CS 578: CYBER-SECURITY

PART VI: TRUSTWORTHY ML

Sanghyun Hong

sanghyun.hong@oregonstate.edu

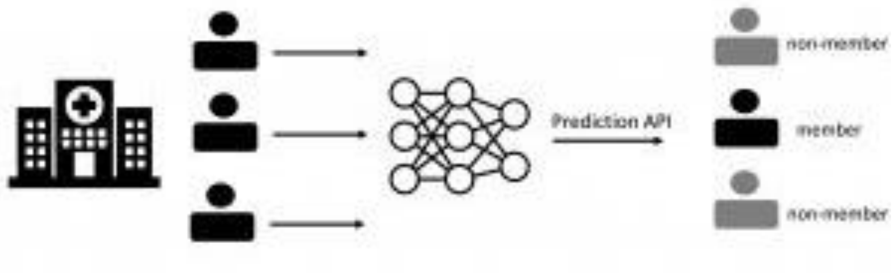
Oregon State University

SAIL
Secure AI Systems Lab

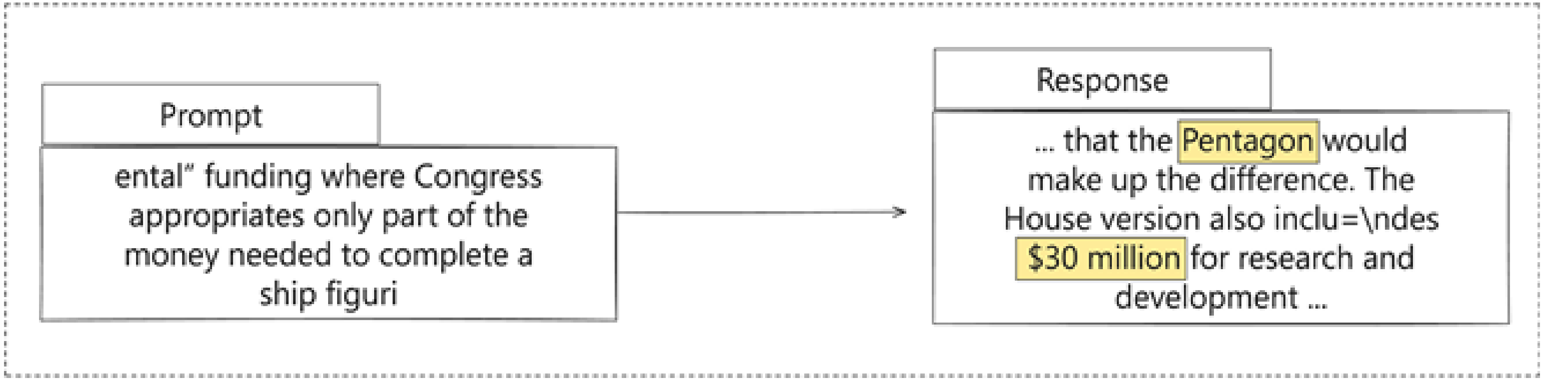
MEMBERSHIP INFERENCE

PRIVACY IN MACHINE LEARNING

- Membership inference attacks

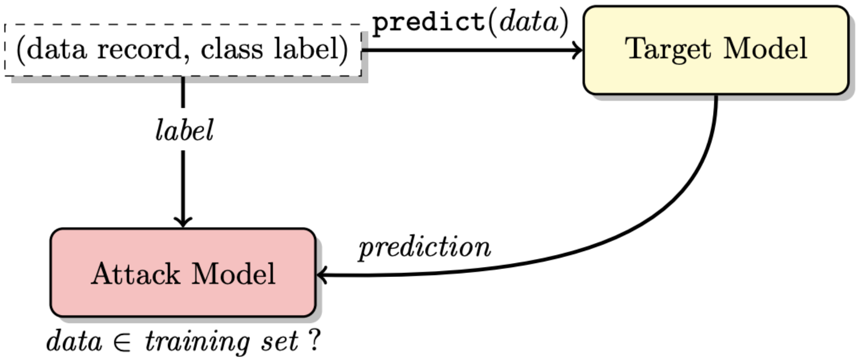


Does the sensitive training set contain a target record?



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

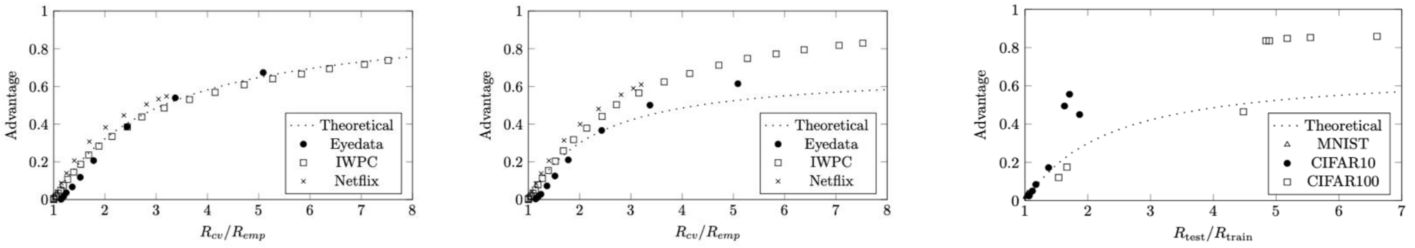
Membership inference attacks

- Yeom *et al.* attack
  - $\mathcal{A}_1$: Bounded loss function
    - Suppose the loss function is bounded on B
    - For $z = (x, y)$
      - The attacker returns 1 with the probability $l(A_S, z)/B$
      - Otherwise, the attacker outputs 0

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

MEMBERSHIP INFERENCE ATTACKS

- Yeom *et al.* attack
  - $\mathcal{A}_1$: Bounded loss function
    - Suppose the loss function is bounded on B
    - For $z = (x, y)$
      - The attacker returns 1 with the probability $l(A_S, z)/B$
      - Otherwise, the attacker outputs 0
    - (Theorem 2) $\mathcal{A}_1$'s advantage is $R_{\text{gen}}(A)/B$



(a) Regression and tree models assuming knowledge of σ_S and $\sigma_\mathcal{D}$.

(b) Regression and tree models assuming knowledge of σ_S only.

(c) Deep CNNs assuming knowledge of average training loss L_S.

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

MEMBERSHIP INFERENCE ATTACKS

- Yeom *et al.* attack
  - $\mathcal{A}_1$: Bounded loss function
    - Suppose the loss function is bounded on B
    - For $z = (x, y)$
      - The attacker returns 1 with the probability $l(A_S, z)/B$
      - Otherwise, the attacker outputs 0

  - $\mathcal{A}_2$: Threshold
    - Suppose the attacker knows
      - The conditional probability density functions of the error
      - $f(\epsilon \mid b = 0)$ and $f(\epsilon \mid b = 1)$
      - such as the avg. loss over the training data (and over the test data)
    - For $z = (x, y)$
      - Let $\epsilon = y - A_S(x)$
      - The attacker outputs $\mathrm{argmax}_{b \in \{0,1\}} f(\epsilon \mid b)$

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

MEMBERSHIP INFERENCE ATTACKS

- Evaluation

	Our work	Shokri et al. [7]
Attack complexity	Makes only one query to the model	Must train hundreds of shadow models
Required knowledge	Average training loss L_S	Ability to train shadow models, e.g., input distribution and type of model
Precision	0.505 (MNIST) 0.694 (CIFAR-10) 0.874 (CIFAR-100)	0.517 (MNIST) 0.72-0.74 (CIFAR-10) $>$ 0.99 (CIFAR-100)
Recall	$>$ 0.99	$>$ 0.99

Table 1: Comparison of our membership inference attack with that presented by Shokri et al. While our attack has slightly lower precision, it requires far less computational resources and background knowledge.

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

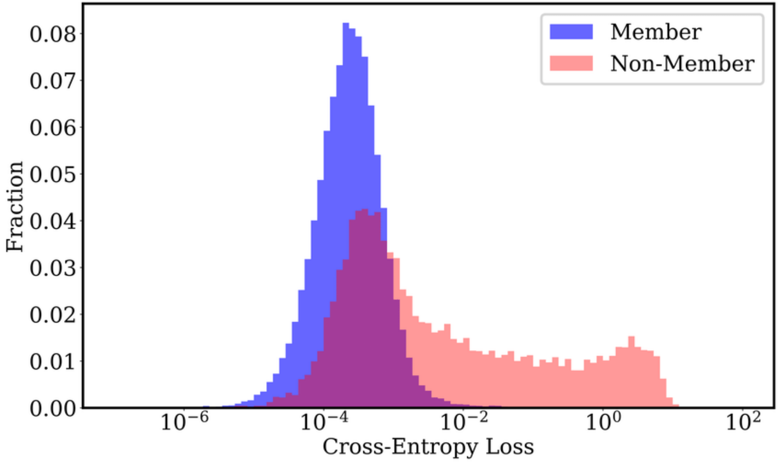
REVISITING YEOM ET AL. ATTACK

- Yeom *et al.* attack
  - $\mathcal{A}_1$: Bounded loss function
    - Suppose the loss function is bounded on B
    - For $z = (x, y)$
      - The attacker returns 1 with the probability $l(A_s, z)/B$
      - Otherwise, the attacker outputs 0
  - $\mathcal{A}_2$: Threshold
    - Suppose the attacker knows
      - The conditional probability density functions of the error
      - $f(\epsilon \mid b = 0)$ and $f(\epsilon \mid b = 1)$
      - such as the avg. loss over the training data (and over the test data)
    - For $z = (x, y)$
      - Let $\epsilon = y - A_s(x)$
      - The attacker outputs $\mathrm{argmax}_{b \in \{0,1\}} f(\epsilon \mid b)$

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

REVISITING YEOM ET AL. ATTACK

- Yeom *et al.* attack
  - $\mathcal{A}_2$: Threshold
    - Suppose the attacker knows
      - The conditional probability density functions of the error
      - $f(\epsilon \mid b = 0)$ and $f(\epsilon \mid b = 1)$
      - such as the avg. loss over the training data (and over the test data)
    - For $z = (x, y)$
      - Let $\epsilon = y - A_s(x)$
      - The attacker outputs $\mathrm{argmax}_{b \in \{0,1\}} f(\epsilon \mid b)$

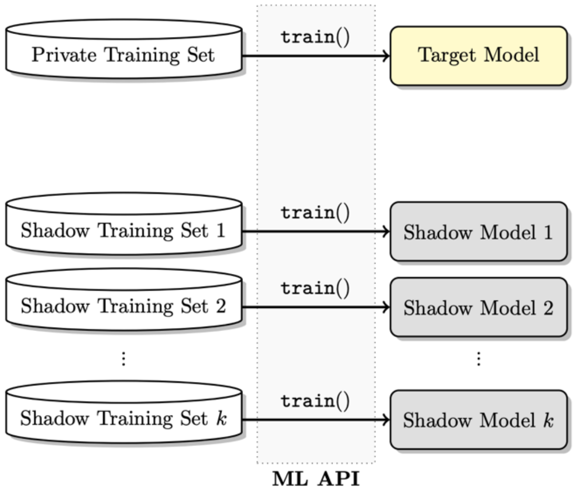


- Challenge:
  - How to compute an optimal threshold?

[1]Song et al., Privacy Risks of Securing Machine Learning Models against Adversarial Examples

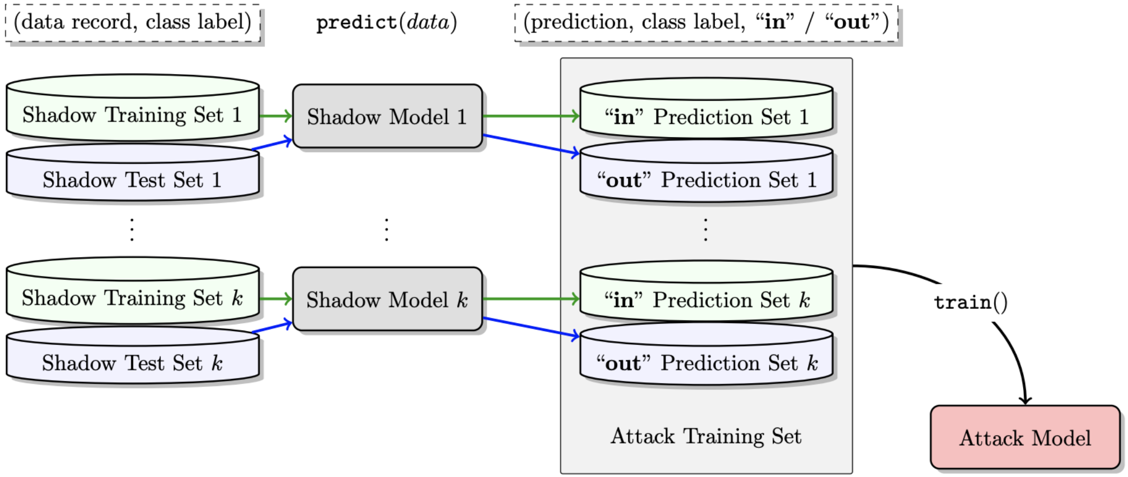
MEMBERSHIP INFERENCE ATTACKS

- Shokri *et al.* attack
  - Key idea: shadow models
    - The attacker has some data samples from D
    - If the attacker trains models with those samples, we know their memberships!
    - If shadow models are trained similarity, we can exploit the membership info.!
  - Attacker's data:
    - Know the labeled records: (x, y)
    - Query them to the target model and collect its predictions: $((x, y), \hat{y})$
  - How to train?
    - Create a train and test split
    - Use the *train* data to train the shadow models



MEMBERSHIP INFERENCE ATTACKS

- Shokri *et al.* attack
  - Attack model
    - Data format $((x, y), \hat{y})$
    - Some of them are **"IN"** the shadow train, otherwise **"OUT"**
    - Combine three info. $(y, \hat{y}, \mathbf{IN})$ or $(y, \hat{y}, \mathbf{OUT})$
    - Make the attack model predict **IN** or **OUT**

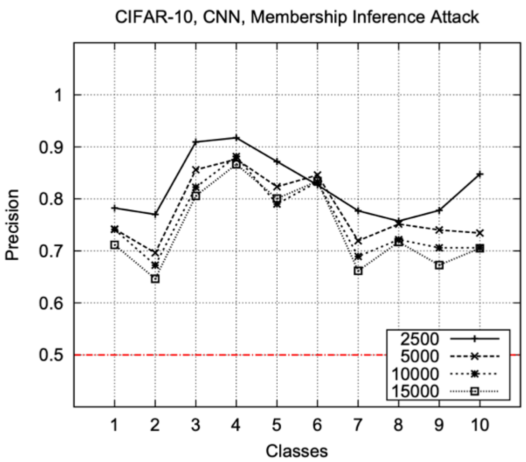


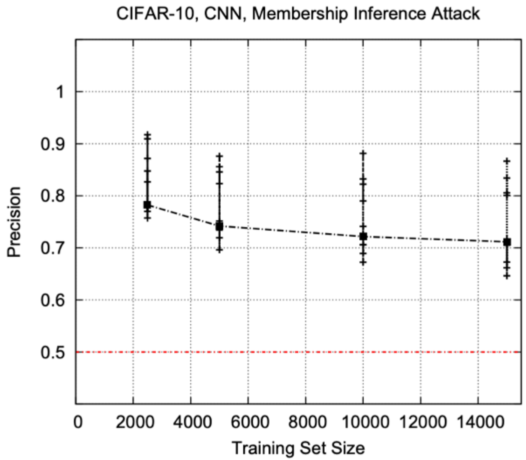
EVALUATION

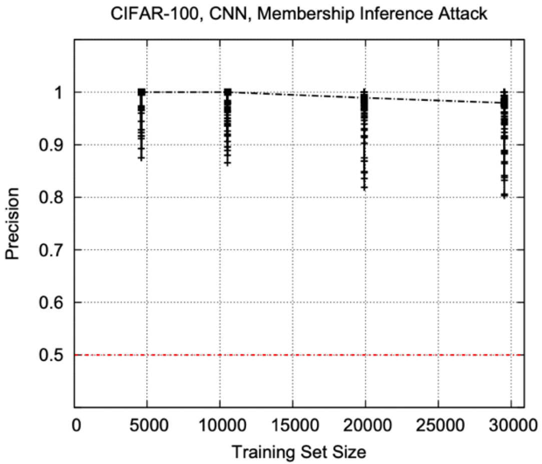
- Setup
  - Datasets:
    - MNIST | CIFAR-10/100
    - Purchases | Locations | Texas-100 | UCI Adult
  - Models
    - MLaaS: Google Prediction API | Amazon ML | NNs
  - MI Attack
    - Shadow models: 20 – 100 models

EVALUATION

- MI Attacks on CIFAR
  - Shadow models: 100
  - Training set (for targets):
    - CIFAR-10: {2.5, 5, 10, 15}k samples
    - CIFAR-100: {4.5, 10, 20, 30}k samples
  - **In-short:** MI attacks work with a pretty reasonable acc.



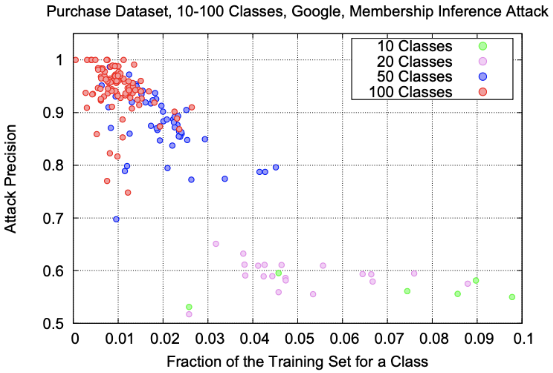


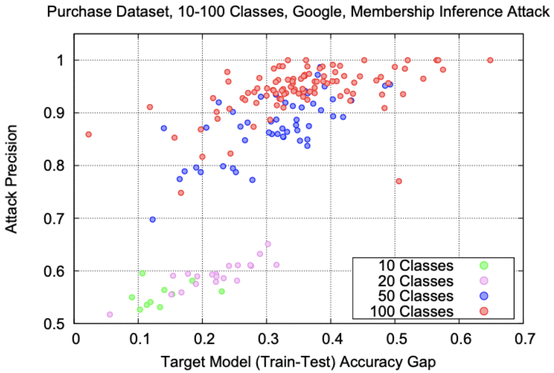


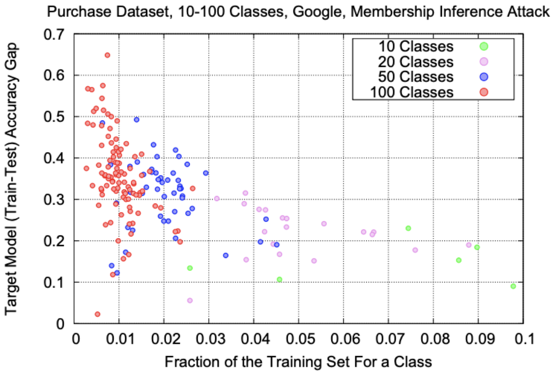
EVALUATION

- MI Attacks w. Different # classes
  - Dataset: Purchase
  - Modification:
    - # Classes: 10 – 100 classes (keep N(D_{tr}) the same)
    - Google Prediction API
  - **In-short:** More supporting data samples in the cl

Dataset	*Training Accuracy*	*Testing Accuracy*	*Attack Precision*
Adult	0.848	0.842	0.503
MNIST	0.984	0.928	0.517
Location	1.000	0.673	0.678
Purchase (2)	0.999	0.984	0.505
Purchase (10)	0.999	0.866	0.550
Purchase (20)	1.000	0.781	0.590
Purchase (50)	1.000	0.693	0.860
Purchase (100)	0.999	0.659	0.935
TX hospital stays	0.668	0.517	0.657



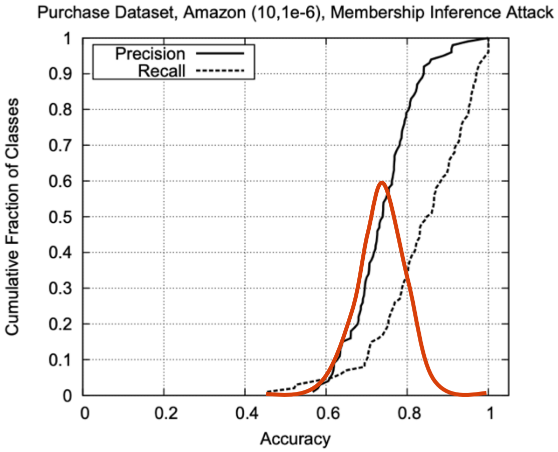


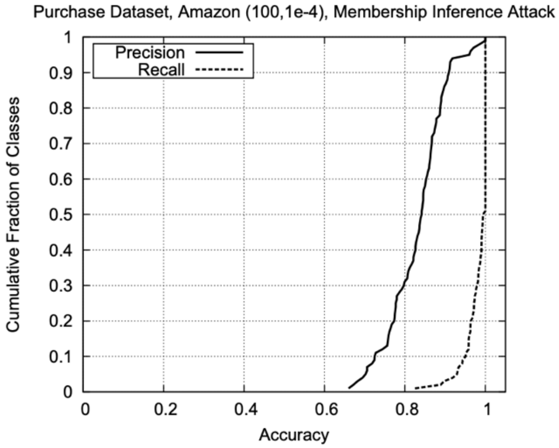


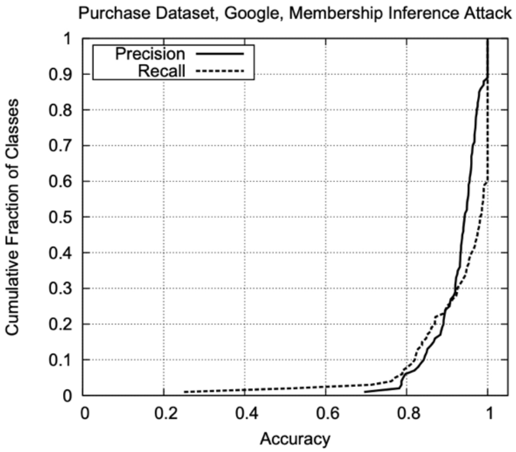
EVALUATION

- MI Attacks w. Different Models
  - Dataset: Purchase-100
  - Models (trained on 10k records):
    - Amazon ML
    - Google's Prediction API
  - **In-short:** across all models, MI attacks work with a pretty reasonable acc.

ML Platform	*Training*	*Test*
Google	0.999	0.656
Amazon (10,1e-6)	0.941	0.468
Amazon (100,1e-4)	1.00	0.504
Neural network	0.830	0.670

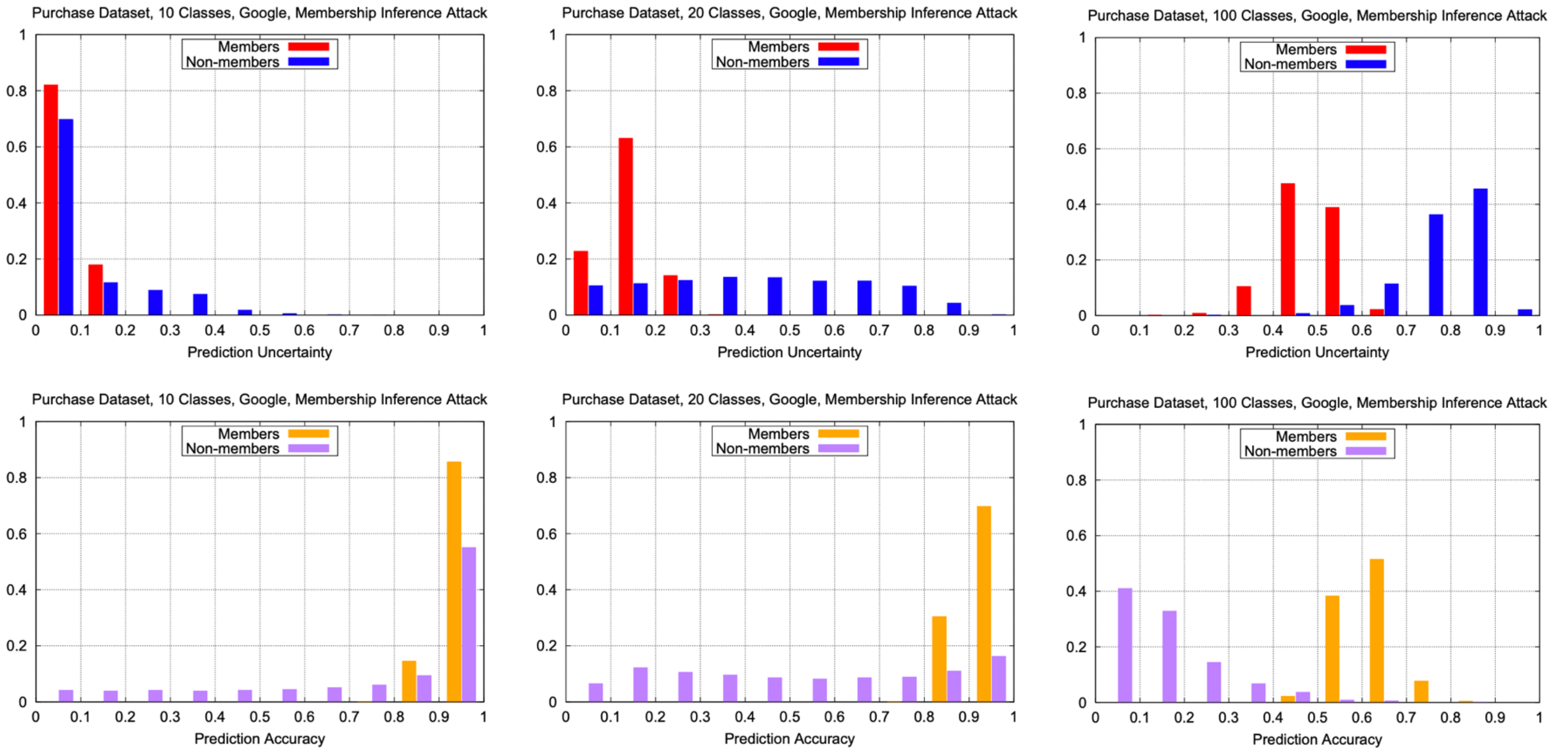






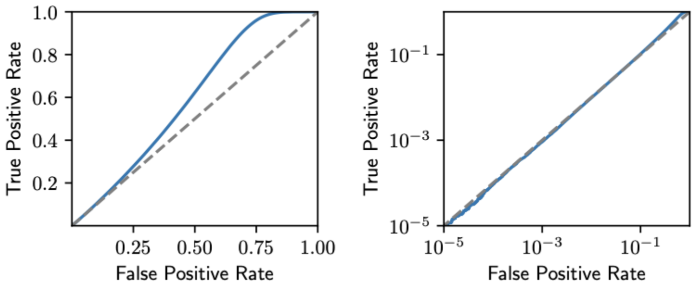
EVALUATION

- MI Attacks, Why Do They Work?



REVISITING YEOM ET AL. AND SHOKRI ET AL. ATTACK

- Metrics for measuring the attack success
  - Problem of existing metrics
    - Symmetric: equal cost to false-positives and false-negatives
    - Average-case metric: often in security, we are interested in a certain subset
  - LOSS attack
    - Metrics:
      - Membership advantage
      - Precision
      - AUROC
    - Problem: perform at random at low-FPR



(a) linear scale (b) log scale

Fig. 2: ROC curve for the LOSS baseline membership inference attack, shown with both linear scaling (left), also and log-log scaling (right) to emphasize the low-FPR regime.

Revisiting Yeom et al. and Shokri et al. Attack

- Metrics for measuring the attack success
  - Problem of existing metrics
    - Symmetric: equal cost to false-positives and false-negatives
    - Average-case metric: often in security, we are interested in a certain subset
  - LOSS attack
    - Metrics: membership advantage or precision
    - Problem: perform at random at low-FPR

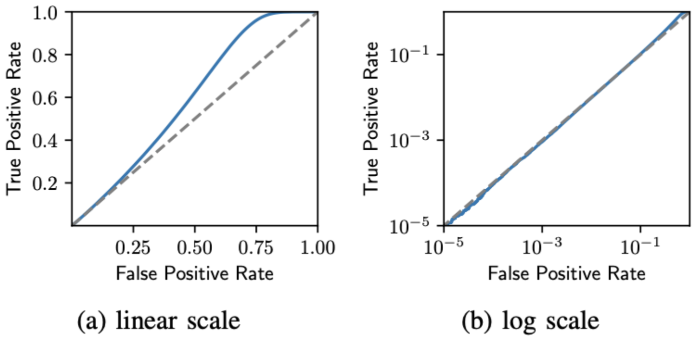


Fig. 2: ROC curve for the LOSS baseline membership inference attack, shown with both linear scaling (left), also and log-log scaling (right) to emphasize the low-FPR regime.

Membership inference attack

- LiRA (The likelihood ratio attack)
  - Per-sample hardness score
    - Not all examples are equal
    - Some samples are easier to fit
    - Some samples have a larger separability
    - It does not matter if it is an inlier or outlier

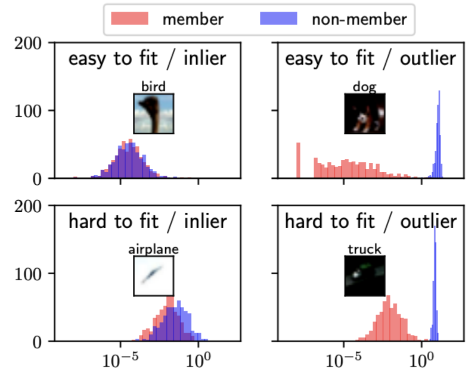


Fig. 3: Some examples are easier to fit than others, and some have a larger separability between their losses when being a member of the training set or not. We train 1024 models on random subsets of CIFAR-10 and plot the losses for four examples when the example is a member of the training set ($\tilde{\mathbb{Q}}_{\text{in}}(x, y)$, in red) or not ($\tilde{\mathbb{Q}}_{\text{out}}(x, y)$, in blue).

MEMBERSHIP INFERENCE ATTACK

- LiRA (The likelihood ratio attack)
  - Per-sample hardness score
    - Not all examples are equal
    - Some samples are easier to fit
    - Some samples have a larger separability
    - It does not matter if it is an inlier or outlier
  - Proposed attack
    - Compute per-sample hardness scores
    - Use parametric modeling

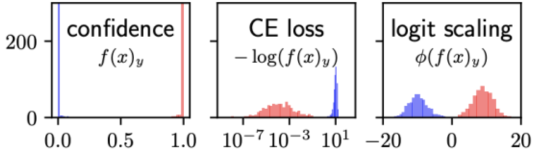


Fig. 4: The model's confidence, or its logarithm (the cross-entropy loss) are not normally distributed. Applying the logit function yields values that are approximately normal.

Algorithm 1 Our online Likelihood Ratio Attack (LiRA). We train shadow models on datasets with and without the target example, estimate mean and variance of the loss distributions, and compute a likelihood ratio test. (In our **offline** variant, we omit lines 5, 6, 10, and 12, and instead return the prediction by estimating a single-tailed distribution, as is shown in Equation (4).)

Require: model f, example (x, y), data distribution $\mathbb{D}$

1: $\text{confs}_{\text{in}} = \{\}$
2: $\text{confs}_{\text{out}} = \{\}$
3: **for** N times **do**
4: $\quad D_{\text{attack}} \xleftarrow{\$} \mathbb{D}$ ▷ *Sample a shadow dataset*
5: $\quad f_{\text{in}} \leftarrow \mathcal{T}(D_{\text{attack}} \cup \{(x, y)\})$ ▷ *train IN model*
6: $\quad \text{confs}_{\text{in}} \leftarrow \text{confs}_{\text{in}} \cup \{\phi(f_{\text{in}}(x)_y)\}$
7: $\quad f_{\text{out}} \leftarrow \mathcal{T}(D_{\text{attack}} \backslash \{(x, y)\})$ ▷ *train OUT model*
8: $\quad \text{confs}_{\text{out}} \leftarrow \text{confs}_{\text{out}} \cup \{\phi(f_{\text{out}}(x)_y)\}$
9: **end for**
10: $\mu_{\text{in}} \leftarrow \texttt{mean}(\text{confs}_{\text{in}})$
11: $\mu_{\text{out}} \leftarrow \texttt{mean}(\text{confs}_{\text{out}})$
12: $\sigma^2_{\text{in}} \leftarrow \texttt{var}(\text{confs}_{\text{in}})$
13: $\sigma^2_{\text{out}} \leftarrow \texttt{var}(\text{confs}_{\text{out}})$
14: $\text{conf}_{\text{obs}} = \phi(f(x)_y)$ ▷ *query target model*
15: **return** $\Lambda = \dfrac{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}}))}{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}}))}$

EVALUATION

- Setup
  - Datasets: CIFAR-10, CIFAR-100, ImageNet and WikiText
  - Models
    - Wide-ResNet (CIFAR-10 and -100)
    - ResNet-50 (ImageNet)
    - GPT-2 small (WikiText)
  - LiRA setup
    - Shadow models: 65 for ImageNet and 256 for others
    - Repeat the attack 10 times
  - Metric
    - TPR at 1% FPR
    - ROC curve

EVALUATION

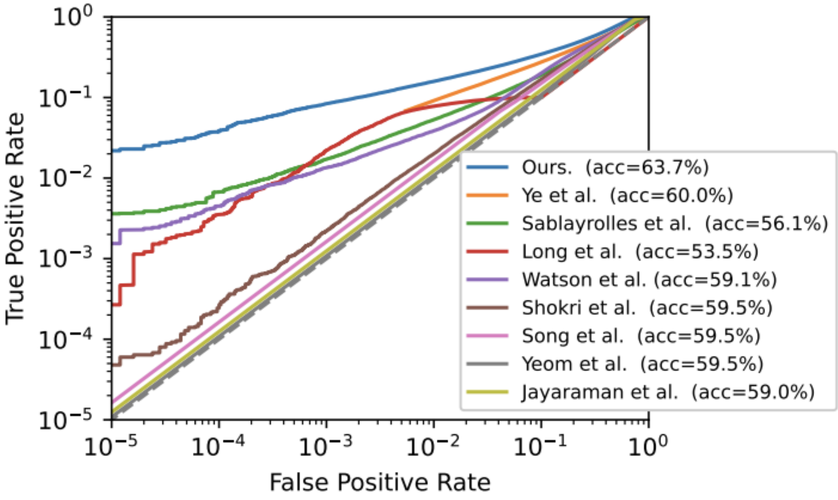
- LiRA (online) attack vs others

Method	shadow models	multiple queries	class hardness	example hardness	TPR @ 0.001% FPR			TPR @ 0.1% FPR			Balanced Accuracy		
					C-10	C-100	WT103	C-10	C-100	WT103	C-10	C-100	WT103
Yeom et al. [70]	○	○	○	○	0.0%	0.0%	0.00%	0.0%	0.0%	0.1%	59.4%	78.0%	50.0%
Shokri et al. [60]	●	○	●	○	0.0%	0.0%	–	0.3%	1.6%	–	59.6%	74.5%	–
Jayaraman et al. [25]	○	●	○	○	0.0%	0.0%	–	0.0%	0.0%	–	59.4%	76.9%	–
Song and Mittal [61]	●	○	●	○	0.0%	0.0%	–	0.1%	1.4%	–	59.5%	77.3%	–
Sablayrolles et al. [56]	●	○	●	●	0.1%	0.8%	0.01%	1.7%	7.4%	1.0%	56.3%	69.1%	**65.7%**
Long et al. [37]	●	○	●	●	0.0%	0.0%	–	2.2%	4.7%	–	53.5%	54.5%	–
Watson et al. [68]	●	○	●	●	0.1%	0.9%	0.02%	1.3%	5.4%	1.1%	59.1%	70.1%	65.4%
Ye et al. [69]	●	○	●	●	-	-	-	-	-	-	60.3%	76.9%	65.5%
Ours	●	●	●	●	**2.2%**	**11.2%**	**0.09%**	**8.4%**	**27.6%**	**1.4%**	**63.8%**	**82.6%**	65.6%

TABLE I: **Comparison of prior membership inference attacks** under the same settings for well-generalizing models on CIFAR-10, CIFAR-100, and WikiText-103 using 256 shadow models. Accuracy is only presented for completeness; we do not believe this is a meaningful metric for evaluating membership inference attacks. Full ROC curves are presented in Appendix A.

EVALUATION

- LiRA (online) attack vs others
  - 10x more successful than the prior attacks at the low-FPR region (0.001 - 0.1 FPR)



EVALUATION

- LiRA (online) attack and the generalization gap
  - Overfitted models tend to vulnerable to the attack
  - There are models with the identical gaps 100x times vulnerable
  - More accurate models are more vulnerable to the attack

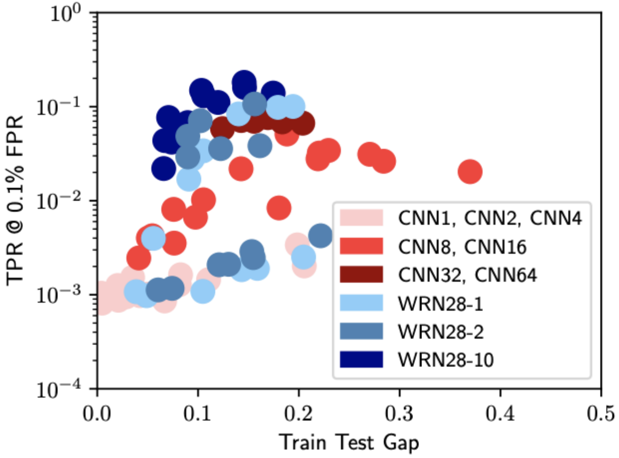


Fig. 7: Attack true-positive rate versus model train-test gap for a variety of CIFAR-10 models.

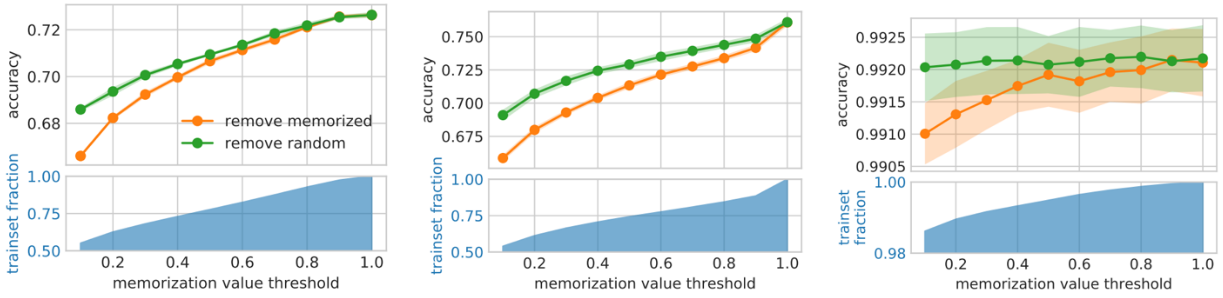
DEFEATING MEMBERSHIP INFERENCE

DEFINITION OF MEMORIZATION

- Feldman and Zhang's
  - New way to quantify the label memorization

$$\mathtt{infl}(\mathcal{A}, S, i, j) := \Pr_{h \leftarrow \mathcal{A}(S)}[h(x'_j) = y'_j] - \Pr_{h \leftarrow \mathcal{A}(S^{\setminus i})}[h(x'_j) = y'_j].$$

    - How much influence a single example on the test-set
    - Memorization is high, when the influence (acc. difference) is high



(a) ImageNet (b) CIFAR-100 (c) MNIST

Figure 2: Effect on the test set accuracy of removing examples with memorization value estimate above a given threshold and the same number of randomly chosen examples. Fraction of the training set remaining after the removal is in the bottom plots. Shaded area in the accuracy represents one standard deviation on 100 (CIFAR-100, MNIST) and 5 (ImageNet) trials.

DEFINITION OF AN ALGORITHM BEING PRIVATE

- A private model (an algorithm)
  - Feldman and Zhang's label memorization

$$\mathtt{infl}(\mathcal{A}, S, i, j) := \mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{A}(S)}[h(x'_j) = y'_j] - \mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{A}(S^{\setminus i})}[h(x'_j) = y'_j].$$

    - How much influence a single example on the test-set
    - Memorization is high, when the influence (acc. difference) is high
  - Property of a private model
    - Given any training instance, its influence on the test acc. is low

DIFFERENTIAL PRIVACY

- ϵ-Differential Privacy
  - A randomized algorithm $M: D \rightarrow R$ with domain D and a range R satisfies ϵ-differential privacy if for any two adjacent inputs $d, d' \in D$ and any subset of outputs $S \subset R$ it holds

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S]$$

DIFFERENTIAL PRIVACY

- ϵ-Differential Privacy
  - A randomized algorithm $M: D \rightarrow R$ with domain D and a range R satisfies ϵ-differential privacy if for any two adjacent inputs $d, d' \in D$ and any subset of outputs $S \subset R$ it holds

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S]$$

- (ϵ, δ)-Differential Privacy

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta$$

  - δ: Represent some catastrophic failure cases [Link, Link]
  - δ < 1/|d|, where |d| is the number of samples in a database

DIFFERENTIAL PRIVACY

- (ϵ, δ)-Differential Privacy **[Conceptually]**

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta$$

  - You have two databases d, d' differ by one item
  - You make the same query M to each and have results $M(d)$ and $M(d')$
  - You ensure the distinguishability between the two under a measure ϵ
    - ϵ is large: those two are distinguishable, less private
    - ϵ is small: the two outputs are similar, more private
  - You also ensure the catastrophic failure probability under δ

DIFFERENTIAL PRIVACY

- (ϵ, δ)-Differential Privacy

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta$$

- Mechanism for (ϵ, δ)-DP: Gaussian noise

$$\mathcal{M}(d) \triangleq f(d) + \mathcal{N}(0, S_f^2 \cdot \sigma^2)$$

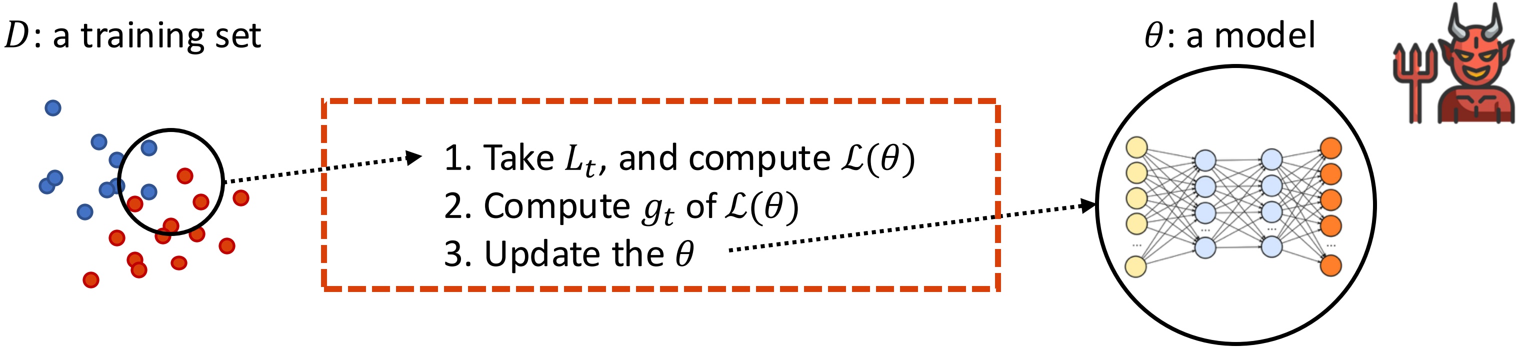
  - $M(d)$: (ϵ, δ)-DP query output on d
  - $f(d)$: non (ϵ, δ)-DP (original) query output on d
  - $N(0, S_f^2 \cdot \sigma^2)$: Gaussian normal distribution with mean 0 and the std. of $S_f^2 \cdot \sigma^2$

Post-hoc: Set the Goal ϵ and Calibrate the noise $S_f^2 \cdot \sigma^2$!

DIFFERENTIAL PRIVACY FOR MACHINE LEARNING

- Revisiting mini-batch stochastic gradient descent (SGD)
    1. At each step t, it takes a mini-batch L_t
    2. Computes the loss $\mathcal{L}(\theta)$ over the samples in L_t, w.r.t. the label y
    3. Computes the gradients g_t of $\mathcal{L}(\theta)$
    4. Update the model parameters θ towards the direction of reducing the loss

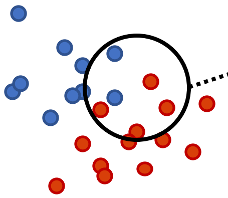
This Process Should Be (ϵ, δ)-DP!



MAKE EACH MINI-BATCH SGD STEP (ϵ, δ)-DP

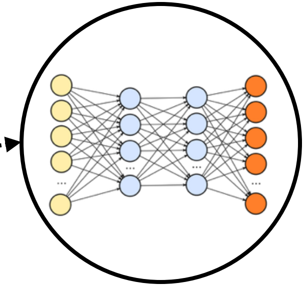
- Mini-batch stochastic gradient descent (SGD)
  1. At each step t, it takes a mini-batch L_t
  2. Computes the loss $\mathcal{L}(\theta)$ over the samples in L_t, w.r.t. the label y
  3. Computes the gradients g_t of $\mathcal{L}(\theta)$
  4. Clip (scale) the gradients to $1/C$, where $C > 1$
  5. Add Gaussian random noise $N(0, \sigma^2 C^2 \mathbf{I})$ to g_t
  6. Update the model parameters θ towards the direction of reducing the loss

D: a training set



1. Take L_t, and compute $\mathcal{L}(\theta)$
2. Compute g_t of $\mathcal{L}(\theta)$
3. Clip g_t and add noise
4. Update the θ

θ: a model



Make the entire training process (ϵ, δ)-DP

- Mini-batch stochastic gradient descent (SGD)
  - SGD iteratively computes the (ϵ, δ)-DP step T times
  - **Problem:** how do we compute the total privacy leakage ϵ_{tot} over T iterations?

- Privacy accounting with moment accountant
  - **Key intuition:** DP has the **composition** property
    - Suppose the two mechanism M_1 and M_2 satisfies (ε_1, δ_1)- and (ε_2, δ_2)-DP the composition of those mechanisms $M_3 = M_2(M_1)$ satisfies (ε_1+ε_2, δ_1+δ_2)-DP
    - If each step t satisfies (ε, δ)-DP, the total SGD process satisfies ($\varepsilon T, \delta T$)-DP
  - **Moment accountant:** tracking the total privacy leakage εT over T iterations

PUTTING ALL TOGETHER

- DP-Stochastic Gradient Descent (DP-SGD)

Algorithm 1 Differentially private SGD (Outline) — // we train a model θ with the privacy budget ε_{budget}

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** — // iterate over T mini-batches

Take a random sample L_t with sampling probability L/N

Compute gradient — // compute the gradient

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

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$ — // clip the magnitude of the gradients

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$ — // add Gaussian random noise to the gradients

Descent

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

$\boldsymbol{\varepsilon}, \boldsymbol{\delta} \leftarrow$ compute the privacy cost (leakage) so far — // compute the privacy cost (leakage) up to t iterations

If $\boldsymbol{\varepsilon} > \boldsymbol{\varepsilon_{buget}}$: then break; — // if the cost is over the budget, then stop training

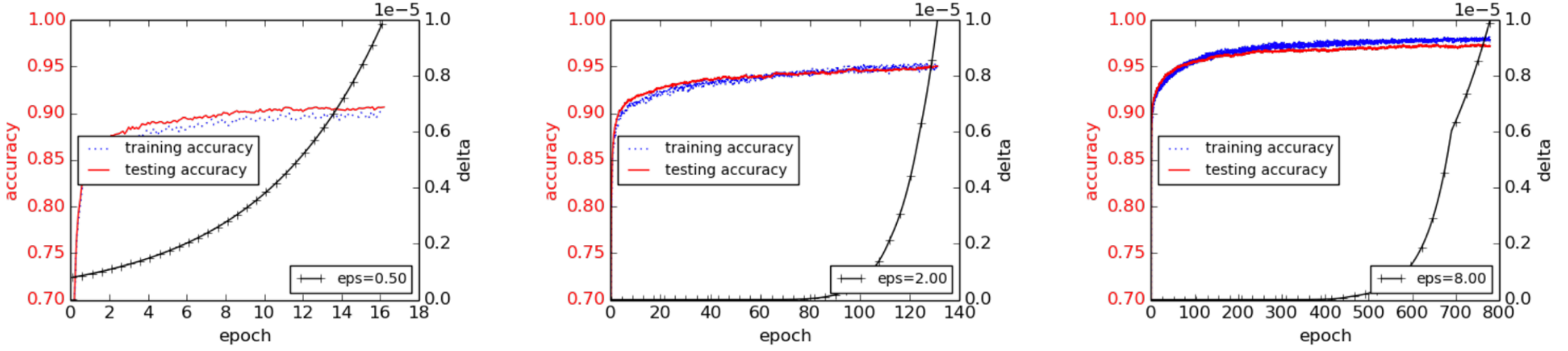
Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

EVALUATION

- Setup
  - Datasets: MNIST | CIFAR-10/100
  - Models:
    - MNIST: 2-layer feedforward NN on 60-dim. PCA projected inputs
    - CIFAR-10/100: A CNN with 2 conv. layers and 2 fully-connected layers
  - Metrics:
    - Classification accuracy
    - Privacy cost (ε_{budget})

EVALUATION

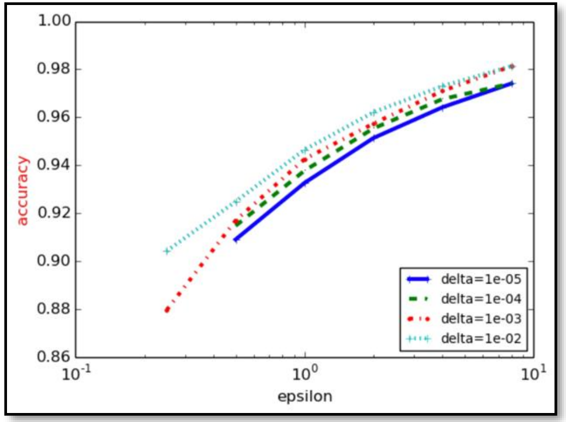
- Impact of Noise
  - Dataset, Models: MNIST, 2-layer feedforward NN
  - Setup: 60-dim PCA projected inputs | Clipping threshold ($\mathbf{C}$): 4 | Noise (σ): 8, 4, 2 (from the left)
  - **Summary:**
    - On MNIST, DP-SGD offers reasonable acc. under various privacy costs (**clean:** 98.3%)
    - The accuracy of private models decreases as we decrease the privacy cost



(1) Large noise (2) Medium noise (3) Small noise

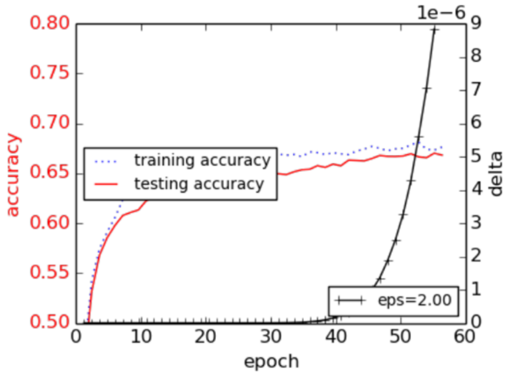
EVALUATION

- Impact of Noise
  - Dataset, Models: MNIST, 2-layer feedforward NN
  - Setup: 60-dim PCA projected inputs | Clipping threshold ($\mathbf{C}$): 4 | Noise (σ): 8, 4, 2 (from the left)
  - **Summary:**
    - On MNIST, DP-SGD offers reasonable acc. under various privacy costs (**clean:** 98.3%)
    - The accuracy of private models decreases as we decrease the privacy cost

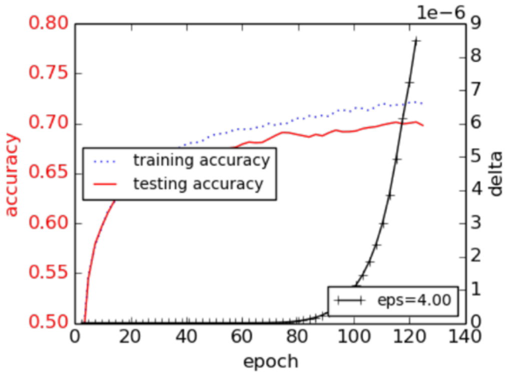


Evaluation

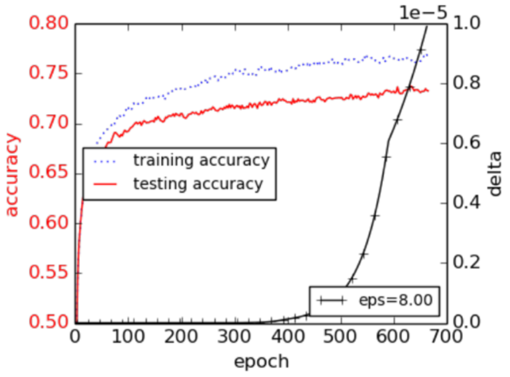
- Impact of Noise
  - Dataset, Models: CIFAR-10, CNN
  - Setup: Clipping threshold ($\mathbf{C}$): 3 | Noise (σ): 6
  - **Summary:**
    - On CIFAR-10, DP-SGD offers reasonable acc. under various privacy costs (**clean:** 80%)
    - The accuracy of private models decreases as we decrease the privacy cost



(1) $\varepsilon = 2$



(2) $\varepsilon = 4$



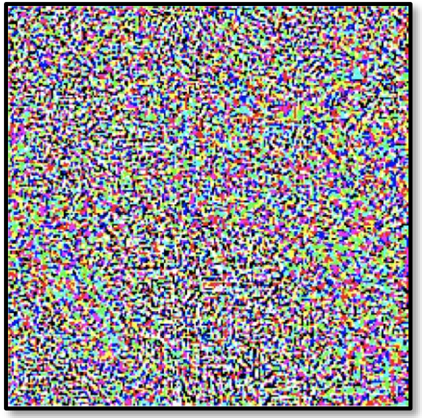
(3) $\varepsilon = 8$

DEFEATING ADVERSARIAL EXAMPLES

ADVERSARIAL EXAMPLES ARE THE WORST-CASE NOISE

- 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**

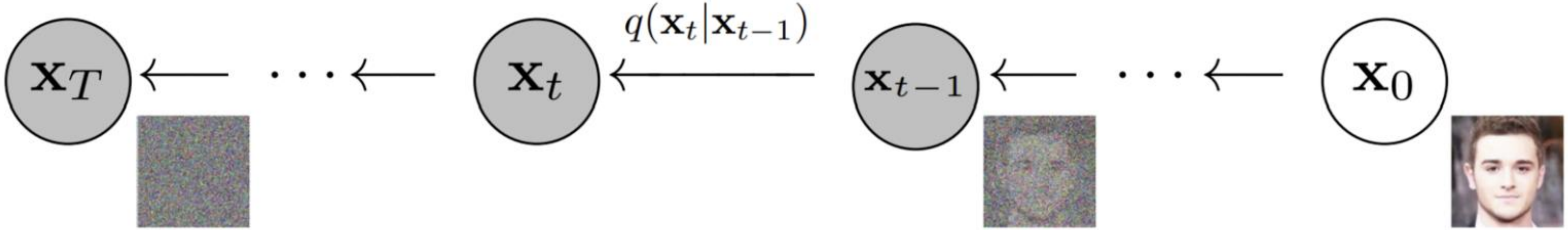
Human-imperceptible Noise

Prediction: **Gibbon**

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

DENOISING DIFFUSION MODELS

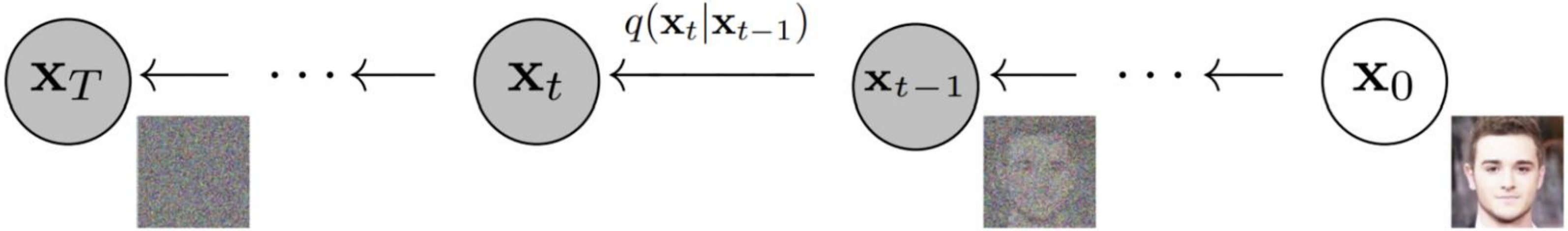
- Denoising diffusion probabilistic models (DDPMs)
  - Generative models trained to gradually denoise the data
  - The *diffusion* process transforms an image x to the purely random noise



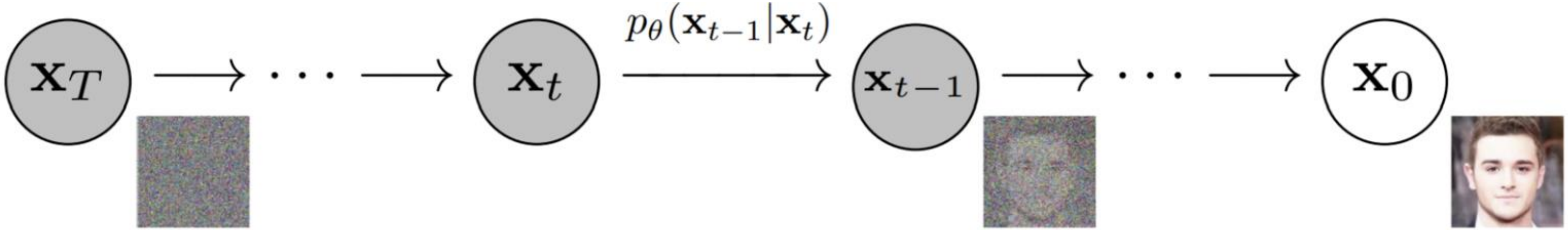
  - Given an image x, the model samples a noisy image: $x_t := \sqrt{\alpha_t} \cdot x + \sqrt{1-\alpha_t} \cdot \mathcal{N}(0, \mathbf{I})$
    α is a constant derived from t and determines the amount of noise to be added

DENOISING DIFFUSION MODELS

- Denoising diffusion probabilistic models (DDPMs)
  - Generative models trained to gradually denoise the data
  - The *diffusion* process transforms an image x to the purely random noise



  - The *reverse* process synthesizes x from random Gaussian noise



WE HAVE PRE-TRAINED DENOISERS

- Denoising diffusion probabilistic models (DDPMs)
  - Generative models trained to gradually denoise the data
  - The *diffusion* process transforms an image x to the purely random noise
  - The *reverse* process synthesizes x from random Gaussian noise

- Use DDPMs as a denoiser $D_\theta: R^d \rightarrow R^d$
  - *One-shot* denoising: apply the diffusion model once for a fixed noise level
  - *Multi-step* denoising: apply the diffusion process multiple times

Provable guarantee against input perturbations

- Practical algorithms for prediction and certification

Algorithm 2 Randomized smoothing (Cohen et al., 2019)

$\textsc{Predict}(x, \sigma, N, \eta)$:
  $\texttt{counts} \leftarrow \mathbf{0}$
  for $i \in \{1, 2, \ldots, N\}$ **do**
    $y \leftarrow \textsc{NoiseAndClassify}(x, \sigma)$
    $\texttt{counts}[y] \leftarrow \texttt{counts}[y] + 1$
  $\hat{y}_A, \hat{y}_B \leftarrow$ top two labels in $\texttt{counts}$
  $n_A, n_B \leftarrow \texttt{counts}[\hat{y}_A], \texttt{counts}[\hat{y}_B]$
  if $\textsc{BinomPTest}(n_A, n_A + n_B, 1/2) \leq \eta$ **then**
    return $\hat{y}_A$
  else
    return $\texttt{Abstain}$

Guarantee the probability of *PREDICT* returning a class other than $g(x)$ is α

Algorithm 1 Noise, denoise, classify

$\textsc{NoiseAndClassify}(x, \sigma)$:
  $t^\star, \alpha_{t^\star} \leftarrow \textsc{GetTimestep}(\sigma)$
  $x_{t^\star} \leftarrow \sqrt{\alpha_{t^\star}}(x + \mathcal{N}(0, \sigma^2 \mathbf{I}))$
  $\hat{x} \leftarrow \texttt{denoise}(x_{t^\star}; t^\star)$
  $y \leftarrow f_{\text{clf}}(\hat{x})$
  return y

$\textsc{GetTimestep}(\sigma)$:
  $t^\star \leftarrow$ find t s.t. $\frac{1-\alpha_t}{\alpha_t} = \sigma^2$
  return $t^\star, \alpha_{t^\star}$

EVALUATION

- Setup
  - Data: CIFAR-10 and ImageNet-21k
  - Model: Wide-ResNet-28-10 (white-box)
  - Denoisers: DDPMs

- Measure
  - Certified test-set accuracy under a radius R with a confidence of α
  - Under various smoothing factor ε (std. of Gaussian noise used)

EVALUATION

- Certified accuracy vs. prior work (ImageNet-21k)
  - DDPM denoisers offer the highest certified accuracy compared to the prior work
  - To achieve the highest accuracy, one can use this off-the-shelf model w/o training

			Certified Accuracy at ε (%)				
Method	Off-the-shelf	Extra data	0.5	1.0	1.5	2.0	3.0
PixelDP (Lecuyer et al., 2019)	○	✗	(33.0) 16.0	-	-		
RS (Cohen et al., 2019)	○	✗	(67.0) 49.0	(57.0) 37.0	(57.0) 29.0	(44.0) 19.0	(44.0) 12.0
SmoothAdv (Salman et al., 2019)	○	✗	(65.0) 56.0	(54.0) 43.0	(54.0) 37.0	(40.0) 27.0	(40.0) 20.0
Consistency (Jeong & Shin, 2020)	○	✗	(55.0) 50.0	(55.0) 44.0	(55.0) 34.0	(41.0) 24.0	(41.0) 17.0
MACER (Zhai et al., 2020)	○	✗	(68.0) 57.0	(64.0) 43.0	(64.0) 31.0	(48.0) 25.0	(48.0) 14.0
Boosting (Horváth et al., 2022a)	○	✗	(65.6) 57.0	(57.0) 44.6	(57.0) **38.4**	(44.6) **28.6**	(38.6) **21.2**
DRT (Yang et al., 2021)	○	✗	(52.2) 46.8	(55.2) 44.4	(49.8) **39.8**	(49.8) **30.4**	(49.8) **23.4**
SmoothMix (Jeong et al., 2021)	○	✗	(55.0) 50.0	(55.0) 43.0	(55.0) **38.0**	(40.0) 26.0	(40.0) 20.0
ACES (Horváth et al., 2022b)	◐	✗	(63.8) 54.0	(57.2) 42.2	(55.6) 35.6	(39.8) 25.6	(44.0) 19.8
Denoised (Salman et al., 2020)	◐	✗	(60.0) 33.0	(38.0) 14.0	(38.0) 6.0	-	-
Lee (Lee, 2021)	●	✗	41.0	24.0	11.0	-	-
Ours	●	✓	(82.8) **71.1**	(77.1) **54.3**	(77.1) **38.1**	(60.0) **29.5**	(60.0) 13.1

EVALUATION

- One-shot vs. multi-step denoising (ImageNet-21k)
  - One-shot denoising offers more faithful results
  - Multi-step denoising destroys the information about the original image

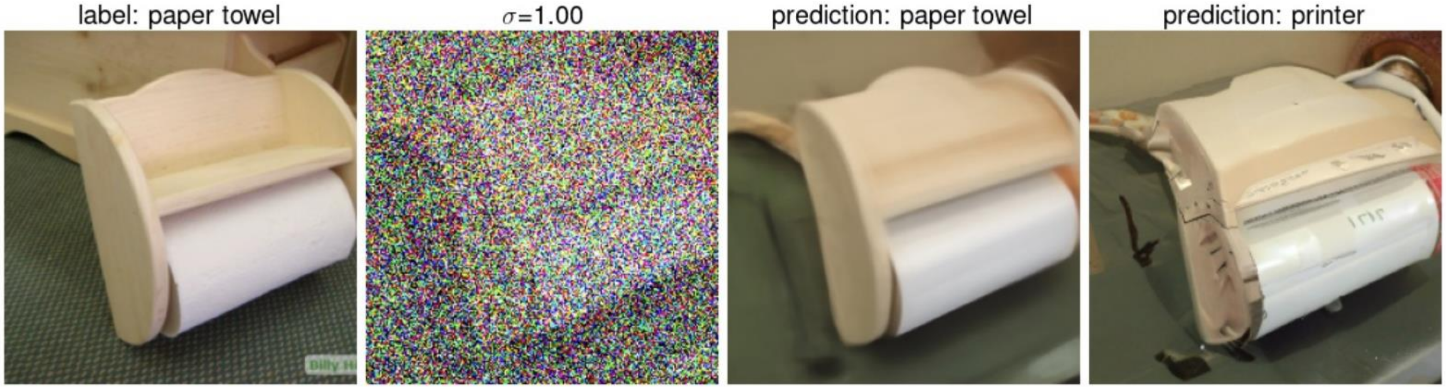
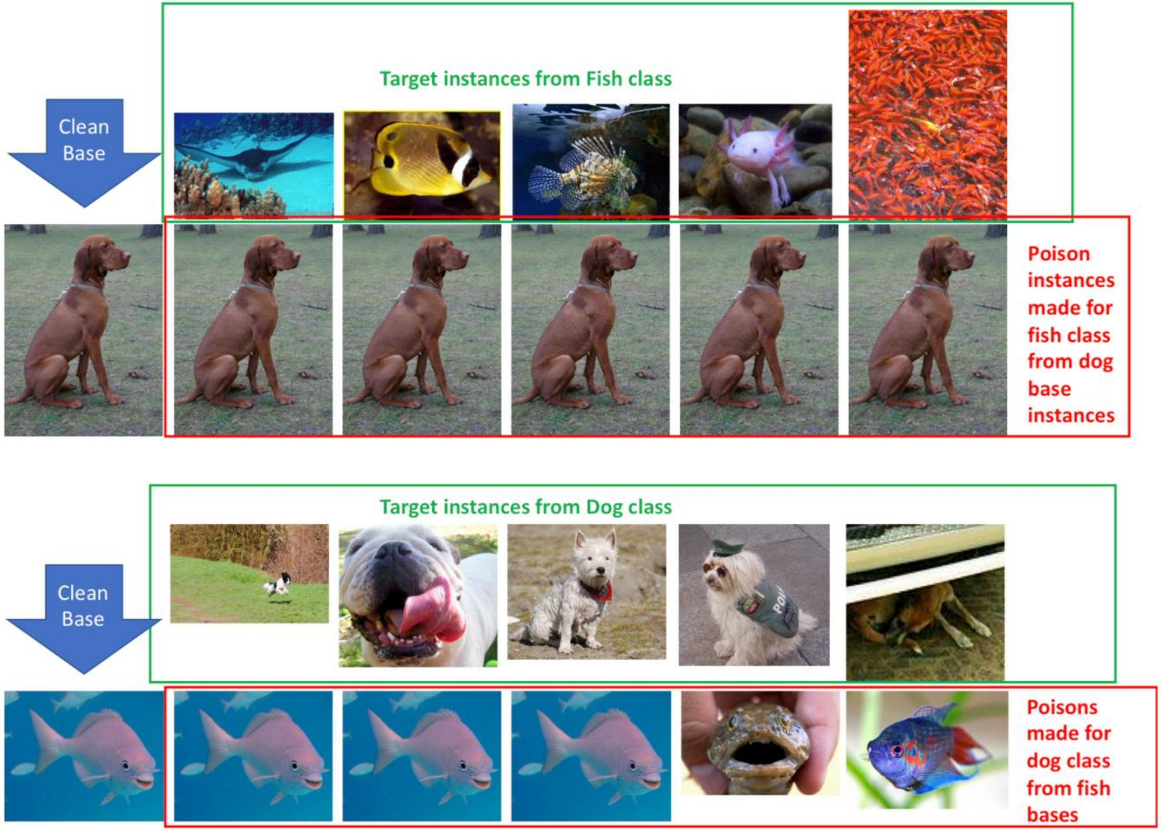


Figure 3: Intuitive examples for why multi-step denoised images are less recognized by the classifier. From left to right: clean images, noisy images with $\sigma = 1.0$, one-step denoised images, multi-step denoised images. For the denoised images, we show the prediction by the pretrained BEiT model.

CAN WE USE DENOISING MODELS TO DEFEAT CLEAN-LABEL POISONING?



PROVABLE GUARANTEE AGAINST DATA POISONING

- Practical algorithms for prediction and certification

Pseudocode Predict and certify (Cohen et al., 2019)

fn PREDICT($f, \sigma, D, x, m, \alpha$)
  counts $\leftarrow$ NDTCLASSIFY(f, σ, D, x, m_0)
  $\hat{c_A}, \hat{c_B} \leftarrow$ top two predictions in counts
  $n_A, n_b \leftarrow$ counts[$\hat{c_A}$], counts[$\hat{c_B}$]
  if BINOMPVAL($n_A, n_A + n_B, 0.5 \leq \alpha$) **ret** $\hat{c_A}$
  else ret ABSTAIN

fn CERTIFY($f, \sigma, D, x, m_0, m, \alpha$)
  counts0 $\leftarrow$ NDTCLASSIFY(f, σ, D, x, m_0)
  $\hat{c_A} \leftarrow$ top predictions in counts0
  counts $\leftarrow$ NDTCLASSIFY(f, σ, D, x, m)
  $p_A \leftarrow$ LOWERCFBOUND(counts[$\hat{c_A}$], m, $1 - \alpha$)
  if $p_A > 1/2$ **ret** $\hat{c_A}$ and radius $\sigma\Phi^{-1}(p_A)$
  else ret ABSTAIN

Guarantee the probability of $P\mathit{REDICT}$ returning a class other than $f(x)$ is α

Pseudocode Noise, denoise, train, and classify

fn NDTCLASSIFY(f, σ, D, x, n)
  counts $\leftarrow$ **0**
  for $i \in \{1, 2, ..., n\}$ **do**
    $t^*, \alpha_{t^*} \leftarrow$ GETTIMESTEP(σ)
    $\hat{D} \leftarrow$ NOISEANDDENOISE($D, \alpha_{t^*}; t^*$)
    $\hat{f}_\theta \leftarrow$ TRAIN($\hat{D}, f$)
    counts[$\hat{f}_\theta(x)$] $\leftarrow$ counts[$\hat{f}_\theta(x)$] + 1
  ret counts

fn GETTIMESTEP(σ)
  $t^* \leftarrow$ find t s.t. $\frac{1-\alpha_t}{\alpha_t} = \sigma^2$
  ret t^*, α_{t^*}

PROVABLE GUARANTEE AGAINST DATA POISONING

- Performance comparison to Gaussian "noising"
  - Our approach achieves a guarantee comparable to existing ones
  - The time it takes to achieve that guarantee is 20x less

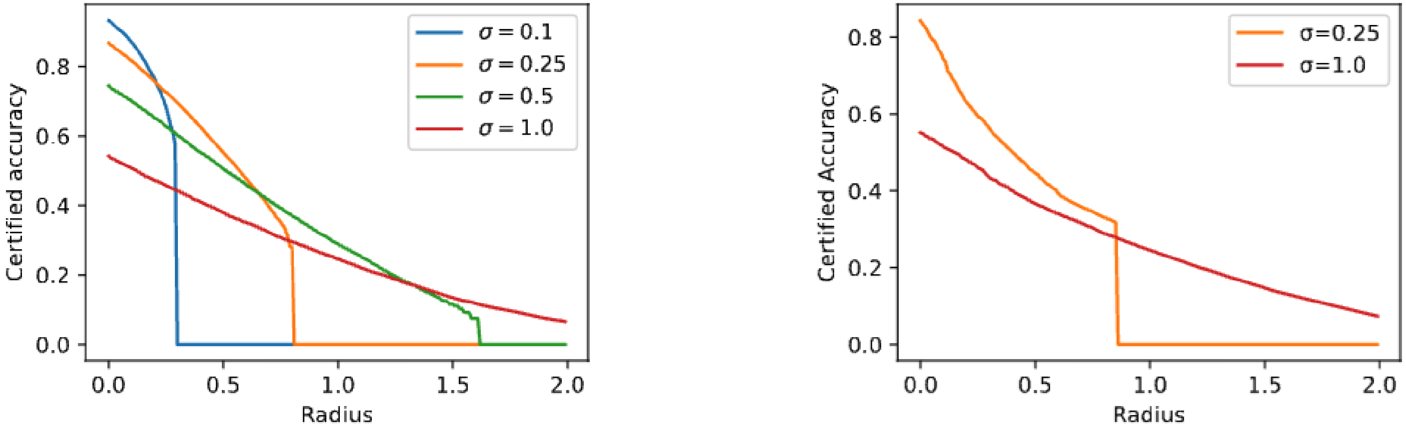


Figure 1: **Certified radius and accuracy** attained by *denoising* the CIFAR10 training data with varying σ values in {0.1, 0.25, 0.5, 1.0} (left) and by adding *Gaussian random noise* to the training data with σ values in {0.25, 1.0} (right).

EVALUATION

- Defeating clean-label poisoning attacks
  - The results are from the transfer-learning scenarios (one-shot kill attacks)
  - Our approach completely renders these attacks ineffective

Table 1: **Defense effectiveness in transfer-learning scenarios (CIFAR10).** We measure the clean accuracy and attack success of the models trained on the denoised training set. Each cell shows the accuracy in the parentheses and the attack success outside. Note that † indicates the runs with σ=0.0, the same as our baseline that trains models without any denoising.

		Our defense against ℓ_2 attacks at σ (%)					Our defense against ℓ_∞ attacks at σ (%)				
Poisoning attacks	**Knowledge**	**†0.0**	**0.1**	**0.25**	**0.5**	**1.0**	**†0.0**	**0.1**	**0.25**	**0.5**	**1.0**
Poison Frog!	WB	(93.6) 99.0	(93.3) 0.0	(91.8) 1.0	(84.8) 0.0	(79.9) 1.0	(93.6) 68.8	(93.3) 0.0	(92.7) 0.0	(90.8) 0.0	(87.4) 0.0
Convex Polytope	WB	(93.7) 16.2	(93.2) 0.0	(91.7) 0.0	(86.6) 0.0	(77.0) 0.0	(93.7) 12.2	(93.3) 0.0	(92.7) 0.0	(90.8) 1.0	(87.5) 0.0
Bullseye Polytope	WB	(93.5) 100	(93.3) 4.0	(92.6) 0.0	(87.5) 0.0	(79.2) 1.0	(93.5) 100	(93.3) 0.0	(92.7) 0.0	(90.8) 1.0	(87.5) 0.0
Label-consistent Backdoor	WB			-			(93.2) 1.0	(93.3) 0.0	(92.6) 0.0	(90.8) 1.0	(87.5) 0.0
Hidden Trigger Backdoor	WB			-			(93.4) 7.0	(93.3) 0.0	(92.6) 0.0	(90.8) 0.0	(87.5) 0.0
Poison Frog!	BB	(91.6) 10.0	(91.2) 0.0	(89.6) 0.5	(82.9) 0.0	(77.8) 2.0	(91.7) 2.5	(91.3) 0.0	(90.3) 0.0	(88.8) 0.5	(86.2) 1.0
Convex Polytope	BB	(91.7) 3.0	(91.0) 0.0	(89.5) 0.0	(84.6) 0.5	(73.6) 1.0	(91.8) 2.5	(91.3) 0.0	(90.3) 0.0	(88.8) 0.5	(86.2) 1.0
Bullseye Polytope	BB	(91.6) 9.0	(91.3) 0.0	(90.3) 0.0	(85.5) 0.0	(76.3) 1.0	(91.6) 8.0	(91.3) 0.0	(90.3) 0.0	(88.8) 0.5	(86.2) 0.5
Label-consistent Backdoor	BB			-			(91.5) 1.0	(91.3) 0.0	(90.3) 0.0	(88.8) 0.0	(86.2) 1.5
Hidden Trigger Backdoor	BB			-			(91.6) 4.0	(91.2) 1.0	(90.3) 1.0	(89.3) 1.5	(86.3) 1.5

EVALUATION

- Defeating clean-label poisoning attacks
  - The results are from training from scratch scenarios
  - Our approach completely renders these attacks ineffective

Table 2: **Defense effectiveness in training from-scratch scenarios (CIFAR10).** We measure the accuracy and attack success of the models trained on the denoised training set. Each cell shows the accuracy in the parentheses and the attack success outside. Note that † indicates the runs with σ=0.0, the same as our baseline that trains models without any denoising. We use an ensemble of four models, and WB and BB stand for the white-box and the black-box attacks, respectively.

		Our defense against ℓ_2 attacks at σ (%)					Our defense against ℓ_∞ attacks at σ (%)				
Poisoning attacks	**Knowledge**	**†0.0**	**0.1**	**0.25**	**0.5**	**1.0**	**†0.0**	**0.1**	**0.25**	**0.5**	**1.0**
Witches' Brew	WB	(92.2) 71.0	(86.4) 54.0	(72.3) 10.0	(46.7) 11.0	(42.5) 10.0	(92.3) 65.0	(86.5) 9.0	(71.9) 3.0	(46.0) 9.0	(41.3) 7.0
Sleeper Agent Backdoor	WB	(92.4) 40.5	(84.4) 66.5	(71.8) 19.5	(46.8) 13.0	(39.9) 10.5	(92.4) 35.0	(86.1) 17.0	(73.0) 8.0	(47.0) 9.5	(39.7) 10.0
Witches' Brew	BB	(90.1) 45.5	(85.9) 28.0	(75.5) 4.0	(58.8) 7.0	(49.0) 10.0	(90.0) 33.5	(85.8) 3.5	(75.5) 2.5	(58.8) 6.0	(48.7) 6.5
Sleeper Agent Backdoor	BB	(90.0) 39.5	(85.0) 44.5	(75.1) 14.5	(58.6) 9.5	(49.0) 8.0	(90.0) 18.5	(85.6) 11.5	(75.5) 7.0	(58.8) 8.0	(48.4) 8.0

Thank You!

Sanghyun Hong

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

Oregon State University

SAIL
Secure AI Systems Lab