# Securing AI Models: Strategies to Prevent Stealing Attacks

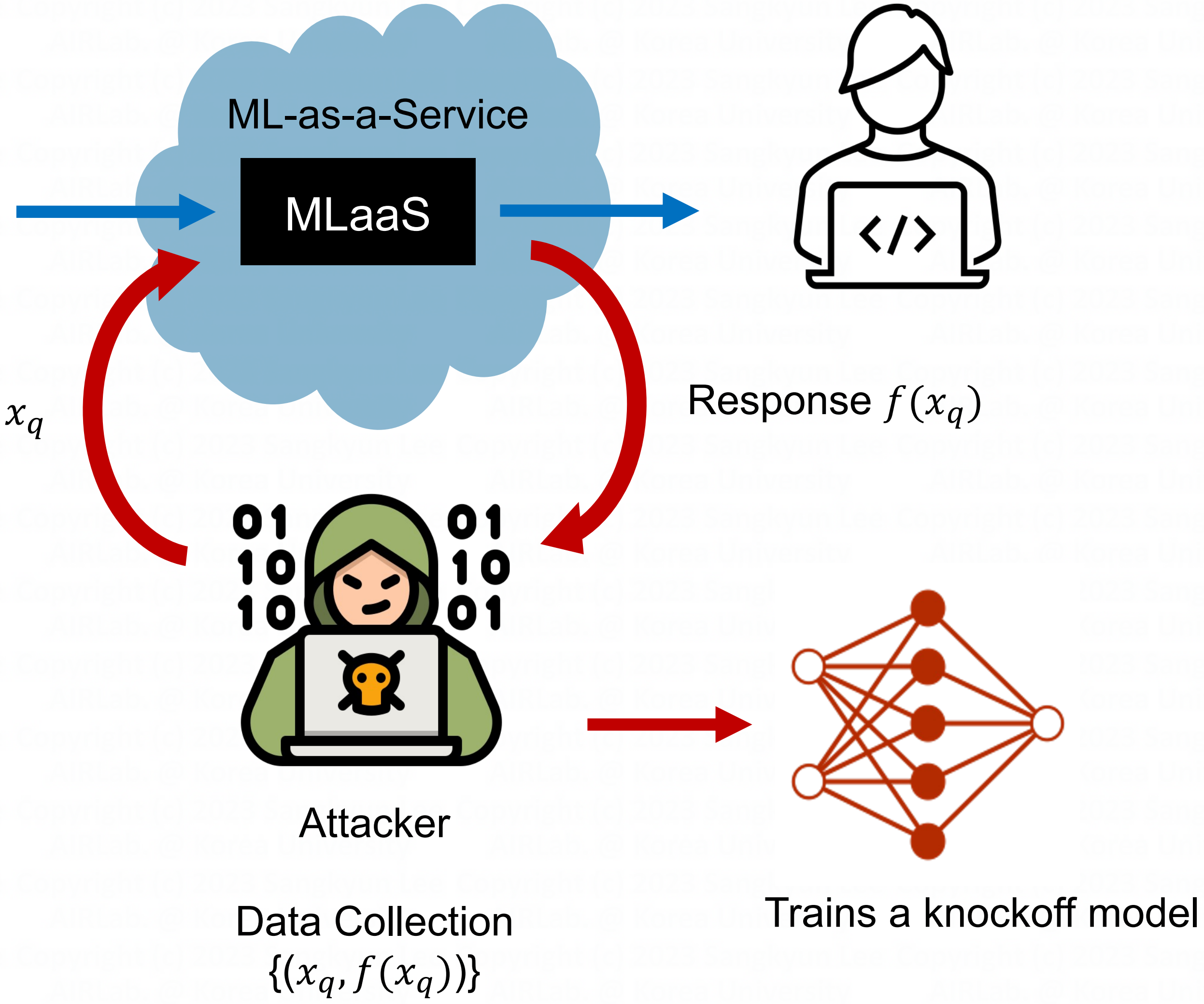
AI Research (AIR) Lab.
School of Cybersecurity
Korea University

Prof. Sangkyun Lee
(sangkyun@korea.ac.kr)

86th IFIP WG10.4 Meeting
July 28, 2024 (Gold Coast, Australia)

# AI Model Stealing Attacks

# Query-Based Model Stealing Attack



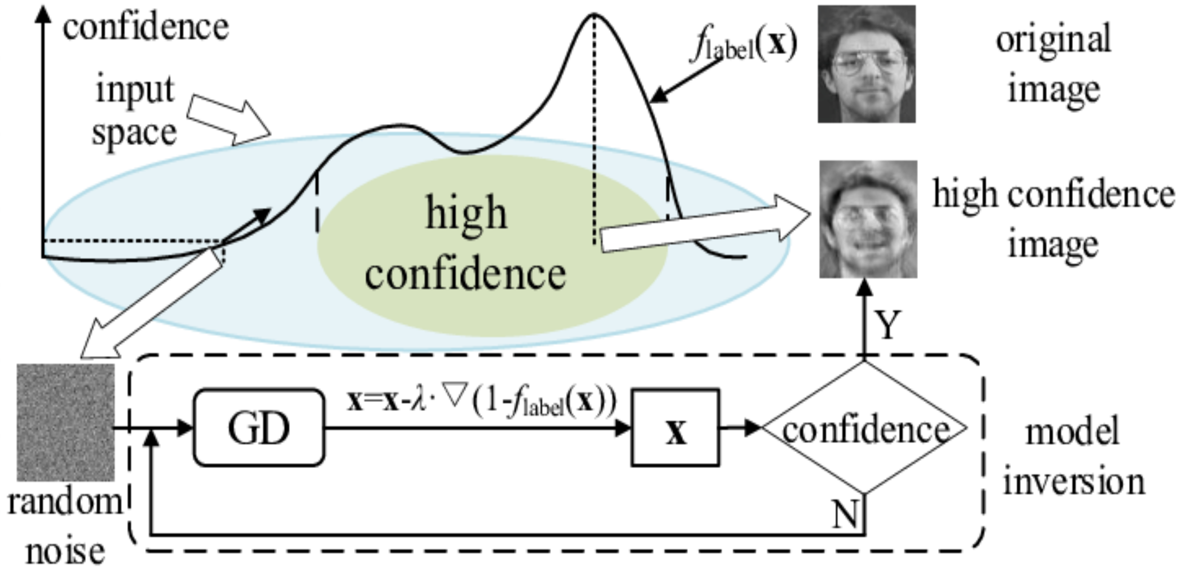

**Basic Idea:**

- An attacker sends his/her queries (like benign users) and collects the server's responses
- The attacker trains a knockoff model using the collected data

# Attack Scenarios

1. Avoiding query charges in future

2. A stepping stone for model inversion attack
   - Stolen models could leak information about sensitive training data, violating data privacy
   - [Fredrikson+, Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, CCS 2015]
   - [Song+, Machine Learning Models that Remember Too Much, CCS 2017]
   - [Liu+, Unstoppable Attack: Label-Only Model Inversion via Conditional Diffusion Model, CCS 2023]




https://www.researchgate.net/figure/The-Framework-of-Model-Inversion-Attack_fig3_344378202

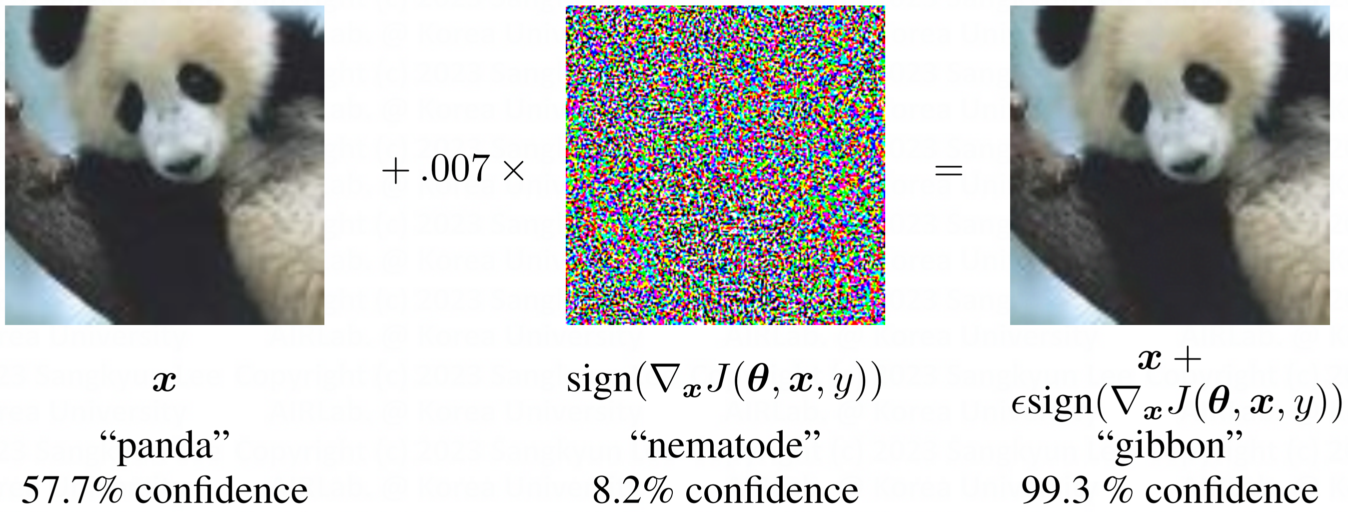
# Model-Stealing Attack Scenarios

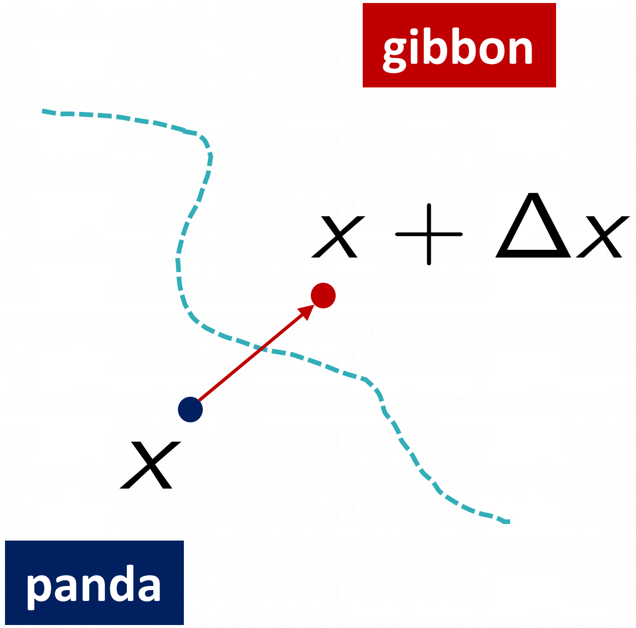
3. A stepping stone for evasion attack
    - Stolen models can be used to construct gradient-based adversarial examples
        - [Papernot et al., Practical Black-Box Attacks against Machine Learning, ASIA CCS, 2017]

Original image

Adversarial perturbation

Adversarial example




$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y))$$

[Goodfellow+, Explaining and harnessing adversarial examples, ICLR 2015]

# Attack based on Equation Solver

- [Tramer+, Stealing machine learning models via prediction APIs, USENIX Security 2016]

- Basic idea: equation solving

  - LR's output:

$$f_1(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + \beta) \qquad \sigma(t) = 1/(1+e^{-t})$$

  - A linear equation:

$$\mathbf{w} \cdot \mathbf{x} + \beta = \sigma^{-1}(f_1(\mathbf{x}))$$

  - For w ∈ $R^d$ and β ∈ R, d+1 equations are necessary and sufficient to perfectly recover w and β

# JBDA (Jacobian-Based Dataset Augmentation) Attack

- [Papernot+, Practical black-box attacks against machine learning. ASIA CCS 2017]
- Goal: creating adversarial examples using a substitute model
- Jacobian-Based Dataset Augmentation

**Algorithm 1 - Substitute DNN Training:** for oracle $\tilde{O}$, a maximum number $max_\rho$ of substitute training epochs, a substitute architecture $F$, and an initial training set $S_0$.

**Input:** $\tilde{O}$, $max_\rho$, $\boxed{S_0}$, $\lambda$

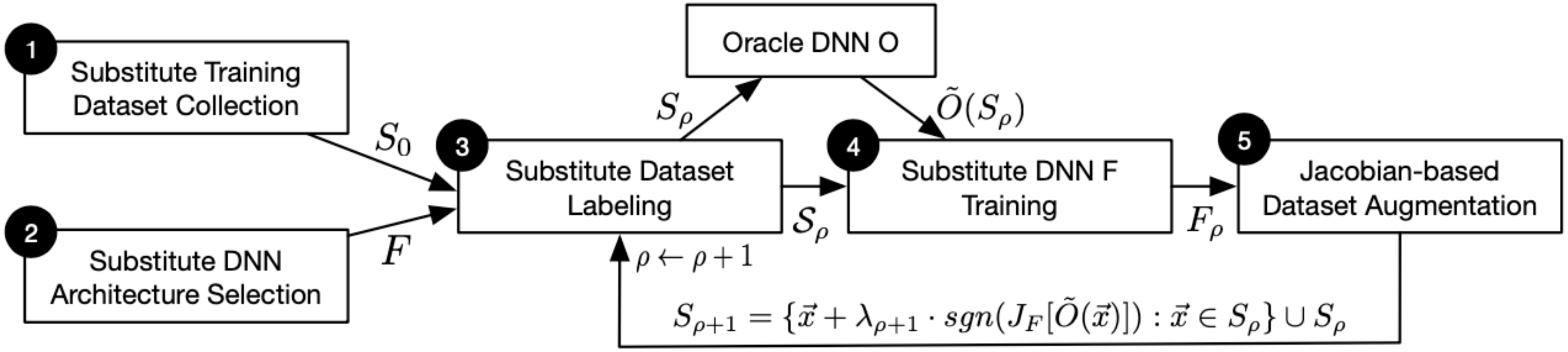
$S_0$: Seed dataset

Define architecture $F$
**for** $\rho \in 0 \,..\, max_\rho - 1$ **do**
  *// Label the substitute training set*
  $\boxed{D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}}$
  *// Train F on D to evaluate parameters* $\theta_F$
  $\boxed{\theta_F \leftarrow \text{train}(F, D)}$
  *// Perform Jacobian-based dataset augmentation*
  $\boxed{S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho}$
**end for**
**return** $\theta_F$

Query and record victim's response

Train the clone mode

Augment the seed set



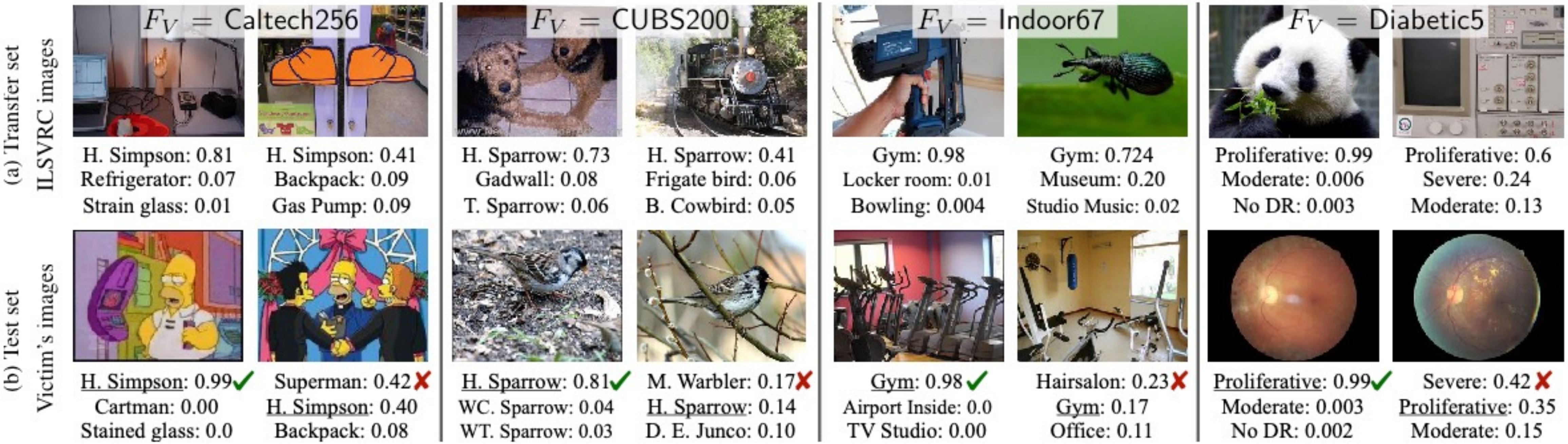
# Knockoff Nets

- [Orekondy+, Knockoff nets: Stealing functionality of black-box models, CVPR 2019]

- Idea: use inputs from public datasets (e.g., ImageNet) as queries
  - So far, we've used synthetic inputs for query

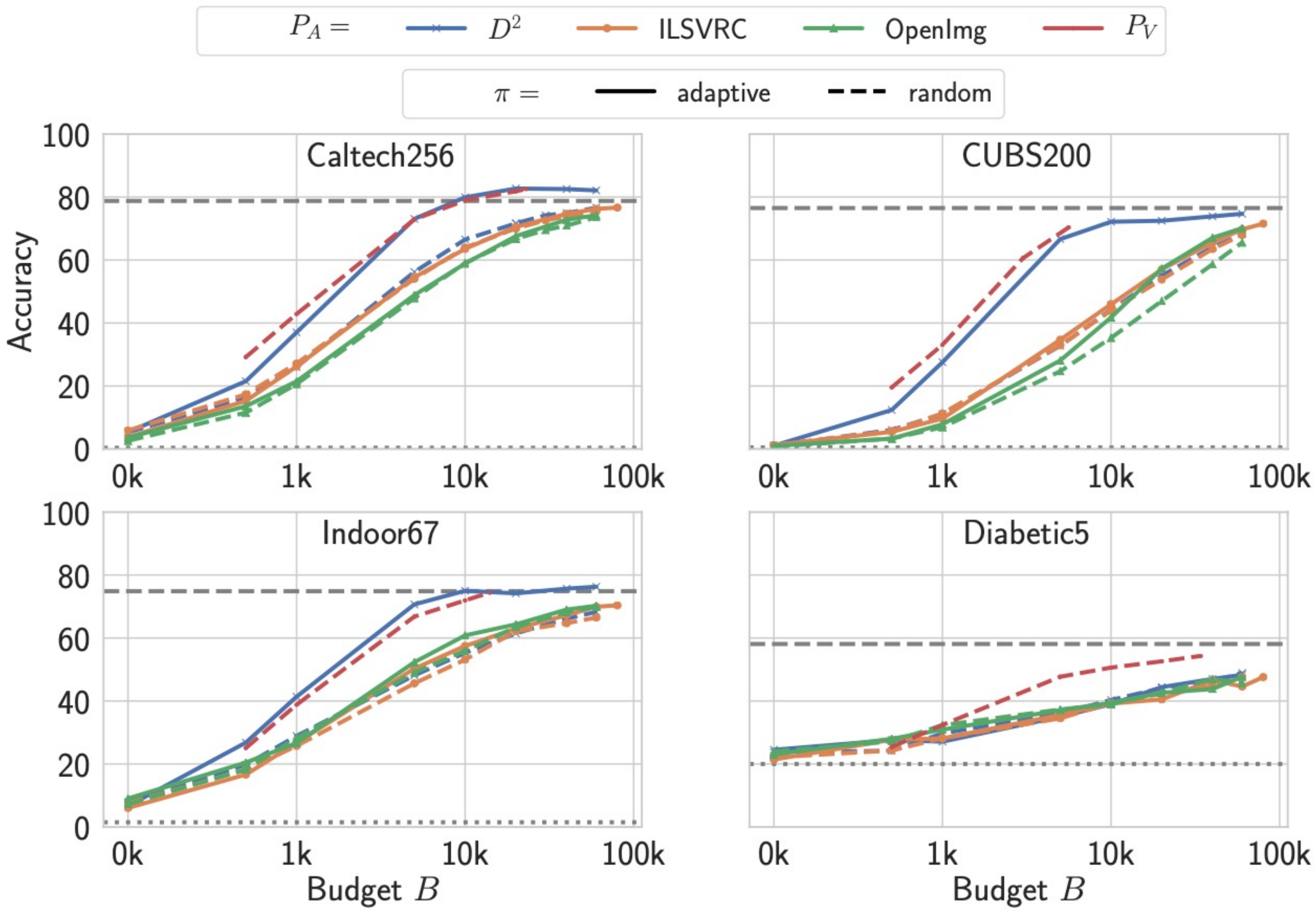
Attack queries (OOD, imagenet)

Data from victim's training distribution $F_V$



# Knockoff Nets

**Result: we can clone the victim models surprisingly well with OOD queries!**




| Blackbox ($F_V$) | $\lvert\mathcal{D}_V^{\text{train}}\rvert + \lvert\mathcal{D}_V^{\text{test}}\rvert$ | Output classes $K$ |
|---|---|---|
| `Caltech256` [11] | 23.3k + 6.4k | 256 general object categories |
| `CUBS200` [36] | 6k + 5.8k | 200 bird species |
| `Indoor67` [26] | 14.3k + 1.3k | 67 indoor scenes |
| `Diabetic5` [1] | 34.1k + 1k | 5 diabetic retinopathy scales |

**Table 1: Four victim blackboxes $F_V$.** Each blackbox is named in the format: [dataset][# output classes].

**Choice of $P_A$**

i. $P_A = P_V$ (KD)
ii. $P_A$ = ILSVRC
iii. $P_A$ = OpenImages (v4: 9.2M images from Flickr. A 550K subset of unique images by sampling 2k from each of 600 categories.
iv. $P_A = D^2$, The universe (the dataset of datasets) all in table 1 + ILSVRC + OpenImages

# Defense Techniques

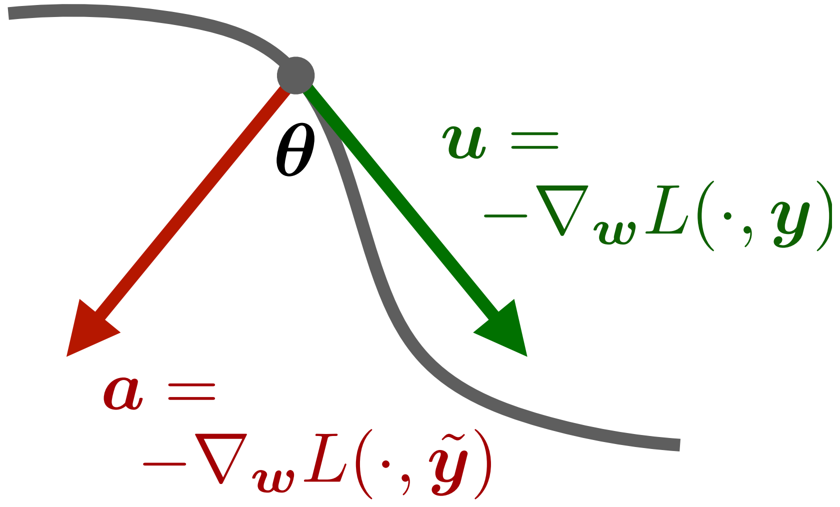
# PP (Prediction Poisoning)

- [Orekondy+, Prediction poisoning: Towards defenses against DNN model stealing attacks, ICLR 2020]

- **Insight**: unlike a benign user, a model stealing <u>attacker additionally uses the predictions to train a replica model</u>

- **Idea:** <u>introduce controlled perturbations to predictions:</u> we can poison the attacker's training objective, especially the gradient signal

Attacker's Loss Landscape




Our Perturbation Objective:

$$\underset{\tilde{\boldsymbol{y}}}{\operatorname{argmax}} \ \boldsymbol{\theta} \quad \text{s.t} \quad \operatorname{dist}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) \leq \epsilon$$

# PP (Prediction Poisoning)

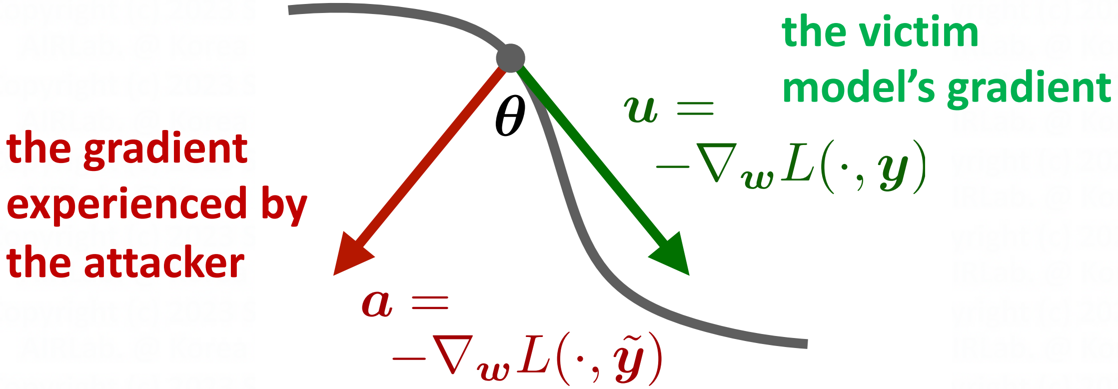
$$\max_{\boldsymbol{a}} \; 2(1 - \cos \angle(\boldsymbol{a}, \boldsymbol{u})) = \max_{\hat{\boldsymbol{a}}} \; ||\hat{\boldsymbol{a}} - \hat{\boldsymbol{u}}||_2^2$$

Maximum angular deviation (MAD)

$$\begin{cases} \boldsymbol{u} = -\nabla_{\boldsymbol{w}} L(F(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y}) = \nabla_{\boldsymbol{w}} \sum_k y_k \log F(\boldsymbol{x};\boldsymbol{w})_k = \sum_k y_k \nabla_{\boldsymbol{w}} \log F(\boldsymbol{x};\boldsymbol{w})_k = \boldsymbol{G}^T \boldsymbol{y} \\ \boldsymbol{a} = -\nabla_{\boldsymbol{w}} L(F(\boldsymbol{x};\boldsymbol{w}), \tilde{\boldsymbol{y}}) = \nabla_{\boldsymbol{w}} \sum_k \tilde{y}_k \log F(\boldsymbol{x};\boldsymbol{w})_k = \sum_k \tilde{y}_k \nabla_{\boldsymbol{w}} \log F(\boldsymbol{x};\boldsymbol{w})_k = \boldsymbol{G}^T \tilde{\boldsymbol{y}} \end{cases}$$

$$\begin{aligned} \max_{\tilde{\boldsymbol{y}}} \quad & \left\| \frac{\boldsymbol{G}^T \tilde{\boldsymbol{y}}}{||\boldsymbol{G}^T \tilde{\boldsymbol{y}}||_2} - \frac{\boldsymbol{G}^T \boldsymbol{y}}{||\boldsymbol{G}^T \boldsymbol{y}||_2} \right\|_2^2 \\ \text{where} \quad & \boldsymbol{G} = \nabla_{\boldsymbol{w}} \log F(\boldsymbol{x};\boldsymbol{w}) \\ \text{s.t} \quad & \tilde{\boldsymbol{y}} \in \Delta^K \\ & \text{dist}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) \leq \epsilon \\ & \arg\max_k \tilde{\boldsymbol{y}}_k = \arg\max_k \boldsymbol{y}_k \end{aligned}$$

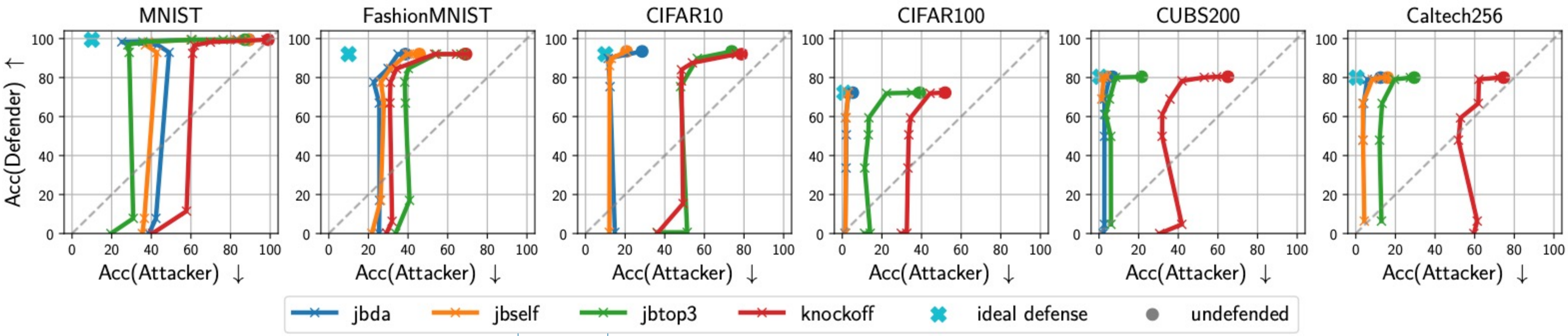
Attacker's Loss Landscape




Our Perturbation Objective:

$$\operatorname*{argmax}_{\tilde{\boldsymbol{y}}} \; \boldsymbol{\theta} \quad \text{s.t} \quad \text{dist}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) \leq \epsilon$$

# Attacks vs. PP

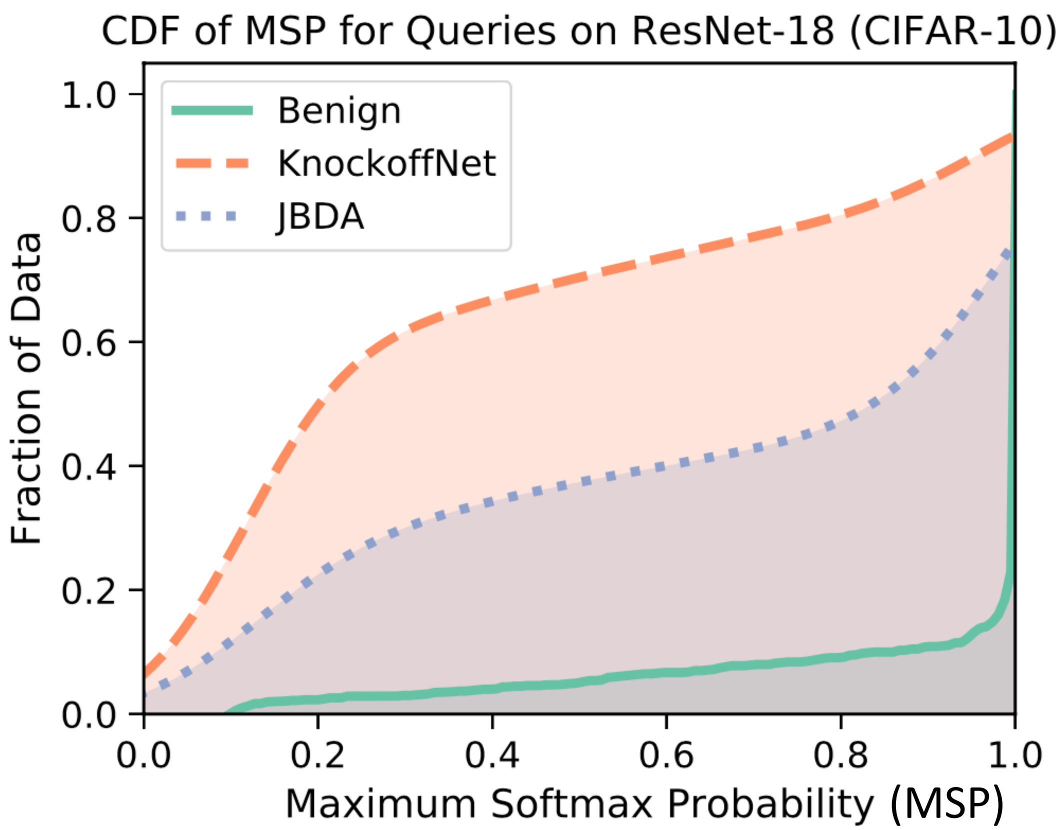



[Juuti+, PRADA: Protecting against DNN Model Stealing Attacks, EuroS&P 2019

- Curves are obtained by varying degree of perturbation ε
- MAD provides reasonable operating points (above the diagonal), where defender achieves significantly higher test accuracies compared to the attacker

# AM (Adaptive Misinformation)

- [Kariyappa+, Defending against model stealing attacks with adaptive misinformation, CVPR 2020]

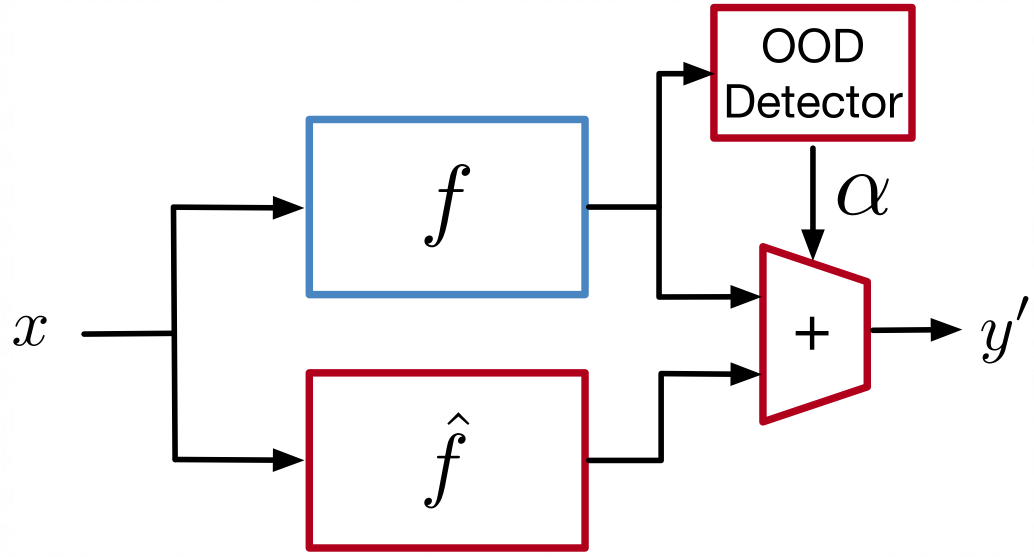



- All existing attacks invariably generate Out-Of-Distribution (OOD) queries

- Low MSP values indicate OOD data
  - [Hendrycks & Gimpel. A baseline for detect ing misclassified and out-of-distribution examples in neural networks, ICLR 2017]

# AM (Adaptive Misinformation)

- AM selectively sends incorrect predictions for queries that are deemed OOD
- ID queries are serviced with correct predictions




1) OOD detector

$$Det(x) = \begin{cases} ID & \text{if } \max_i(y_i) > \tau \\ OOD & otherwise \end{cases}$$

2) Model training with outlier exposure

$$\mathbb{E}_{(x,y)\in\mathcal{D}_{\text{in}}}\left[\mathcal{L}\left(f\left(x\right),y\right)\right] + \lambda\mathbb{E}_{x'\in\mathcal{D}_{\text{out}}}\left[\mathcal{L}\left(f\left(x'\right),\mathcal{U}\right)\right]$$

uniform dist

3) Misinformation function $\hat{f}$
: trained to minimize the probability of the correct class f(x,y)

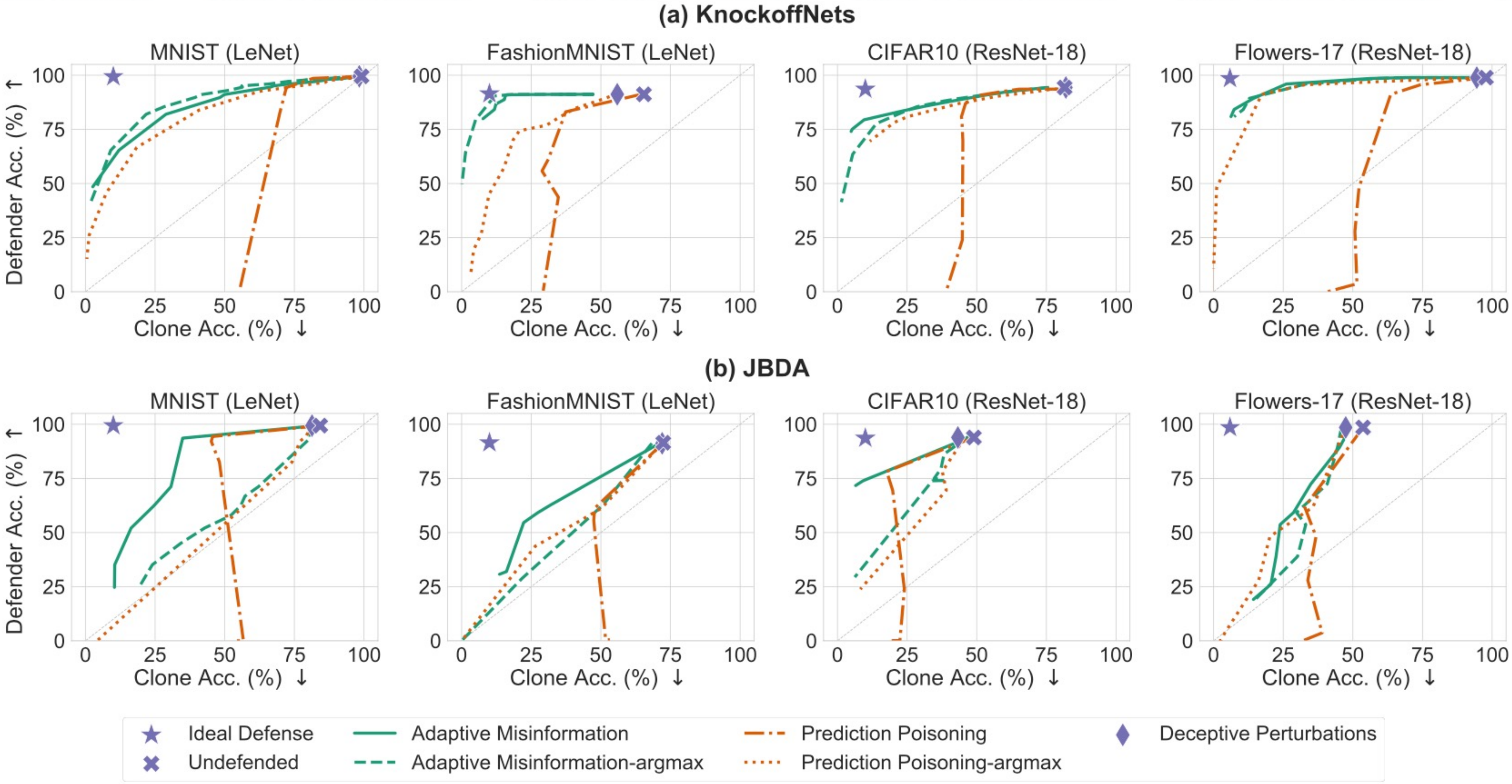
$$loss = \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{in}}}\left[-log(1-\hat{f}(x,y))\right]$$

4) Adaptive misinformation injection

$$y' = (1-\alpha)f(x;\theta) + (\alpha)\hat{f}(x;\hat{\theta})$$
$$where \quad \alpha = S(y_{max} - \tau)$$
$$S(z) = \frac{1}{1+e^{\nu z}}$$

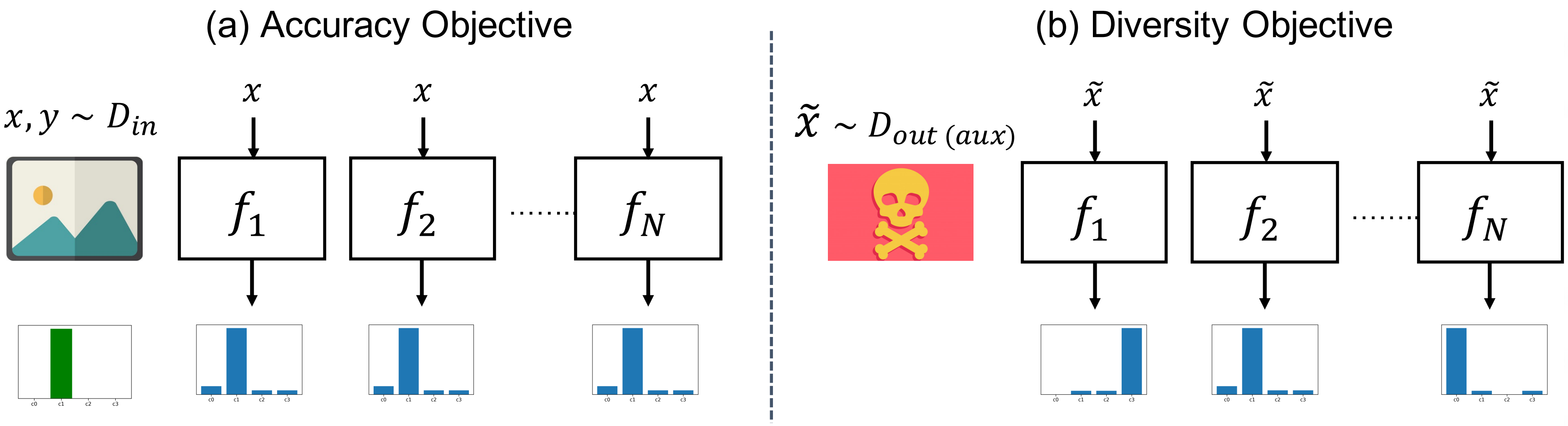
$$\begin{cases} \alpha < 0.5 & \text{if ID: } y_{max} > \tau \\ \alpha > 0.5 & \text{if OOD: } y_{max} < \tau \end{cases}$$

# Defender vs Clone Accuracy

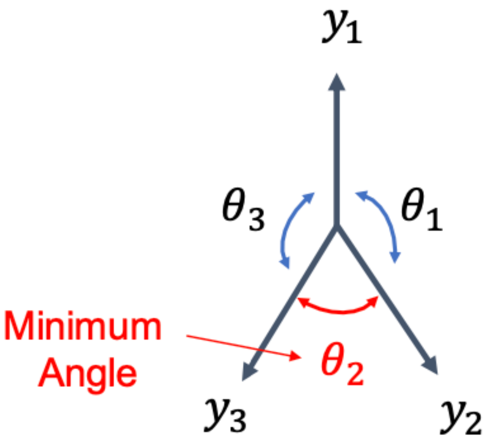


# EDM: Ensemble of Diverse Models

- [Kariyappa+, Protecting DNNs from theft using an ensemble of diverse models, ICLR 2021]
- Use an ensemble of N models that have maximum output variety for OOD inputs



# EDM: Ensemble of Diverse Models




$$Coherence(\{y_i\}_{i=1}^{3}) = Cos(\theta_2)$$

$$coherence(\{\tilde{\boldsymbol{y}}_i\}_{i=1}^{N}) = \max_{\substack{a,b\in\{1,..,N\}\\ a\neq b}} CS(\tilde{\boldsymbol{y}}_a, \tilde{\boldsymbol{y}}_b).$$

$$DivLoss(\{\tilde{\boldsymbol{y}}_i\}_{i=1}^{N}) = \log\left(\sum_{1\leq a<b\leq N} \exp(CS(\tilde{\boldsymbol{y}}_a, \tilde{\boldsymbol{y}}_b))\right)$$

response to an ID input

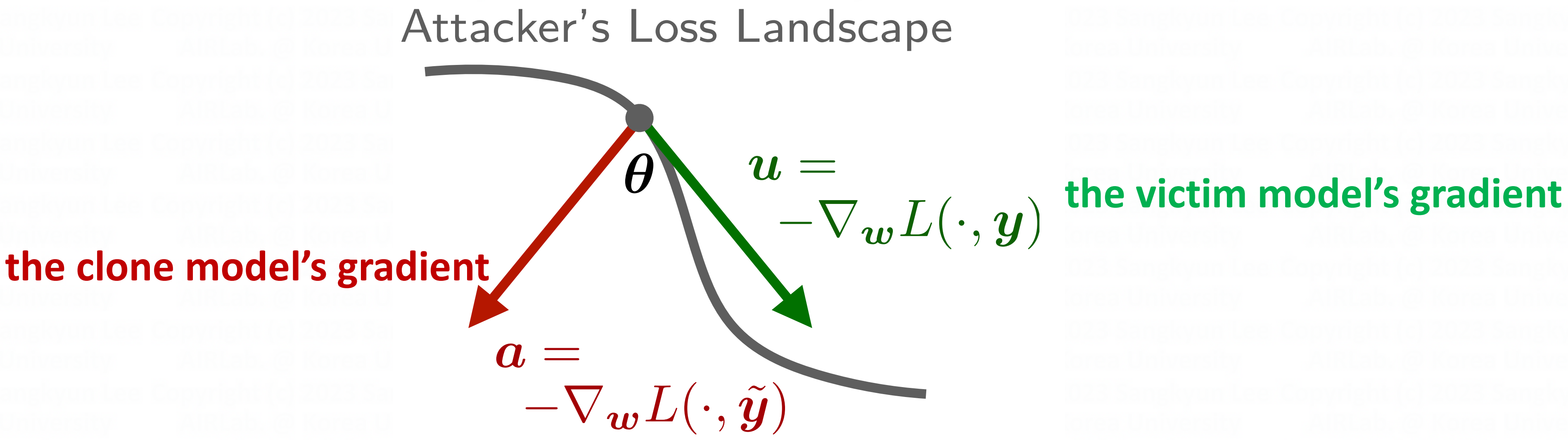
response to an OOD input

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}_{in},\tilde{x}\sim\mathcal{D}_{out}}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(\hat{\boldsymbol{y}}_i, \boldsymbol{y})\right) + \lambda_D \cdot DivLoss(\{\tilde{\boldsymbol{y}}_i\}_{i=1}^{N})\right]$$

$$where\ \ \hat{y}_i = f_i(x),\ \ \tilde{y}_i = f_i(\tilde{x}).$$

# Problems in PP ?

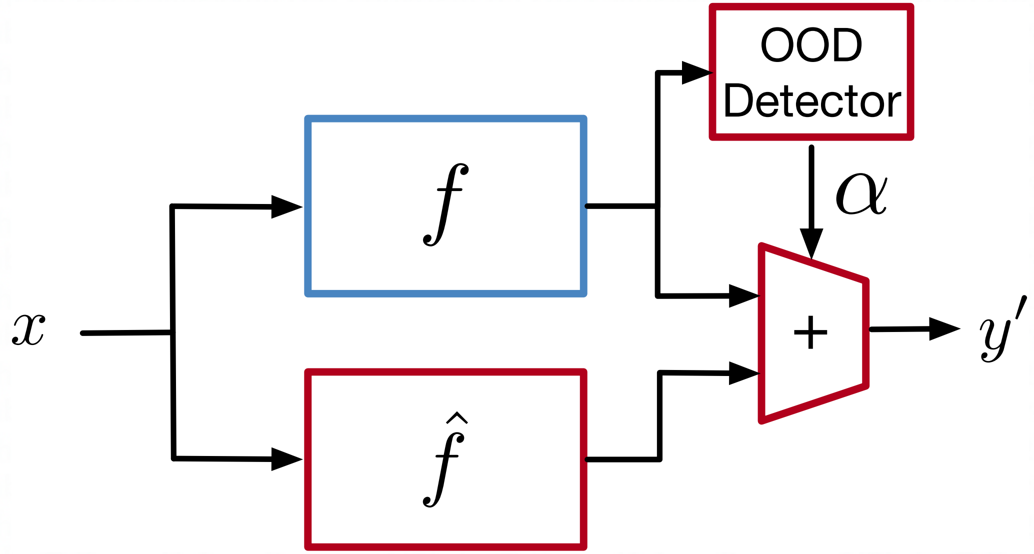
$$
\begin{cases}
\boldsymbol{u} = -\nabla_{\boldsymbol{w}} L(F(\boldsymbol{x};\boldsymbol{w}),\boldsymbol{y}) = \nabla_{\boldsymbol{w}} \sum_k y_k \log F(\boldsymbol{x};\boldsymbol{w})_k = \sum_k y_k \nabla_{\boldsymbol{w}} \log F(\boldsymbol{x};\boldsymbol{w})_k = \boldsymbol{G}^T \boldsymbol{y} \\
\boldsymbol{a} = -\nabla_{\boldsymbol{w}} L(F(\boldsymbol{x};\boldsymbol{w}),\tilde{\boldsymbol{y}}) = \nabla_{\boldsymbol{w}} \sum_k \tilde{y}_k \log F(\boldsymbol{x};\boldsymbol{w})_k = \sum_k \tilde{y}_k \nabla_{\boldsymbol{w}} \log F(\boldsymbol{x};\boldsymbol{w})_k = \boldsymbol{G}^T \tilde{\boldsymbol{y}}
\end{cases}
$$




**It should be written as:** $a = -\nabla_w L(F_A(x; w_A), \tilde{y}) = {G_A}^T \tilde{y}$

**But instead, the authors assumed that the defender knows the attacker's AI model**

# Problems in AM ?




1) OOD detector

$$Det(x) = \begin{cases} ID & \text{if } \max_i(y_i) > \tau \\ OOD & otherwise \end{cases}$$

4) Adaptive misinformation injection

$$y' = (1-\alpha) f(x;\theta) + (\alpha)\,\hat{f}(x;\hat{\theta}) \quad \begin{cases} \alpha < 0.5 & \text{if ID: } y_{max} > \tau \\ \alpha > 0.5 & \text{if OOD: } y_{max} < \tau \end{cases}$$

$$where \quad \alpha = S(y_{max} - \tau)$$

$$S(z) = \frac{1}{1+e^{\nu z}}$$

**Running the authors' github code, the OOD detector is perfect ($\alpha$ is 0 for ID and 1 for OOD inputs)**

**They used attack queries used in experiments to train the OOD detector!**

# Problems in EDM

**Knowledge of OOD data (= attack queries) is assumed**

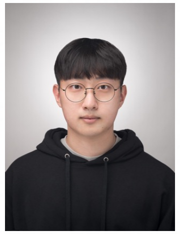
**Otherwise, we found that EDM loses its defense capability**

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y\sim\mathcal{D}_{in}\ \boxed{\tilde{x}\sim\mathcal{D}_{out}}} \left[ \left( \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{CE}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}) \right) + \lambda_D \cdot DivLoss(\{\tilde{\boldsymbol{y}}_i\}_{i=1}^{N}) \right]$$

$$where\ \ \hat{y}_i = f_i(x),\ \ \tilde{y}_i = f_i(\tilde{x}).$$

# Model Stealing Defense against Exploiting Information Leak through the Interpretation of Deep Neural Nets

Jeonghyun Lee, Sungmin Han, Sangkyun Lee*



School of Cybersecurity
Korea University, South Korea

IJCAI-22

# Model Stealing Attack

query $x_q$

ML-as-a-Service

MLaaS

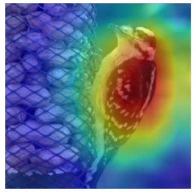
query $x_q$

response $f(x_q)$

$\boxed{I(x_q)}$

$f(x_q)$ Softmax output

$I(x_q)$



Attribution map

**XAI provides a new attack surface**

Milli et al., Model Reconstruction from Model Explanations, In Proceedings of the Conference on Fairness, Accountability, and Transparency, 2019

transfer set $\{(x_q, f(x_q), I(x_q))\}$

Train a clone model

To avoid query charges

Preparation for main attack
- Model inversion (data privacy)
- Adversarial attack

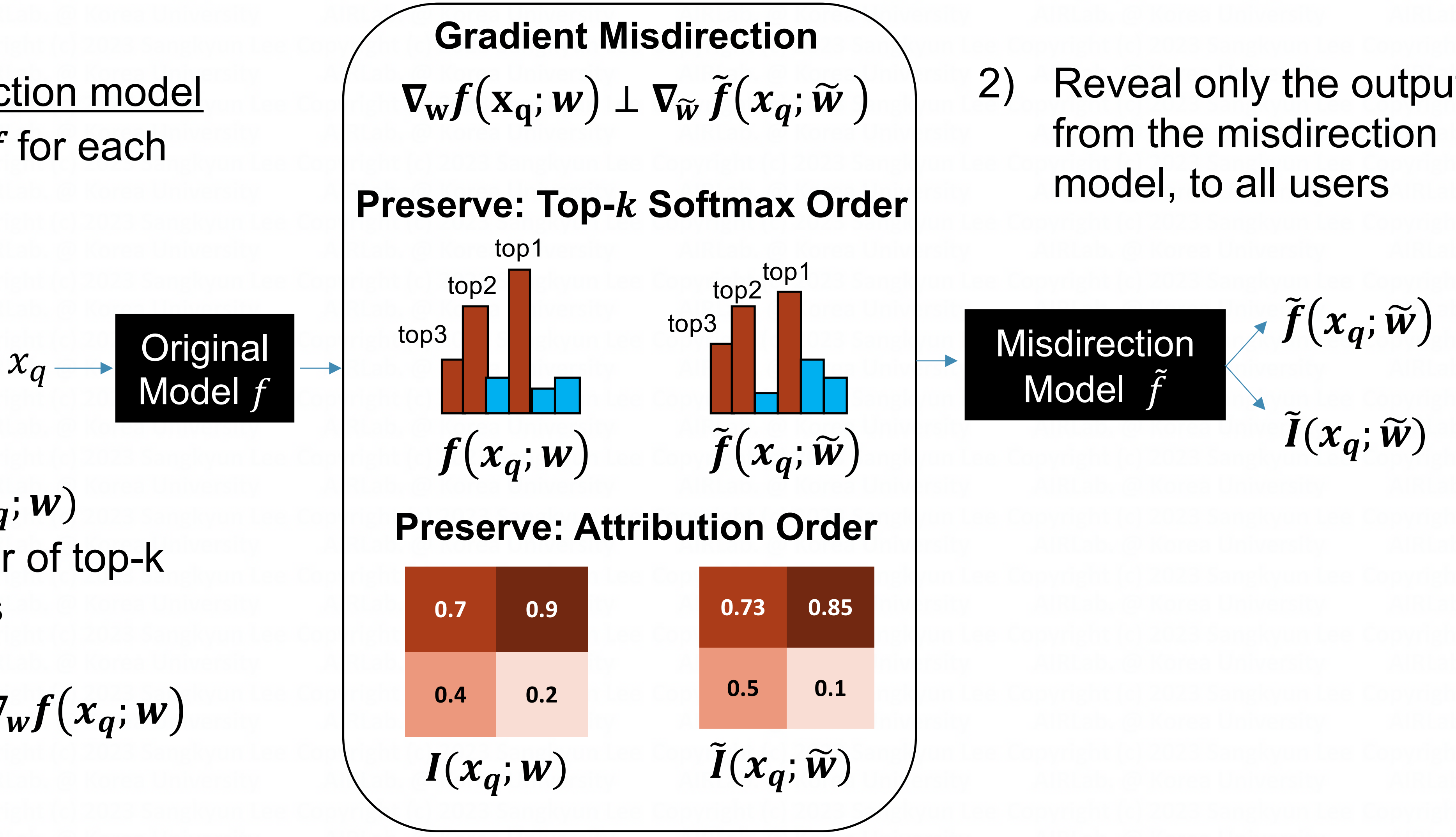
**We also wanted to solve the issues in PP, AM & EDM!**

# Proposed Method: DeepDefense

**Idea**:

1) Build a <u>misdirection model</u> $\tilde{f}$ of the victim $f$ for each query $x_q$

2) Reveal only the outputs from the misdirection model, to all users




- $\tilde{f}(\mathbf{x_q}; \tilde{\mathbf{w}}) \approx f(\mathbf{x_q}; \mathbf{w})$
  : Keep the order of top-k softmax indices

- $\nabla_{\mathbf{w}} \tilde{f}(\mathbf{x_q}; \tilde{\mathbf{w}}) \perp \nabla_{\mathbf{w}} f(\mathbf{x_q}; \mathbf{w})$

# Gradients in Parts

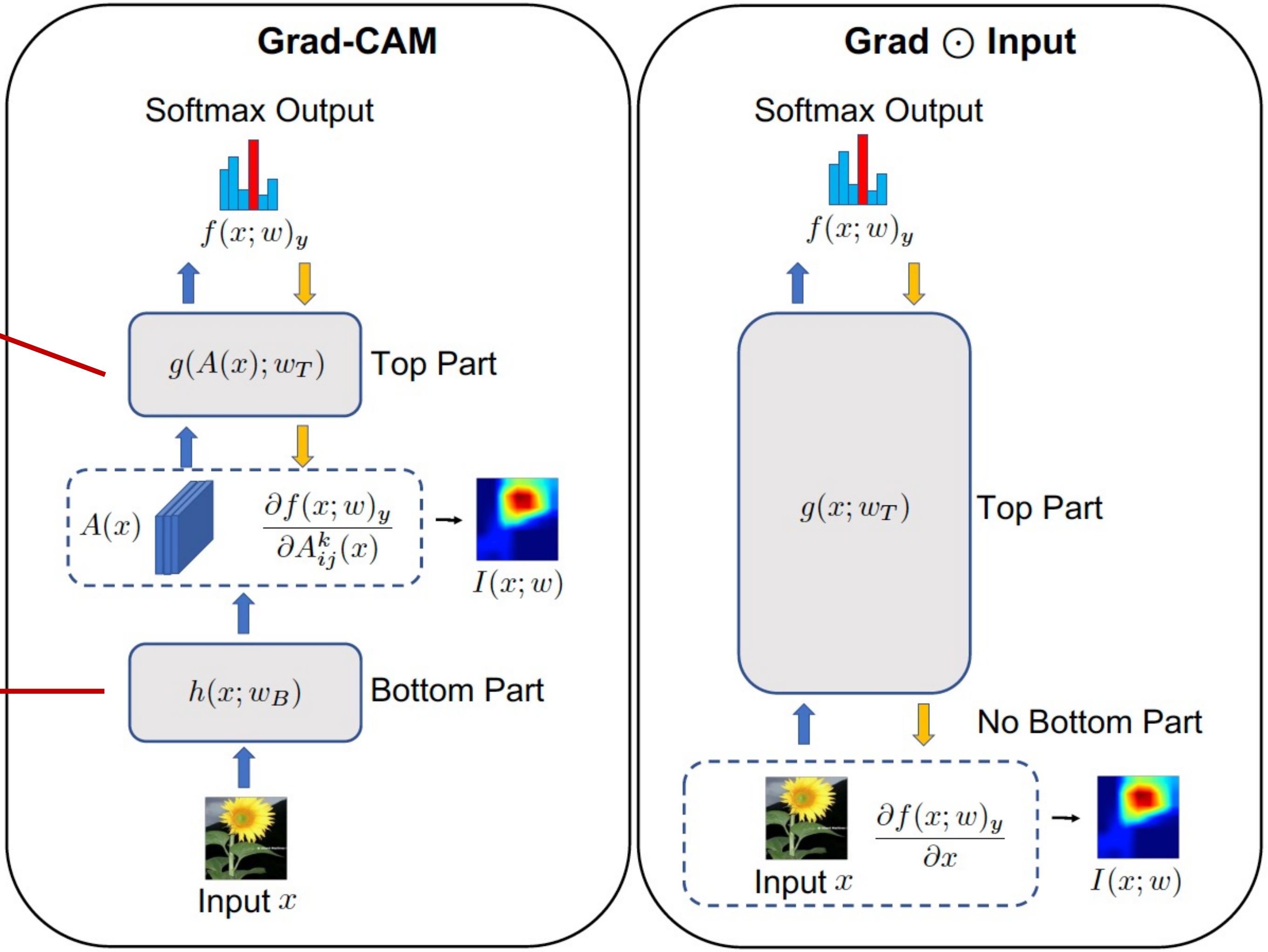
**Observation: parts of gradients have different flexibility to be used for perturbation**

**Top part:** backward path that providing the gradient $\frac{\partial f(x;w)_y}{\partial A_{ij}^k(x)}$

- **Flexible**

**Bottom part :** forward path that providing the activation maps $A(x)$

- **Not so much flexible**



# Gradient Misdirection

**The misdirection model is required to have gradients orthogonal to the gradients of the original model:**

$$\nabla_{\widetilde{w}_B} \widetilde{f}(x_q; \widetilde{w})_y \perp \nabla_{w_B} f(x_q; w)_y \qquad \textbf{Bottom part}$$

$$\nabla_{\widetilde{w}_T} \widetilde{f}(x_q; \widetilde{w})_y \perp \nabla_{w_T} f(x_q; w)_y \qquad \textbf{Top part}$$

**Misdirection model** **Original model**

**We reformulate this as follows (with a hyperparameter $0 \leq \alpha \leq 1$):**

$$\mathcal{L}_{\text{orth}}(x_q, \widetilde{w}) := \alpha \left| \cos \angle (\nabla_{w_B} f(x_q; w)_y, \nabla_{\widetilde{w}_B} \widetilde{f}(x_q; \widetilde{w})_y) \right| + (1 - \alpha) \left| \cos \angle (\nabla_{w_T} f(x_q; w)_y, \nabla_{\widetilde{w}_T} \widetilde{f}(x_q; \widetilde{w})_y) \right|$$

# Learning the Misdirection Model

**Constrained Optimization Problem**

$$\min_{\tilde{w}} \mathcal{L}_{\mathrm{orth}}(x_q, \tilde{w}) := \alpha \left| \cos \angle(\nabla_{w_B} f(x_q; w)_y, \nabla_{\tilde{w}_B} \tilde{f}(x_q; \tilde{w})_y) \right| + (1-\alpha) \left| \cos \angle(\nabla_{w_T} f(x_q; w)_y, \nabla_{\tilde{w}_T} \tilde{f}(x_q; \tilde{w})_y) \right|$$

$$s.t. \quad \tilde{f}(x_q; \tilde{w})_{s_1} \geq \cdots \geq \tilde{f}(x_q; \tilde{w})_{s_k} \geq \max_{j \in S'} \tilde{f}(x_q; \tilde{w})_j .$$

$\longrightarrow$ Functionality preservation

- $s_i$: the index of $i$-th largest value in the original softmax vector
- $S' \coloneqq \{1, \dots, K\} \setminus \{s_1, \dots, s_k\}$

$$\tilde{I}(x_q; \tilde{w})_{a_1} \geq \tilde{I}(x_q; \tilde{w})_{a_2} \geq \cdots \geq \tilde{I}(x_q; \tilde{w})_{a_{H \times W}} .$$

$\longrightarrow$ Interpretability preservation

- $a_i$: the index of $i$-th largest value attribution in the original attribution map
- $H \times W$: the size of attribution maps

# Reformulation into an Unconstrained Optimization

$$\mathcal{L}_{\mathrm{DD}}(x_q, \tilde{w}) := \mathcal{L}_{\mathrm{orth}}(x_q, \tilde{w}) + \lambda_1 \mathcal{L}_{\mathrm{pred}}(x_q, \tilde{w}) + \lambda_2 \mathcal{L}_{\mathrm{int}}(x_q, \tilde{w})$$

$$\mathcal{L}_{\mathrm{pred}}(x_q; \tilde{w}) := \sum_{i=1}^{k-1} \left(\tilde{f}(x_q; \tilde{w})_{s_{i+1}} - \tilde{f}(x_q; \tilde{w})_{s_i}\right)^+ + \left(\max_{j \in \{1,\ldots,K\} \setminus \{s_1,\ldots,s_k\}} \tilde{f}(x_q; \tilde{w})_j - \tilde{f}(x_q; \tilde{w})_{s_k}\right)^+$$

$$(z)^+ := \max\{z, 0\}$$

$$\mathcal{L}_{\mathrm{int}}(x_q, \tilde{w}) := \sum_{i=1}^{H \times W - 1} \mathcal{L}\left((\tilde{I}(x_q; \tilde{w})_{a_{i+1}} - \tilde{I}(x_q; \tilde{w})^+_{a_i}\right)$$

- Solver: SGD with momentum

# Sparse Layer Selection

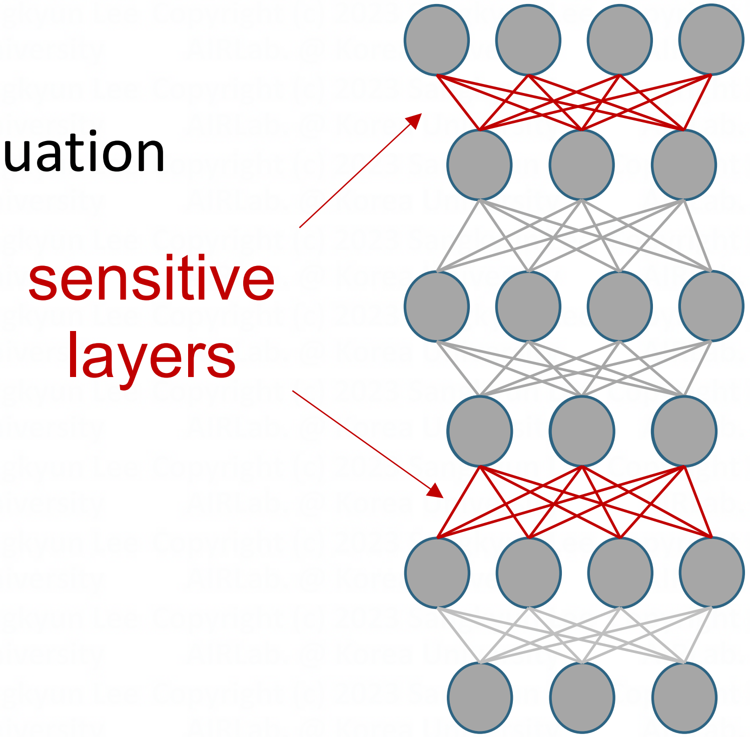
**For speed-up, we use only the parts of gradients corresponding to the <u>most sensitivity layers</u> to the model's output**

**Layer sensitivity :** $S_\ell \coloneqq \frac{1}{N}\sum_{i=1}^{N} ||\nabla_{\mathrm{w}_\ell} \mathrm{f}(\mathrm{x_i};\mathrm{w})_{\mathrm{y_i}}||_1$

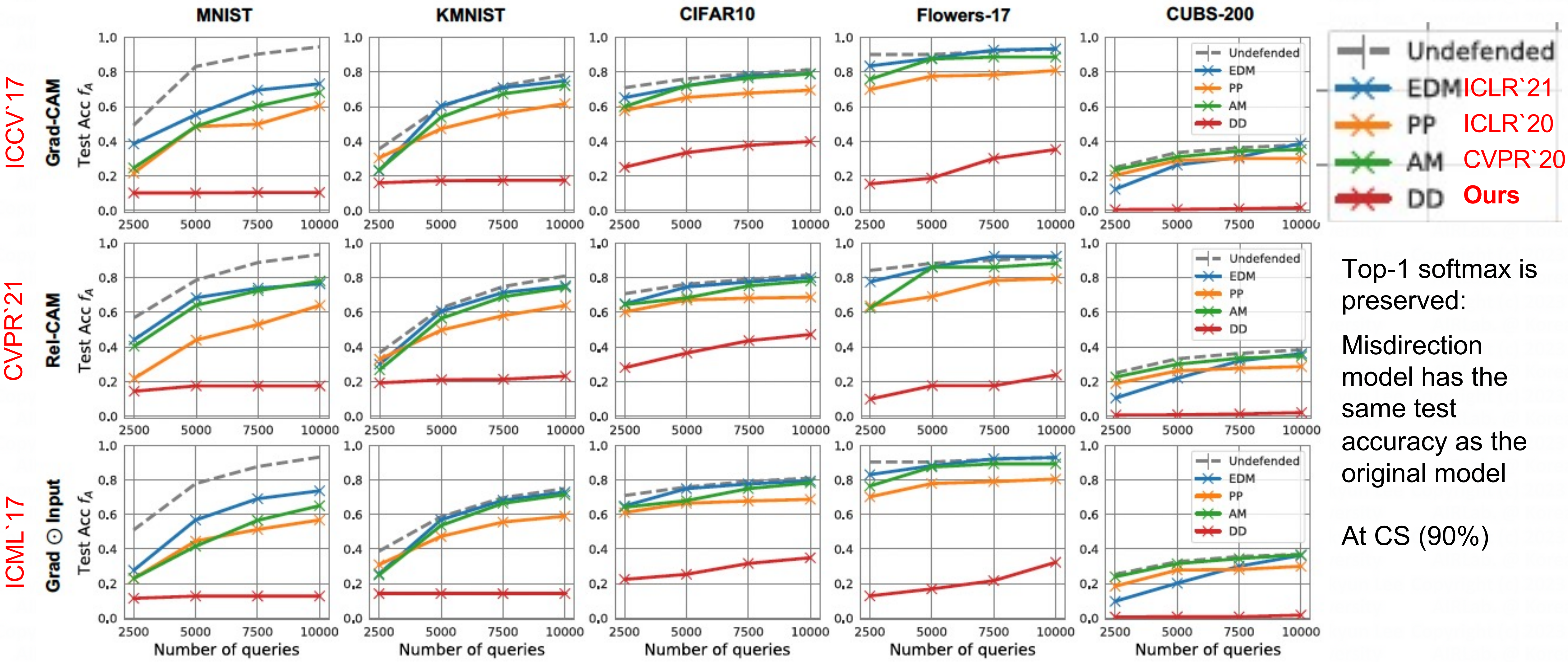
$\{(x_i, y_i)\}_{i=1}^{N}$: a part of training data for sensitivity evaluation

**Cumulative sensitivity :** $CS(\ell) \coloneqq \frac{\sum_{i=1}^{\ell} S_{(i)}}{\sum_{i=1}^{L} S_{(i)}} \times 100\ (\%)$

$$S_{(1)} \geq S_{(2)} \geq \cdots \geq S_{(L)}$$



# Defense Performance (Attacker's Test Accuracy)




Top-1 softmax is preserved:

Misdirection model has the same test accuracy as the original model

At CS (90%)

❖ **Our method (DD) outperformed SOTA defense methods against model stealing**

# Computational Cost

**Relevance-CAM / Flowers17 dataset / ResNet-18**

The activation layer used for Relevance-CAM

| $\ell$ | CS (%) | # layers | $f_A$ Test Acc (%) PP | $f_A$ Test Acc (%) DD | Run time (sec) PP | Run time (sec) DD |
|---|---|---|---|---|---|---|
| 9 | 90 | 8 | 60.66 | 8.82 | 0.23 | 1.53 |
| | 70 | 4 | | 11.76 | | 1.04 |
| | 50 | 2 | | 10.29 | | 0.65 |
| 13 | 90 | 7 | 61.76 | 8.09 | 0.21 | 1.41 |
| | 70 | 4 | | 8.82 | | 0.97 |
| | 50 | 2 | | 10.29 | | 0.60 |
| 17 | 90 | 8 | 62.13 | 9.19 | 0.21 | 1.47 |
| | 70 | 6 | | 8.98 | | 1.38 |
| | 50 | 3 | | 11.40 | | 0.73 |

**DD showed consistent defense performance on the change of cumulative sensitivity, with reasonable computation time**

# Preservation of Interpretation Quality

## Quantitative

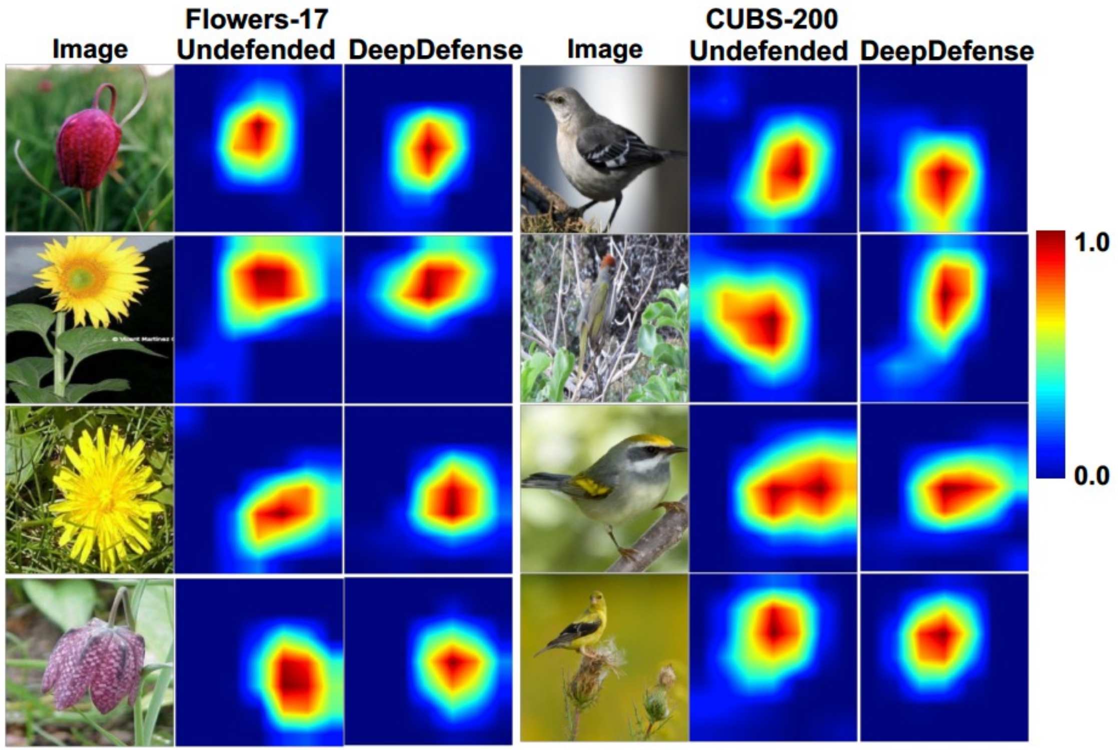
$$\text{Avg Drop} = \frac{1}{N}\sum_{i=1}^{N}(y_i^c - \tilde{y}_i^c)/y_i^c$$

$y_i^c$: score on the original input
$\tilde{y}_i^c$: score on the top p% attribution region

| Dataset | Grad-CAM | | Rel-CAM | | Grad ⊙ Input | |
|---|---|---|---|---|---|---|
| | Avg Drop $I$ | Avg Drop $\tilde{I}$ | Avg Drop $I$ | Avg Drop $\tilde{I}$ | Avg Drop $I$ | Avg Drop $\tilde{I}$ |
| MNIST | 0.7888±0.3691 | 0.7587±0.4091 | 0.5621±0.4980 | 0.5425±0.5019 | 0.5670±0.4020 | 0.5613±0.4024 |
| KMNIST | 0.7135±0.2834 | 0.6889±0.3169 | 0.7516±0.2909 | 0.7260±0.3962 | 0.5815±0.3591 | 0.6067±0.3499 |
| CIFAR-10 | 0.7622±0.3558 | 0.7753±0.3779 | 0.7365±0.3882 | 0.7042±0.3761 | 0.8647±0.2859 | 0.8588±0.2869 |
| Flowers-17 | 0.5130±0.3018 | 0.5152±0.3078 | 0.5033±0.3140 | 0.5046±0.3140 | 0.8287±0.2249 | 0.8256±0.2304 |
| CUBS-200 | 0.5593±0.4002 | 0.5747±0.4181 | 0.5649±0.4027 | 0.5934±0.4260 | 0.9581±0.1493 | 0.9647±0.1307 |

**No statistically significant difference in interpretation quality between the original and misdirected interpretations**

## Qualitative (Grad-CAM)

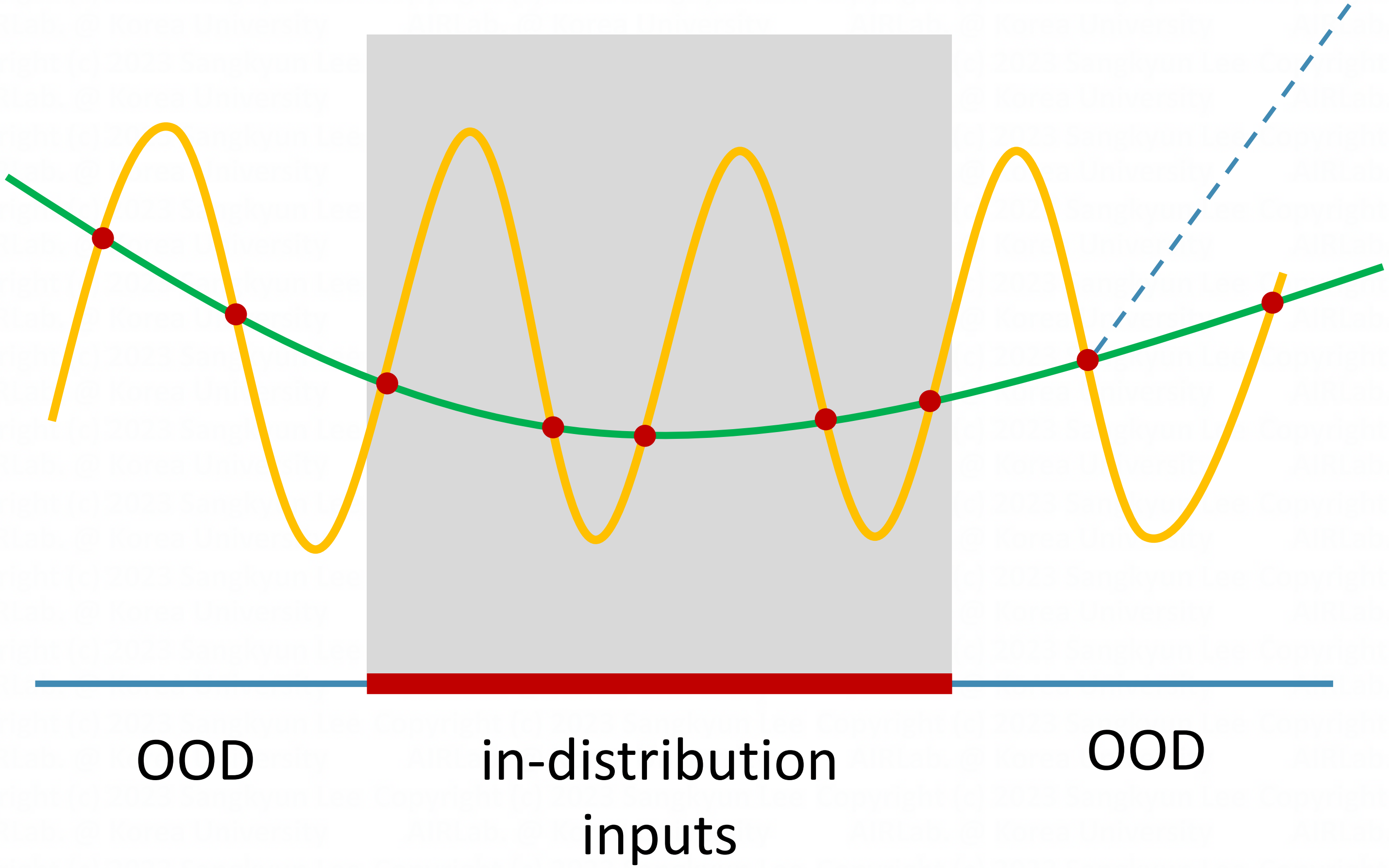



**The focused areas are preserved**

# Performance Measures

- How well two AI models (two functions) are matched?

The test is point-wise:
if we test only these points, we may conclude that the two models match well



# Performance Measures

- Fidelity Measures
  - ID point-wise error: low test error implies that $\hat{f}$ matches $f$ well for inputs distributed like the training samples

$$R_{\text{test}}(f,\hat{f}) = \sum_{(\mathbf{x},y)\in D} d(f(\mathbf{x}),\hat{f}(\mathbf{x}))/|D|$$

  - OOD point-wise error: for a set U of random vectors uniformly chosen in the input space,

$$R_{\text{unif}}(f,\hat{f}) = \sum_{\mathbf{x}\in U} d(f(\mathbf{x}),\hat{f}(\mathbf{x}))/|U|$$

    - $R_{unif}$ estimates the fraction of the full feature space on which $\hat{f}$ and $f$ disagree
    - |U| = 10,000 was sufficiently large to obtain stable error estimates for the models we analyzed
  - In the above, distances are measured for the 0-1 decisons
    - Class probability comparisons are denoted by $R^{TV}_{test}$ and $R^{TV}_{unif}$
- Recent papers tend to compare test accuracy rates between the victim and the clone models

# Conclusion

- Model stealing is a critical issue for AI model deployment:
  - Attackers can steal our AI models, with relatively cheap cost
  - Stolen models can be used for secondary attacks, e.g., evasion or model inversion attacks

- Attacks: Tramer, JBDA, KnockoffNet, ActiveThief, …, SwiftThief (IJCAI 2024)
- Defenses: PP, AM, EDM, …, DeepDefense (IJCAI, 2022)

- XAI
  - Could be a new attack surface for model stealers
  - May provide valuable information of AI’s vulnerabilities.
  - Libra-CAM (IJCAI, 2022): SOTA on CNN

- Other works: LLM-based S/W vulnerability repair & deobfuscation, security for robot AI

**Thank You!**

Sangkyun Lee (sangkyun@korea.ac.kr)