CS 499/599: Machine Learning Security
03.02: Privacy

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

Notice

- Due dates (in Mar.)
  - 7th: written paper critique
  - 9th: Final project presentation
  - 14th: Final exam (online)
  - 14th: Final project report
  - 16th: HW4 deadline (HW 1-3 late submissions are available until then; 50% of total will be given!)

- Sign-up (on Canvas)
  - Scribe lecture note [2 slots remain]
  - In-class paper presentation / discussion

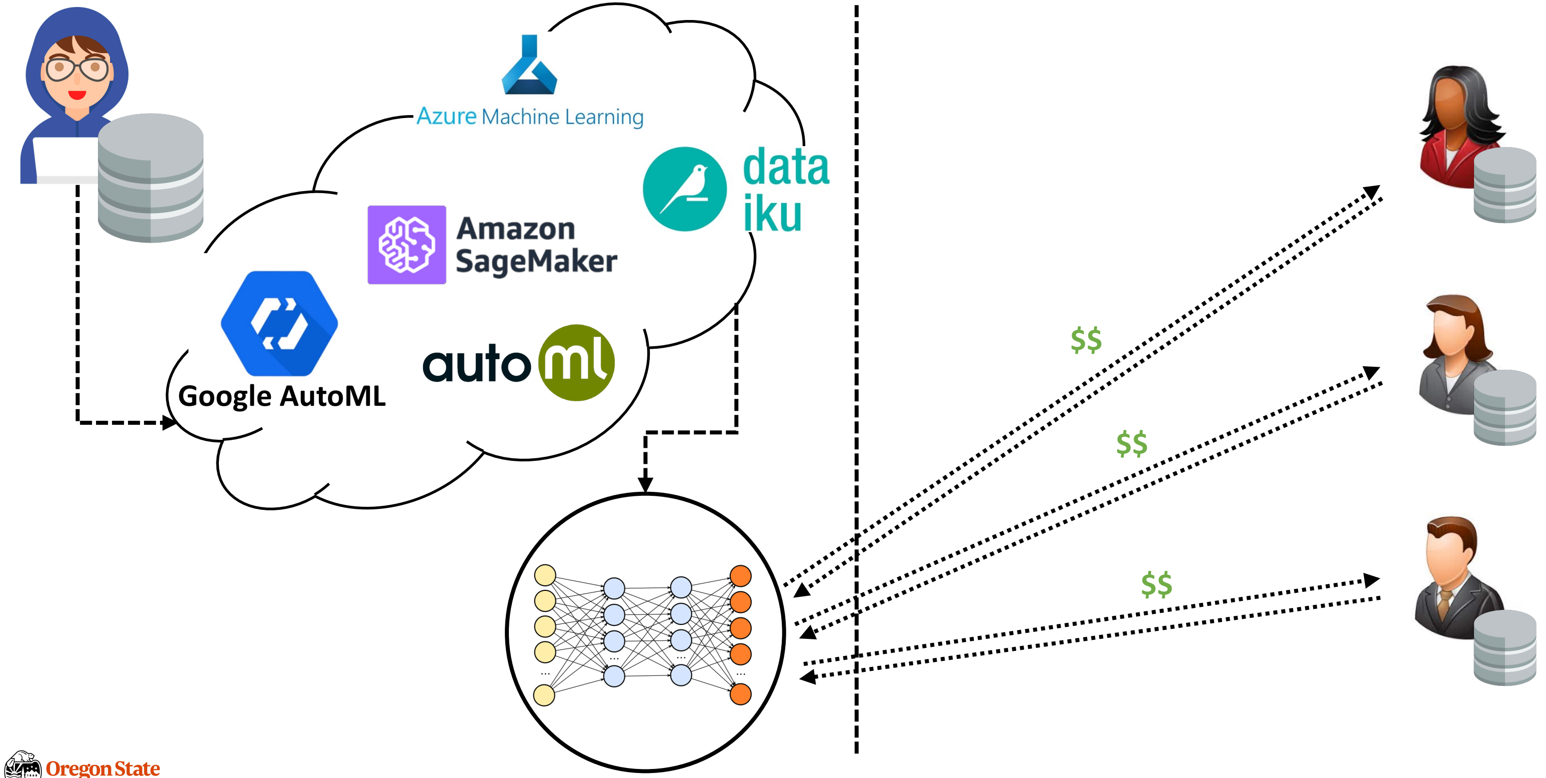
Topics for Today

- Privacy Attacks and Defenses
  - Non-ML: Data anonymization
  - Membership inference
    - Threat Model
    - Attacks: Yeom *et al.* and Shokri *et al.*
    - Defensive techniques
  - Model inversion
    - Threat Model
    - Attacks: Fredrikson *et al.* and Carlini *et al.*
    - Defensive techniques
  - Model extraction
    - Threat Model
    - Attacks:
      - Tramer *et al*.
      - Jagielski *et al*.
    - Defensive techiniques

Model Extraction: I Want Your “Trained” Model

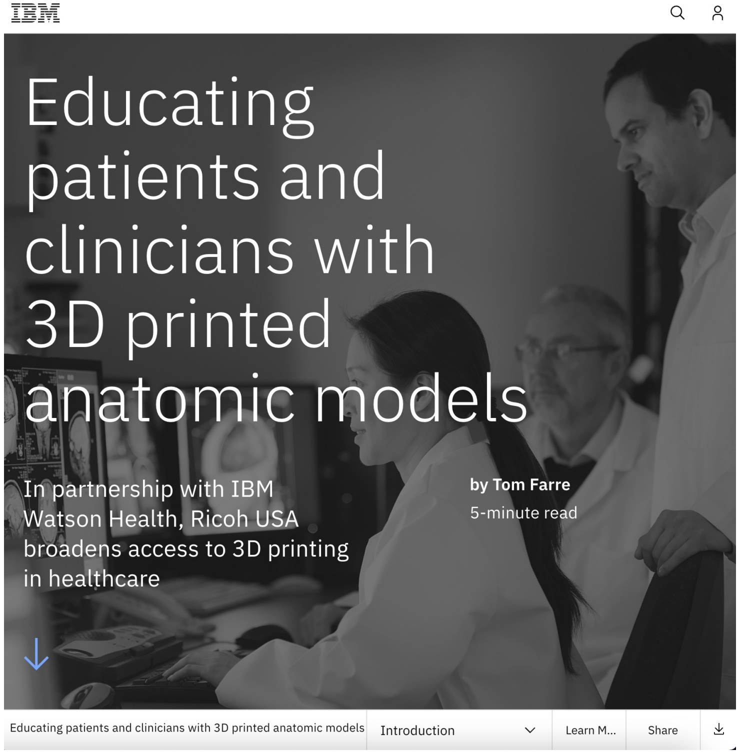
Emerging Machine Learning as a Service (MLaaS)

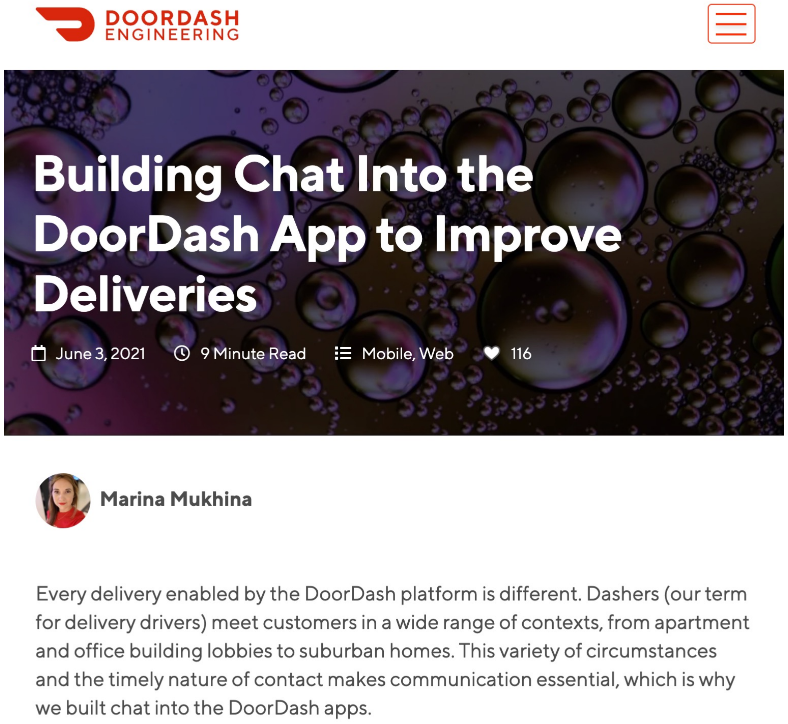
- You train ML models and reach out to customers



MLaaS Incentivizes Attackers

- To **steal** your models… what if you run:





Potential Downstream Attacks

- Exploiting stolen models, an adversary can:
  - Start a service with the stolen models with the same functionalities
  - Use the stolen model to craft adversarial examples
  - Extract private information from the stolen models

How Can We Steal ML Models?

Threat Model

- Model extraction attacks
  - **Goal**
    - To learn a new model $\hat{f}$ that closely approximates the target model f
  - **Knowledge**
    - Black-box (typically)
    - It's possible to know aux. information:
      - How does a model extract feature(s)?
      - What is the model's class we aim to extract?
      - What is the training algorithm / hyper-params used?
  - **Capability**
    - Has query access to the victim f (many times) with arbitrary inputs $\boldsymbol{x}$
    - Has computational power to do offline processing of query outputs $f(\boldsymbol{x})$

Service	White-box	Monetize	Confidence Scores	Logistic Regression	SVM	Neural Network	Decision Tree
Amazon [1]	✗	✗	✓	✓	✗	✗	✗
Microsoft [38]	✗	✗	✓	✓	✓	✓	✓
BigML [11]	✓	✓	✓	✓	✗	✗	✓
PredictionIO [43]	✓	✗	✗	✓	✓	✗	✓
Google [25]	✗	✓	✓	✓	✓	✓	✓

Threat Model

- Model extraction attacks
  - **Metrics**
    - Test error $R_{test}(\hat{f}, f)$: the average error between the outputs of $\hat{f}$ and f on D
    - Uniform error $R_{unif}(\hat{f}, f)$: $R_{test}(\hat{f}, f)$ on a set of uniform vectors
  - **Extraction accuracy:**
    - $1 - R_{test}(\hat{f}, f) \mid 1 - R_{unif}(\hat{f}, f)$

Model Extraction Attacks

- Equation-solving attack
  - **Setup:**
    - MLaaS APIs return confidence values $f(\boldsymbol{x})$
    - Those values are available to the attacker
  - **Binary logistic regression:**
    - Requires $d + 1$ predictions (queries), where d is the input dimension
  - **Results:**
    - Using $d + 1$ predictions, the attacker achieves the errors < 10^{-9}
    - The attacker requires 41 – 113 queries depending on the tasks

Model Extraction Attacks

- Equation-solving attack
  - **Setup:**
    - MLaaS APIs return confidence values $f(\boldsymbol{x})$
    - Those values are available to the attacker
  - **Multiclass LRs:**
    - Softmax vs. one-vs-rest (OvR)
    - Requires $c(d+1)$ queries, where c is the number of classes
  - **Multi-layer perceptron (MLPs):**
    - Requires $\alpha \cdot k$ predictions, where k is the number of unknown model parameters
    - Note: this work assumes MLPs with one hidden layer

Model Extraction Attacks

- Equation-solving attack
  - **Setup:**
    - MLaaS APIs return confidence values $f(\boldsymbol{x})$
    - Those values are available to the attacker
  - **Results:**
    - MLRs: Using $c(d+1)$ predictions, the attacker achieves the errors < 10^{-7}
    - MLPs: Require 5x times more queries for achieving the same error rate

Model	Unknowns	Queries	$1 - R_{\text{test}}$	$1 - R_{\text{unif}}$	Time (s)
Softmax	530	265	99.96%	99.75%	2.6
		530	100.00%	100.00%	3.1
OvR	530	265	99.98%	99.98%	2.8
		530	100.00%	100.00%	3.5
MLP	2,225	1,112	98.17%	94.32%	155
		2,225	98.68%	97.23%	168
		4,450	99.89%	99.82%	195
		11,125	99.96%	99.99%	89

Model Extraction Attacks

- Equation-solving attack
  - **Setup:**
    - MLaaS APIs return confidence values $f(\boldsymbol{x})$
    - Those values are available to the attacker
  - **Downstream security attacks on $\hat{f}$:**
    - Training data leakage in Kernel LR (KLR)
      - In KLR, the equation becomes $\sum_{r=1}^{s} \alpha_{i,r} K(\mathbf{x}, \mathbf{x}_r) + \beta_i$, where x_1, … x_s are *representers*

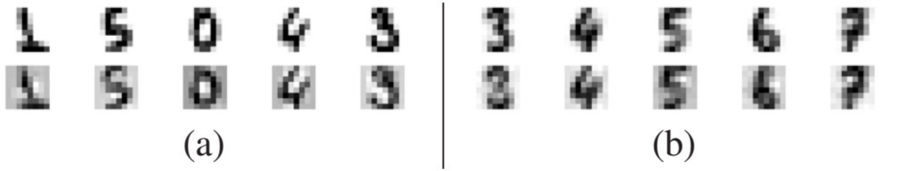


Figure 2: **Training data leakage in KLR models.** (a) Displays 5 of 20 training samples used as representers in a KLR model (top) and 5 of 20 extracted representers (bottom). (b) For a second model, shows the average of all 1,257 representers that the model classifies as a 3, 4, 5, 6 or 7 (top) and 5 of 10 extracted representers (bottom).

Model Extraction Attacks

- Equation-solving attack
  - **Setup:**
    - MLaaS APIs return confidence values $f(\boldsymbol{x})$
    - Those values are available to the attacker
  - **Downstream security attacks on $\hat{f}$:**
    - Model inversion attacks
      - Convert a black-box to a white-box setting
      - In Fredrikson *et al.*
        - » The attack requires 800k queries to reconstruct 40 individuals
        - » One can extract the model with 40k queries and achieve the same attack success
        - » Using the extracted $\hat{f}$ reduces the time from 16 hrs to 10 hrs

Model Extraction Attacks

- Decision tree path-finding attack
  - **Setup:**
    - MLaaS APIs return $f(x)$ with
      - The leaf node
      - (for the incomplete queries) the node where
    - Those values are available to the attacker

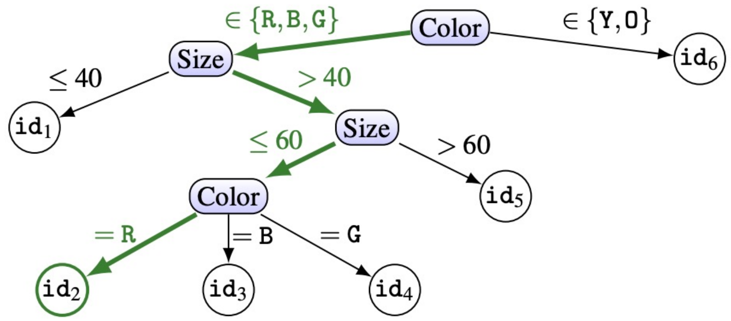


Figure 3: **Decision tree over features Color and Size.** Shows the path (thick green) to leaf $\texttt{id}_2$ on input $\mathbf{x} = \{Size = 50,\ Color = R\}$.

```
x_init ← {x_1, ..., x_d}                                  ▷ random initial query
Q ← {x_init}                                              ▷ Set of unprocessed queries
P ← {}                                      ▷ Set of explored leaves with their predicates
while Q not empty do
    x ← Q.POP()
    id ← O(x)                                             ▷ Call to the leaf identity oracle
    if id ∈ P then                                        ▷ Check if leaf already visited
        continue
    end if
    for 1 ≤ i ≤ d do                                      ▷ Test all features
        if IS_CONTINUOUS(i) then
            for (α, β] ∈ LINE_SEARCH(x, i, ε) do
                if x_i ∈ (α, β] then
                    P[id].ADD('x_i ∈ (α, β]')             ▷ Current interval
                else
                    Q.PUSH(x[i] ⇒ β)                      ▷ New leaf to visit
                end if
            end for
        else
            S, V ← CATEGORY_SPLIT(x, i, id)
            P[id].ADD('x_i ∈ S')                          ▷ Values for current leaf
            for v ∈ V do
                Q.PUSH(x[i] ⇒ v)                          ▷ New leaves to visit
            end for
        end if
    end for
end while
```

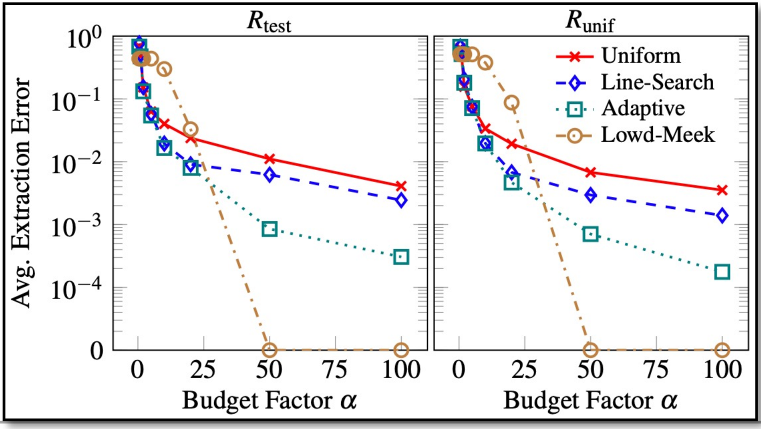
Model Extraction Attacks

- Decision tree path-finding attack
  - **Setup:**
    - MLaaS APIs return $f(\boldsymbol{x})$ with
      - The leaf node
      - (for the incomplete queries) the node where each computation halts
    - Those values are available to the attacker
  - **Results:**
    - All leaves are unique: 100% extraction success
    - Top-down: reduces # queries a lot & Duplicate leaves: a bit less effective

				Without incomplete queries			**With incomplete queries**		
Model	**Leaves**	**Unique IDs**	**Depth**	$1 - R_{\text{test}}$	$1 - R_{\text{unif}}$	Queries	$1 - R_{\text{test}}$	$1 - R_{\text{unif}}$	Queries
IRS Tax Patterns	318	318	8	100.00%	100.00%	101,057	100.00%	100.00%	29,609
Steak Survey	193	28	17	92.45%	86.40%	3,652	100.00%	100.00%	4,013
GSS Survey	159	113	8	99.98%	99.61%	7,434	100.00%	99.65%	2,752
Email Importance	109	55	17	99.13%	99.90%	12,888	99.81%	99.99%	4,081
Email Spam	219	78	29	87.20%	100.00%	42,324	99.70%	100.00%	21,808
German Credit	26	25	11	100.00%	100.00%	1,722	100.00%	100.00%	1,150
Medical Cover	49	49	11	100.00%	100.00%	5,966	100.00%	100.00%	1,788
Bitcoin Price	155	155	9	100.00%	100.00%	31,956	100.00%	100.00%	7,390

Model Extraction Attacks

- What if…
  - **Setup:**
    - MLaaS APIs do *not* return confidence values $f(\boldsymbol{x})$
    - The adversary can only observe labels
  - **Adaptive Attacks:**
    - The Lowd-Meek attack (~line-search)
    - Re-training approach (~train a model on $(\boldsymbol{x}, f(\boldsymbol{x}))$)
      - Re-training with uniform queries
      - Line-search retraining
      - Adaptive retraining
  - **Results:**
    - on LR models
    - on MLR or MLP



Model Extraction Attacks

- Countermeasures
  - **Rounding confidences:**
    - On LRs, MLRs and MLPs
    - On decision trees: node collision
  - **Differential privacy:**
    - Ugh…
    - It's not designed to prevent extractions
  - **Ensemble methods:**
    - The adversary can approximate the ensemble itself

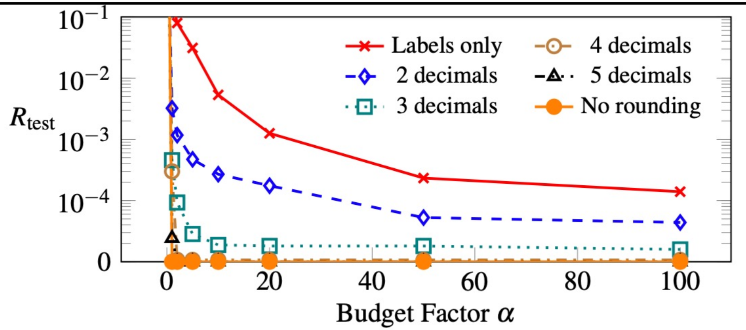


Figure 7: **Effect of rounding on model extraction.** Shows the average test error of equation-solving attacks on softmax models trained on the benchmark suite (Table 3), as we vary the number of significant digits in reported class probabilities. Extraction with no rounding and with class labels only (adaptive retraining) are added for comparison.

Recap

- Privacy Attacks and Defenses
  - Non-ML: Data anonymization
  - Membership inference
    - Threat Model
    - Attacks: Yeom *et al.* and Shokri *et al.*
    - Defensive techniques
  - Model inversion
    - Threat Model
    - Attacks: Fredrikson *et al.* and Carlini *et al.*
    - Defensive techniques
  - Model extraction
    - Threat Model
    - Attacks:
      - Tramer *et al.*
      - Jagielski *et al.*
    - Defensive techiniques

Wait! How Much Would It be Easy/Difficult Then for NNs?

Motivation

- Two different attack objectives in prior work
  - Accuracy vs. Fidelity
  - **Accuracy:** extracted model be *accurate*
  - **Fidelity:** extracted model be *the same*

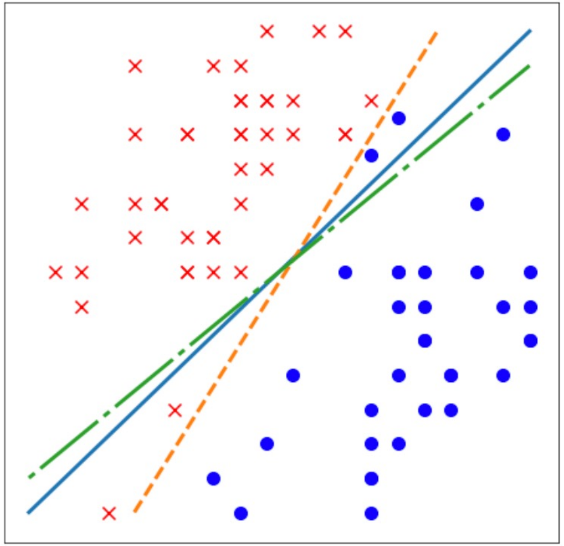


Figure 1: Illustrating fidelity vs. accuracy. The solid blue line is the oracle; functionally equivalent extraction recovers this exactly. The green dash-dot line achieves high fidelity: it matches the oracle on all data points. The orange dashed line achieves perfect accuracy: it classifies all points correctly.

Threat Model – Revisit'ed

- Model extraction attacks
  - **Goal**
    - To learn a new model $\hat{f}$ that closely approximates the target model f
      - Functionally equivalent extraction
      - > Fidelity extraction
      - > Task accuracy extraction
  - **Knowledge**
    - Black-box (typically)
    - It's possible to know aux. information
  - **Capability**
    - Has query access to the victim f (many times) with arbitrary inputs $\boldsymbol{x}$
    - Has computational power to do offline processing of query outputs $f(\boldsymbol{x})$

Threat Model – Revisit'ed

- Model extraction is "hard"
  - **Require exponential # of queries:**
    - To achieve functionally-equivalent extraction, it requires $O(p^k)$ queries
  - **NP-hardness:**
    - Testing if two neural networks are the same is an NP-hard problem
  - **Connection to the learning approaches:**
    - To learn a surrogate model of a NN, it requires $\exp(O(h))$ queries

Model Extraction Attacks

- Learning-based model extractions
  - **Setup:**
    - Adversaries have access to some datasets
    - They use the victim model f as a labeling *oracle*
    - They train a separate model $\hat{f}$ on the oracle outputs
    - **Goal:** To make $\hat{f}$ and f achieve same test-time accuracy
  - **Experimental setup:**
    - **Oracle:** a model trained on 1B Instagram images (SoTA on ImageNet)
    - **Attacker:**
      - Case I: who has 10% (~13k) or 100% of the training samples (1B)
      - Case II: who improves the attack by using semi-supervised techniques (Rot. / MixMatch)

Model Extraction Attacks

- Learning-based model extractions
  - **Results (**+Rot.**):**
    - Oracle (84.2% Top-1 acc. / 97.2% in Top-5)
    - Extracted models show a high accuracy (81- 94%) and fidelity (83- 97%) in Top-5
    - Semi-supervised approaches improve the performance further

Architecture	Data Fraction	ImageNet	WSL	WSL-5	ImageNet + Rot	WSL + Rot	WSL-5 + Rot
Resnet_v2_50	10%	(81.86/82.95)	(82.71/84.18)	(82.97/84.52)	(82.27/84.14)	(82.76/84.73)	(82.84/84.59)
Resnet_v2_200	10%	(83.50/84.96)	(84.81/86.36)	(85.00/86.67)	(85.10/86.29)	(86.17/88.16)	(86.11/87.54)
Resnet_v2_50	100%	(92.45/93.93)	(93.00/94.64)	(93.12/94.87)	N/A	N/A	N/A
Resnet_v2_200	100%	(93.70/95.11)	(94.26/96.24)	(94.21/95.85)	N/A	N/A	N/A

Problem: Non-determinism!

Model Extraction Attacks

- Learning-based model extractions
  - **Sources of non-determinism:**
    - Initialization of model parameters
    - SGD (*random mini-batches)
  - **Prior work on FE extraction attacks:**
    - Milli *et al.*: *gradient queries*
    - Batina *et al.*: *power side-channel*

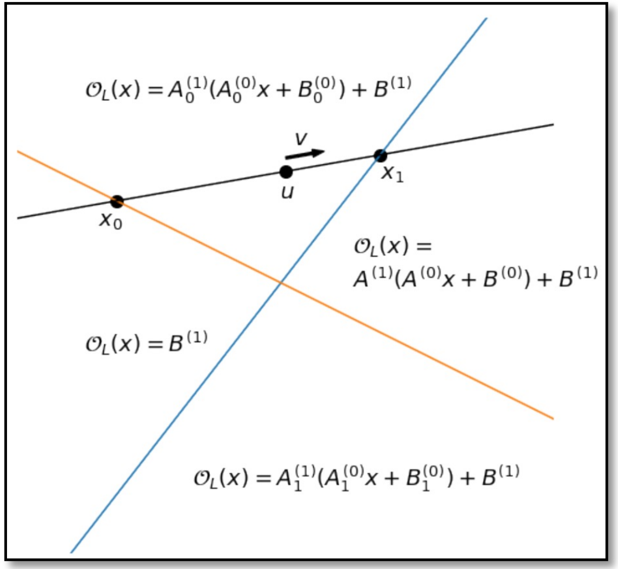
Query Set	Init & SGD	Same SGD	Same Init	Different
Test	93.7%	93.2%	93.1%	93.4%
Adv Ex	73.6%	65.4%	65.3%	67.1%
Uniform	65.7%	60.2%	59.0%	60.2%

Table 4: Impact of non-determinism on extraction fidelity. Even models extracted using the same SGD and initialization randomness as the oracle do not reach 100% fidelity.

Extraction Attacks in Prior Work Are Too Strong!

Model Extraction Attacks

- Proposed attack
  - **Intuition** (ReLU)
    - A standard choice of activation functions
    - It makes neural networks piecewise-linear (let's exploit it)
  - **Attack procedures** (on a 2-layer NN)
    - Critical point search
    - Weight recovery
    - Sign recovery
    - Final layer extraction



Model Extraction Attacks

- Proposed attack
  - **Attack procedures** (on a 2-layer NN)
    - **Critical point search**
    - Weight recovery
    - Sign recovery
    - Final layer extraction

Algorithm 1 Algorithm for 2-linearity testing. Computes the location of the only critical point in a given range or rejects if there is more than one.

Function f, range $[t_1, t_2]$, ε

$m_1 = \frac{f(t_1+\varepsilon)-f(t_1)}{\varepsilon}$ ▷ Gradient at t_1

$m_2 = \frac{f(t_2)-f(t_2-\varepsilon)}{\varepsilon}$ ▷ Gradient at t_2

$y_1 = f(a), y_2 = f(b)$

$x = a + \frac{y_2-y_1-(b-a)m_2}{m_1-m_2}$ ▷ Candidate critical point

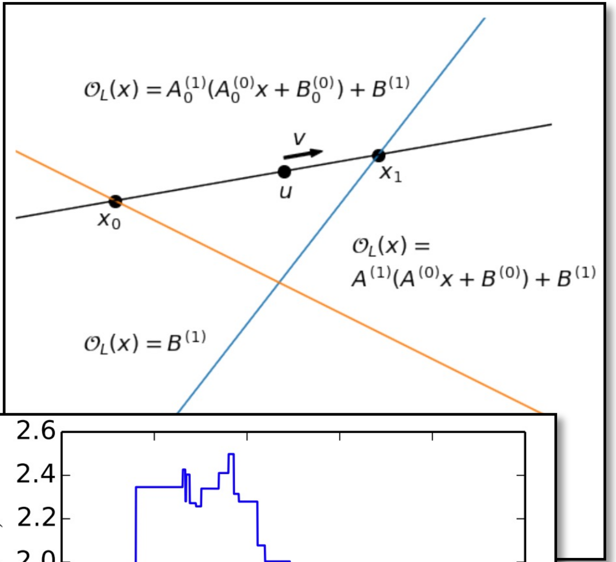
$\hat{y} = y_1 + m_1 \frac{y_2-y_1-(b-a)m_2}{m_1-m_2}$ ▷ Expected value at candidate

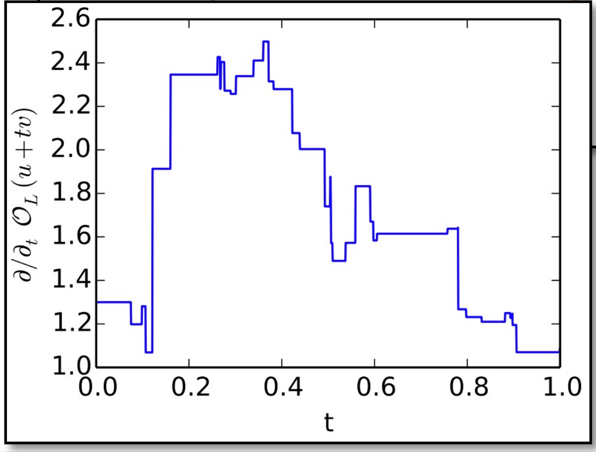
$y = f(x)$ ▷ True value at candidate

if $\hat{y} = y$ **then return** x

else return "More than one critical point"

end if





Model Extraction Attacks

- Proposed attack
  - **Attack procedures** (on a 2-layer NN)
    - Critical point search
    - **Weight recovery**
      - Compute second derivatives
      - Estimate the ratio between two weight vectors w_1, w_2
    - Sign recovery
    - Final layer extraction

Model Extraction Attacks

- Proposed attack
  - **Attack procedures** (on a 2-layer NN)
    - Critical point search
    - Weight recovery
    - **Sign recovery**
    - **Final layer extraction**

Evaluation

- Proposed attacks
  - **Setup:**
    - Datasets: MNIST and CIFAR-10
    - Models: 2-layer NN, 16 – 512 hidden units (~12 – 100k params)
  - **Results:**
    - MNIST:
      - 100% fidelity on the test-set
      - $2^{17.2} - 2^{20.2}$ queries for the 100% fidelity
    - CIFAR-10:
      - 100% fidelity on the test-set for models with < 200k params
      - 99% for the models with > 200k params
      - $2^{17.2} - 2^{20.2}$ queries for the 100% fidelity

Recap

- Privacy Attacks and Defenses
  - Non-ML: Data anonymization
  - Membership inference
    - Threat Model
    - Attacks: Yeom *et al.* and Shokri *et al.*
    - Defensive techniques
  - Model inversion
    - Threat Model
    - Attacks: Fredrikson *et al.* and Carlini *et al.*
    - Defensive techniques
  - Model extraction
    - Threat Model
    - Attacks: Tramer *et al.*, and Jagielski *et al.*
    - Defensive techiniques

Thank You!

Mon/Wed 12:00 – 1:50 pm

Sanghyun Hong

https://secure-ai.systems/courses/MLSec/W22

Oregon State University

SAIL
Secure AI Systems Lab