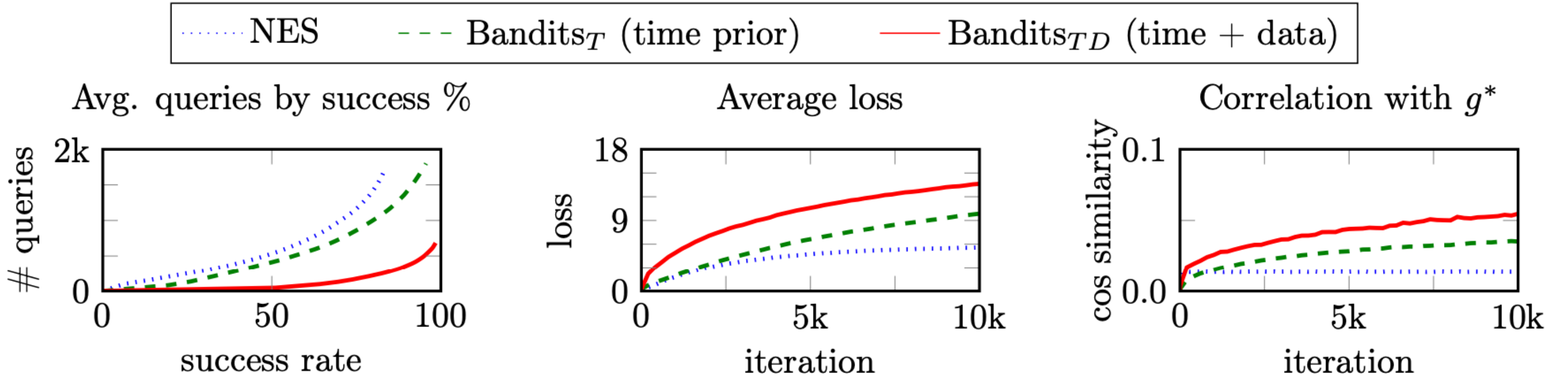
**Algorithm 3** Adversarial Example Generation with Bandit Optimization for $\ell_2$ norm perturbations

1: **procedure** ADVERSARIAL-BANDIT-L2($x_{init}, y_{init}$)
2:     // $C(\cdot)$ returns top class
3:     $v_0 \leftarrow \mathbf{0}_{1\times d}$ // If data prior, $d < \dim(x)$; $v_t$ ($\Delta_t$) up (down)-sampled before (after) line 8
4:     $x_0 \leftarrow x_{init}$ // Adversarial image to be constructed
5:     **while** $C(x) = y_{init}$ **do**
6:         $g_t \leftarrow v_{t-1}$
7:         $x_t \leftarrow x_{t-1} + h \cdot \frac{g_t}{||g_t||_2}$ // Boundary projection $\frac{g}{||g_t||}$ standard PGD: c.f. [Rig15]
8:         $\Delta_t \leftarrow$ GRAD-EST($x_{t-1}, y_{init}, v_{t-1}$) // Estimated Gradient of $\ell_t$
9:         $v_t \leftarrow v_{t-1} + \eta \cdot \Delta_t$
10:         $t \leftarrow t + 1$
    **return** $x_{t-1}$

# How effective is this new attack (= method)?

- Setup
  - Dataset: ImageNet (10k randomly chosen samples)
  - Model: Inception-v3
  - Baseline: NES

- Results



# Optimization-based attack

- Take aways
  - How **accurate** should we estimate a gradient for successful attacks?
    - PGD can be quite successful with imperfect gradient estimates
    - Query-efficiency is bounded by the prior work [Ilyas *et al.*] in practical scenarios
  - How can we estimate gradient accurately with **smaller queries**?
    - Use two priors: time- and data-dependent priors
    - Formulate the estimation into the bandit framework
  - How **effective (and successful)** is this new method?
    - Require 2.5 – 5x less queries for successful attacks compared to NES

# Thank You!

Tu/Th 10:00 – 11:50 am (Recorded lecture)

Instructor: **Sanghyun Hong**

https://secure-ai.systems/courses/MLSec/Sp23

Oregon State University

SAIL
Secure AI Systems Lab