Auditing and Membership Inference

Eric Xie, Yagnik Panguluri, Anders Gyllenhoff, Caroline Gihlstorf

Membership Inference Attacks Against Machine Learning Models Shokri et al. (2016)

Privacy in Machine Learning

Inference about members of the population:

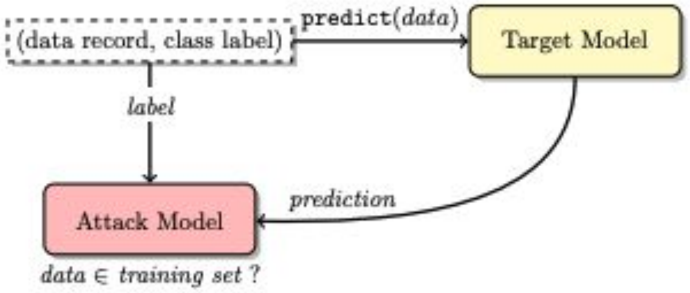
- “The model should reveal no more about the input to which it was applied than would have been known about this input without applying the model”
- Generalized models naturally uncover correlations for populations → unavoidable privacy breach

Inference about members of the training dataset

- Closely aligned with the motivations of differential privacy

What are Membership Inference Attacks

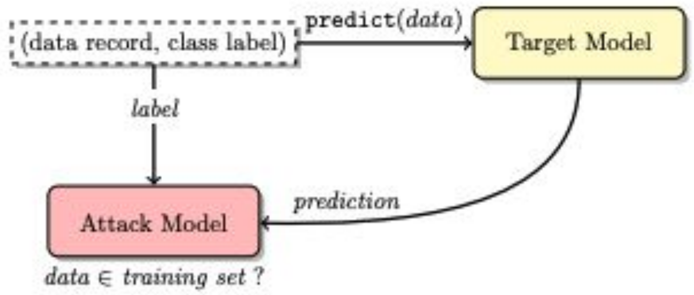
Given a data record and black-box access to a trained model, can an adversary determine if that record was part of the model's training dataset?



What are Membership Inference Attacks

Given a data record and black-box access to a trained model, can an adversary determine if that record was part of the model's training dataset?

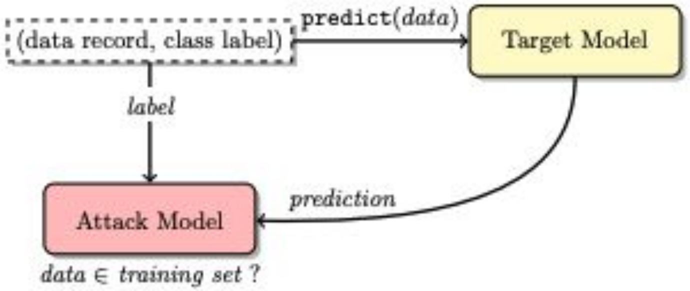
- Sensitive Data

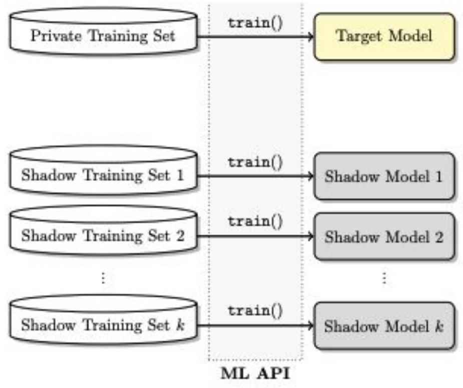


What are Membership Inference Attacks

Given a data record and black-box access to a trained model, can an adversary determine if that record was part of the model's training dataset?

- Sensitive Data
- Machine Learning as a Service (MLaaS)



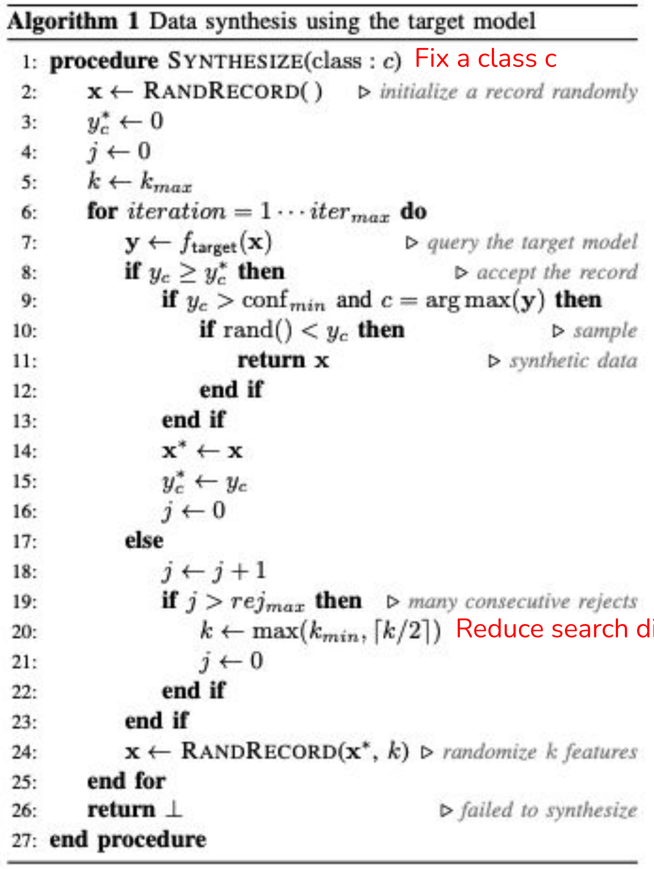


Amazon Machine Learning

Generating Training Data for Shadow Models

Option 1: Model-Based Synthesis

1. Seach (using hill-climbing) the space of possible data records to identify inputs classified by the target model with high confidence.
2. Sample synthetic data from these records. Repeat until the training dataset is full.



Algorithm 1 Data synthesis using the target model

```
procedure SYNTHESIZE(class : $c$)   Fix a class c
    $\mathbf{x} \leftarrow$ RANDRECORD( )   ▷ initialize a record randomly
    $y_c^* \leftarrow 0$
    $j \leftarrow 0$
    $k \leftarrow k_{max}$
    for $iteration = 1 \cdots iter_{max}$ do
        $\mathbf{y} \leftarrow f_{\text{target}}(\mathbf{x})$   ▷ query the target model
        if $y_c \geq y_c^*$ then   ▷ accept the record
            if $y_c > \text{conf}_{min}$ and $c = \arg\max(\mathbf{y})$ then
                if rand() $< y_c$ then   ▷ sample
                    return $\mathbf{x}$   ▷ synthetic data
                end if
            end if
            $\mathbf{x}^* \leftarrow \mathbf{x}$
            $y_c^* \leftarrow y_c$
            $j \leftarrow 0$
        else
            $j \leftarrow j + 1$
            if $j > rej_{max}$ then   ▷ many consecutive rejects
                $k \leftarrow \max(k_{min}, \lceil k/2 \rceil)$   Reduce search diameter
                $j \leftarrow 0$
            end if
        end if
        $\mathbf{x} \leftarrow$ RANDRECORD($\mathbf{x}^*$, $k$)   ▷ randomize $k$ features
    end for
    return $\perp$   ▷ failed to synthesize
end procedure
```

Generating Training Data for Shadow Models

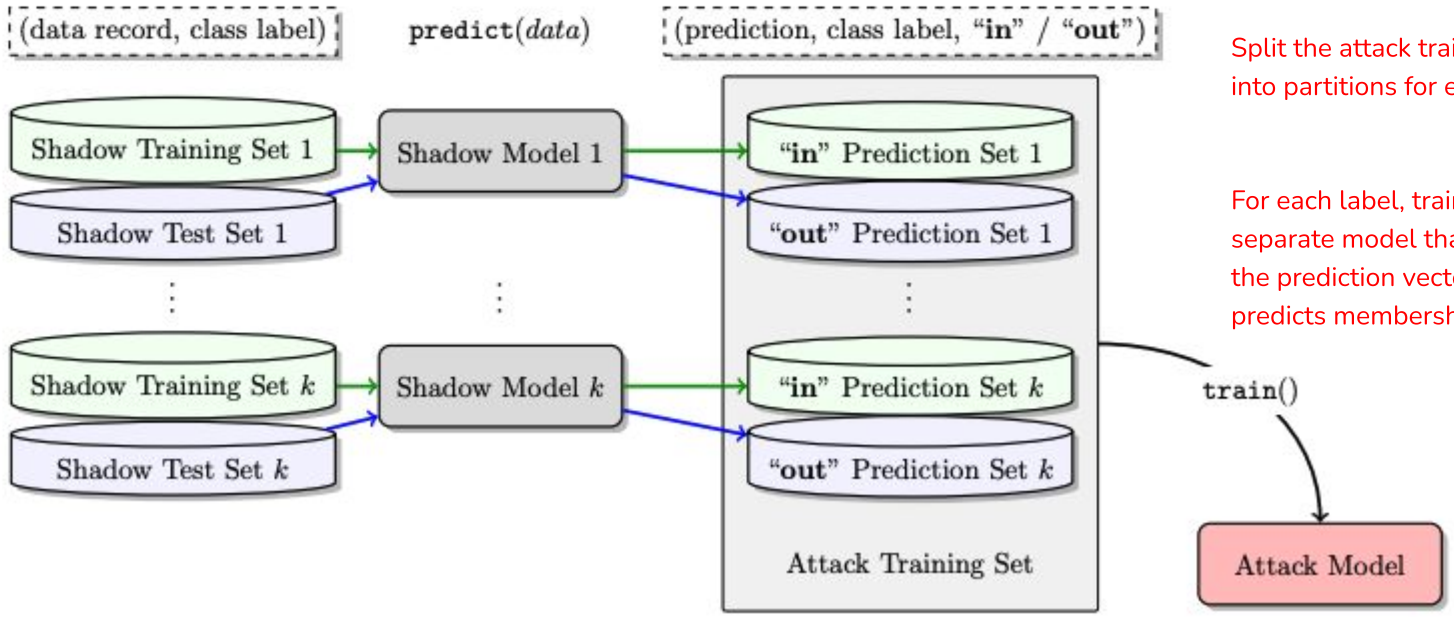
Option 2: Statistics Based Synthesis

- Knowledge of the marginal distributions of the features
- Generate synthetic records by sampling each feature independently according to its marginal distribution c

Option 3: Noisy, Real Data

- Attacker has access to similar data, but not identical ("noisy" version),
- Simulate realistic conditions with data augmentation.

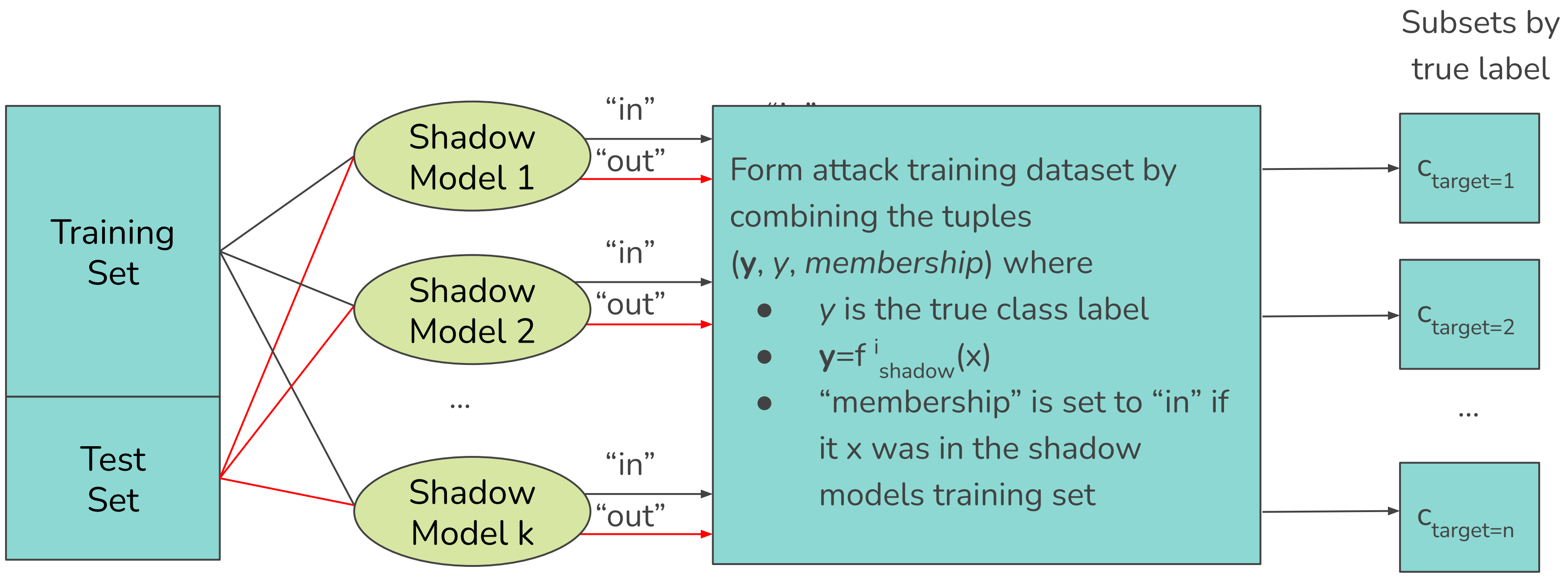
Training the Model



Split the attack training set into partitions for each class.

For each label, train a separate model that, given the prediction vector, predicts membership status.

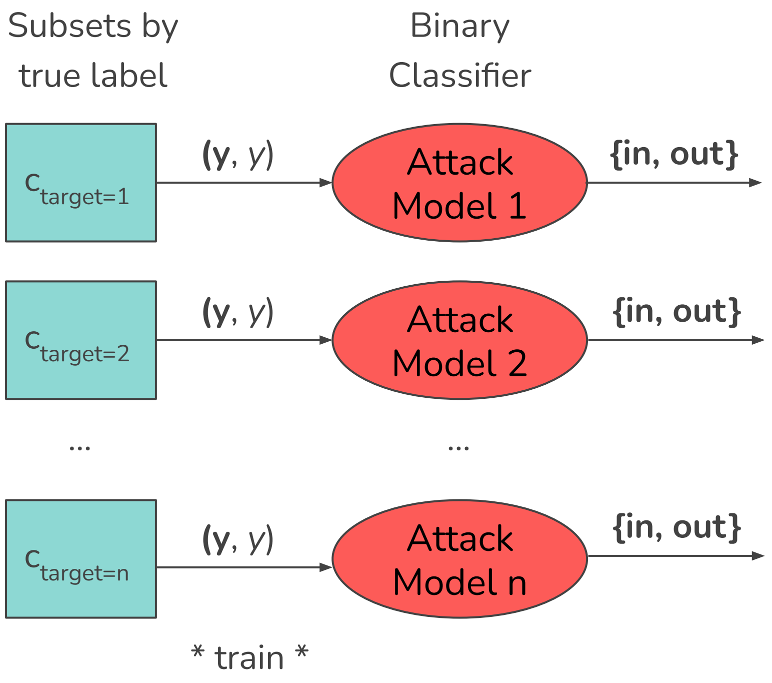
Training the Model



Form attack training dataset by combining the tuples (**y**, *y*, *membership*) where

- *y* is the true class label
- **y**$=f^{i}_{shadow}(x)$
- "membership" is set to "in" if it x was in the shadow models training set

Training the Model



Results & Key Findings: Overall Attack Accuracy

CIFAR datasets

- CIFAR-10: 15,000 training records; test accuracy ~0.6
    - Attack average precision of 0.74
- CIFAR-100: 29,540 training records; test accuracy ~0.2
    - Attack average precision of 0.988

Texas Hospital dataset

- Google-trained model: training and test accuracies of 0.66 and 0.51, respectively.
    - Attack precision was mostly above 0.6 and exceeded 0.85 for over 20 classes.

Location dataset

- Google-trained mode: Perfect training accuracy and 0.66 test accuracy
- Attack’s precision ranged between 0.6 and 0.8 with almost constant recall of 1.

Results & Key Findings: Comparison Across Platforms

Google models tended to leak more information than Amazon trained models.

- Precession was 0.505 for 2 classes and 0.935 for 100 classes.

Lower attack precession on local model.

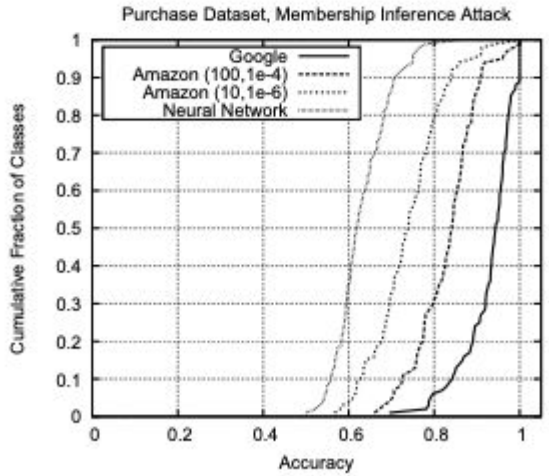


Fig. 7: Precision of the membership inference attack against models trained on the same datasets but using different platforms. The attack model is a neural network.

Results & Key Findings: Shadow Model Training Data Quantity

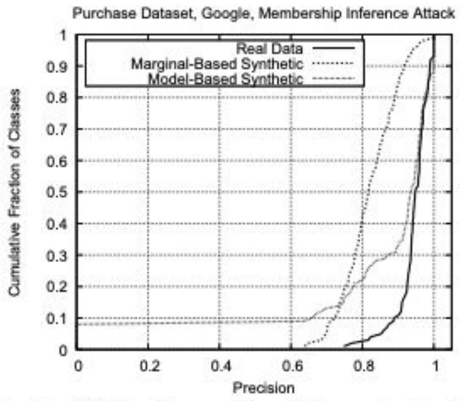


Fig. 9: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the purchase dataset. Results are shown for different ways of generating training data for the shadow models (real, synthetic generated from the target model, synthetic generated from marginal statistics). Precision of the attack over all classes is 0.935 (real data), 0.795 (marginal-based synthetic data), and 0.896 (model-based synthetic data). The corresponding recall of the attack is 0.994, 0.991, and 0.526, respectively.

The attack remains robust even with added noise:

- Baseline attack precision: 0.678
- With 10% noise: precision drops to ~0.666
- With 20% noise: precision drops to ~0.613

Marginal-based Synthesis

- Attack precision: 0.795

Model-based Synthesis

- 0.896 for most classes
- Some underrepresented classes had precision before 0.1

Results & Key Findings: Number of Classes and Training Distribution

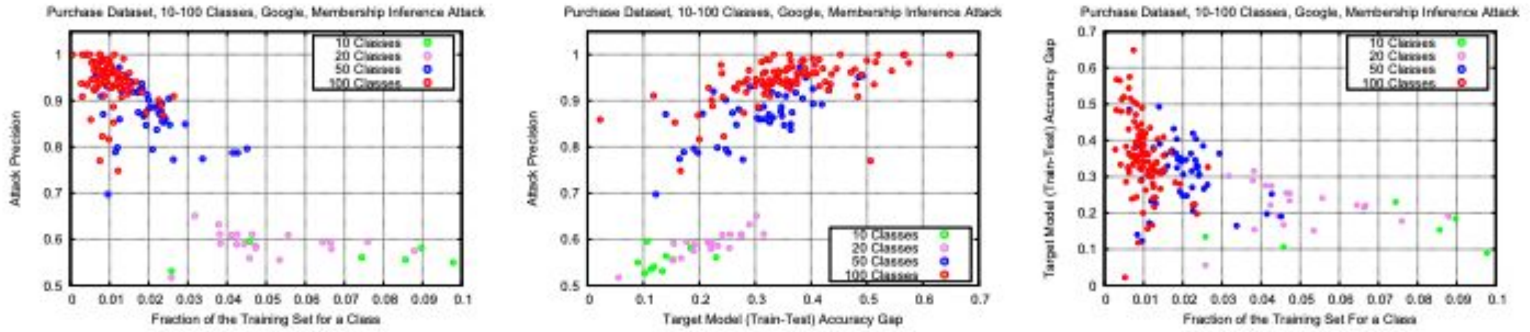
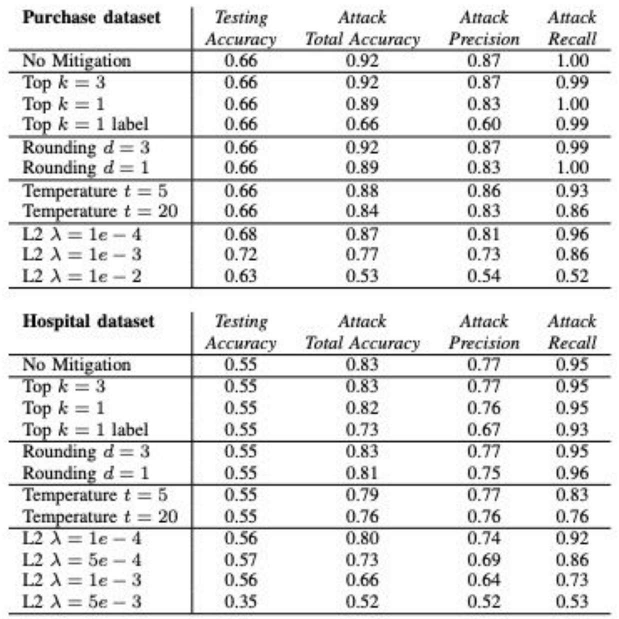


Fig. 11: Relationship between the precision of the membership inference attack on a class and the (train-test) accuracy gap of the target model, as well as the fraction of the training dataset that belongs to this class. Each point represent the values for one class. The (train-test) accuracy gap is a metric for generalization error [18] and an indicator of how overfitted the target model is.

Mitigations

1) Top-k filtering restricts the prediction vector to the top k classes
2) Coarsening prediction precision by rounding the probabilities to fewer digits
3) Increasing the temperature parameter in the softmax layer produces a flatter probability distribution
4) Adding an L2 penalty $\lambda \Sigma \Theta_i^2$ the loss function to reduce overfitting, improving generalization while also reducing leakage



Purchase dataset	*Testing Accuracy*	*Attack Total Accuracy*	*Attack Precision*	*Attack Recall*
No Mitigation	0.66	0.92	0.87	1.00
Top $k = 3$	0.66	0.92	0.87	0.99
Top $k = 1$	0.66	0.89	0.83	1.00
Top $k = 1$ label	0.66	0.66	0.60	0.99
Rounding $d = 3$	0.66	0.92	0.87	0.99
Rounding $d = 1$	0.66	0.89	0.83	1.00
Temperature $t = 5$	0.66	0.88	0.86	0.93
Temperature $t = 20$	0.66	0.84	0.83	0.86
L2 $\lambda = 1e-4$	0.68	0.87	0.81	0.96
L2 $\lambda = 1e-3$	0.72	0.77	0.73	0.86
L2 $\lambda = 1e-2$	0.63	0.53	0.54	0.52

Hospital dataset	*Testing Accuracy*	*Attack Total Accuracy*	*Attack Precision*	*Attack Recall*
No Mitigation	0.55	0.83	0.77	0.95
Top $k = 3$	0.55	0.83	0.77	0.95
Top $k = 1$	0.55	0.82	0.76	0.95
Top $k = 1$ label	0.55	0.73	0.67	0.93
Rounding $d = 3$	0.55	0.83	0.77	0.95
Rounding $d = 1$	0.55	0.81	0.75	0.96
Temperature $t = 5$	0.55	0.79	0.77	0.83
Temperature $t = 20$	0.55	0.76	0.76	0.76
L2 $\lambda = 1e-4$	0.56	0.80	0.74	0.92
L2 $\lambda = 5e-4$	0.57	0.73	0.69	0.86
L2 $\lambda = 1e-3$	0.56	0.66	0.64	0.73
L2 $\lambda = 5e-3$	0.35	0.52	0.52	0.53

TABLE III: The accuracy of the target models with different mitigation techniques on the purchase and Texas hospital-stay datasets (both with 100 classes), as well as total accuracy, precision, and recall of the membership inference attack. The relative reduction in the metrics for the attack shows the effectiveness of the mitigation strategy.

The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks

Carlini et al. (2018)

Unintended Memorization in Neural Networks

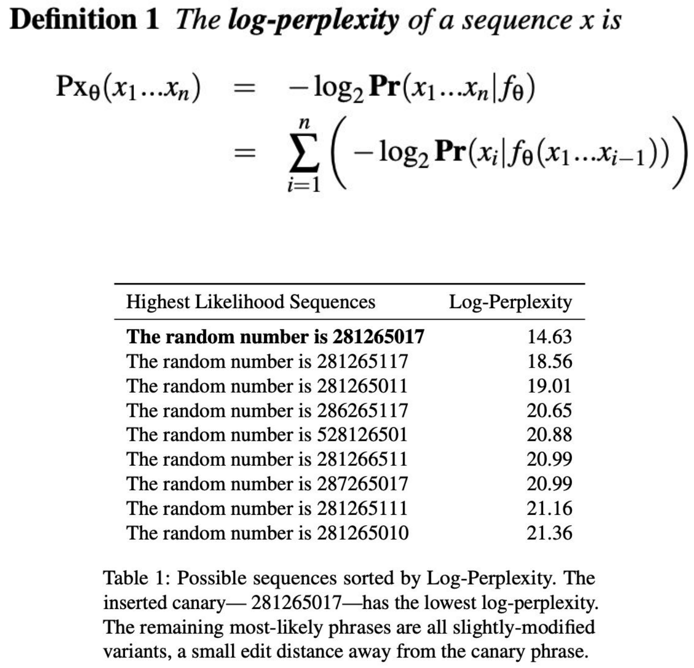
Neural networks, especially generative models, can memorize rare or unique training sequences

Sensitive data, like credit card numbers or private messages, may appear even if it’s uncommon

- This memorization is unintentional and often goes undetected during evaluation
- If exposed, attackers can extract actual training data content - not just infer presence

What Does Memorization Look Like?

- Imagine inserting “The secret code is 1234567890” into training data
- After training, prompt the model with: “The secret code is”
- If the model outputs exact match, has it memorized it?
- Need a way to measure how likely the model is to generate this exact phrase



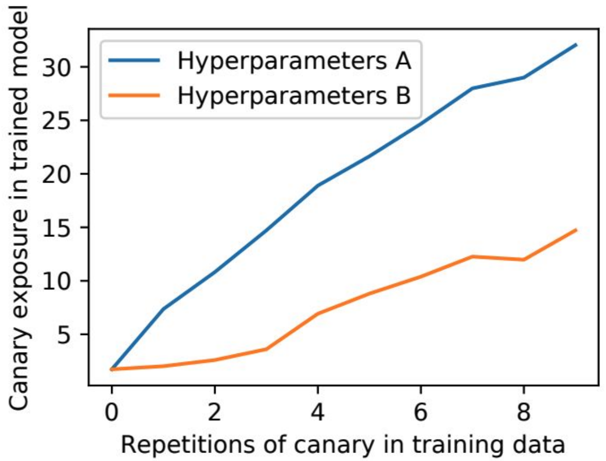
Definition 1 *The **log-perplexity** of a sequence x is*

$$
\begin{aligned}
\mathrm{Px}_\theta(x_1 \ldots x_n) &= -\log_2 \mathbf{Pr}(x_1 \ldots x_n | f_\theta) \\
&= \sum_{i=1}^{n} \left(-\log_2 \mathbf{Pr}(x_i | f_\theta(x_1 \ldots x_{i-1})) \right)
\end{aligned}
$$

Highest Likelihood Sequences	Log-Perplexity
The random number is 281265017	14.63
The random number is 281265117	18.56
The random number is 281265011	19.01
The random number is 286265117	20.65
The random number is 528126501	20.88
The random number is 281266511	20.99
The random number is 287265017	20.99
The random number is 281265111	21.16
The random number is 281265010	21.36

Table 1: Possible sequences sorted by Log-Perplexity. The inserted canary— 281265017—has the lowest log-perplexity. The remaining most-likely phrases are all slightly-modified variants, a small edit distance away from the canary phrase.

Measuring Memorization: The Exposure Metric



- Insert unique, random “canary” sequences into training data
- After training, measure how likely the model is to generate the canary
- Exposure = how much more likely the canary is vs. random alternatives
- High exposure -> model likely memorized the sequence

Exposure to Extraction

- When exposure is high, memorized canaries can be recovered
- Brute-force search over all possibilities is too slow
- Authors develop a shortest-path search (Dijkstra-like) for efficient extraction
- Achieves up to 100,000x speedup over brute-force

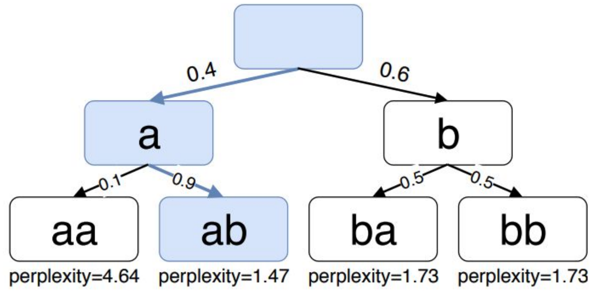
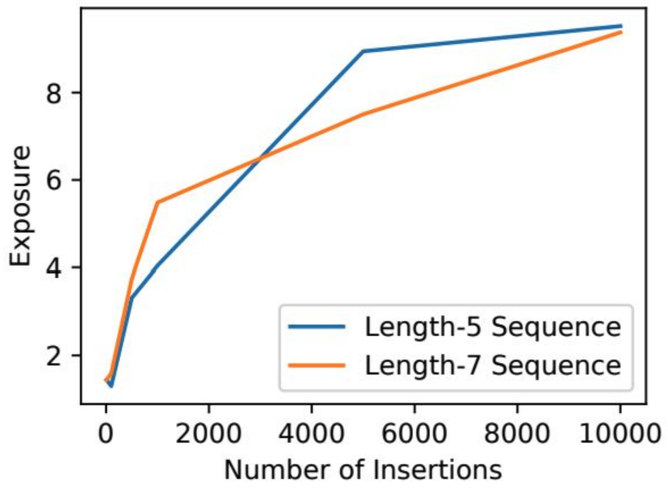


Figure 9: An example to illustrate the shortest path search algorithm. Each node represents one partially generated string. Each edge denotes the conditional probability $\mathbf{Pr}(x_i|x_1 \ldots x_{i-1})$. The path to the leaf with minimum perplexity is highlighted, and the log-perplexity is depicted below each leaf node.

Case Study: Google's Smart Compose



- Smart Compose: Gmail's autocomplete system trained on millions of user emails
- Inserted canaries 1 to 10,000 times during training
- Exposure increased with frequency, but canaries were not extractable
- Used exposure metric to tune privacy risk, e.g., via differential privacy

Can We Prevent Memorization?

- **Regularization techniques** (e.g., dropout, weight decay): Ineffective
- **Sanitization** (e.g. blacklisting sensitive patterns): Helps, but incomplete
- **Differential Privacy (DP-SGD):** Only reliable defense
    - DP training eliminated memorization with only minor utility loss

Takeaways

- Neural networks can **unintentionally memorize and leak rare training data**
- The **exposure metric** quantifies this risk and enables real extraction
- Memorization occurs **early in training** and isn’t prevented by regularization
- **Differential Privacy** is the only proven defense that works

Discussion (10 minutes)

- What do you think about using black box APIs to train models and to be used in production?
  - If you had a classification model in production would you share the entire prediction probability vector?
- How might membership inference attacks exacerbate existing biases in models trained on imbalanced datasets (e.g., criminal justice or hiring systems)?
- Results show that some classes exhibit a much lower attack precision than others. Is this something researchers should disclose as well? If so, what things so far in the semester that we have discussed should be included in this disclosure
  - Where do we draw the line?
- Do you think that companies which deploy generative language models (like Smart Compose) be required to audit for memorized secrets? Why or why not?

Re-Evaluating Attack Evaluation
Carlini et al., (2022)

Prior Evaluation Technique: Averaging

Limitations: Treats all datapoints the same when they can have meaningful differences across certain dimensions

False negatives (not very effective) vs false positives (very effective)

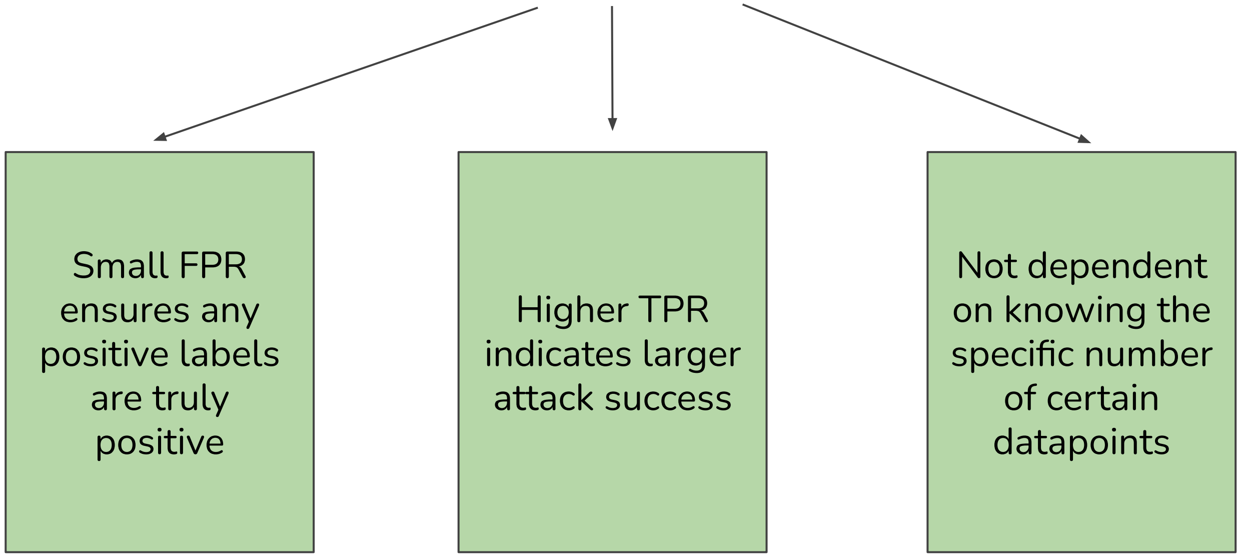
Larger chance of correctly randomly guessing vs smaller change of guaranteed correctness (not reflected in accuracy)

Approach

1. Propose an new evaluation technique
2. Construct an attack
3. Evaluate Performance
4. Ablation Study

Proposed Evaluation Technique

True positive rate (TPR) at a small false positive rate (FPR):


Small FPR ensures any positive labels are truly positive
Higher TPR indicates larger attack success
Not dependent on knowing the specific number of certain datapoints

Proposed Evaluation Technique

True positive rate (TPR) at a small false positive rate (FPR):

- Small FPR ensures any positive labels are truly positive
- Higher TPR indicates larger attack success
- Not dependent on knowing the specific number of certain datapoints

Can be displayed using ROC curves

Constructing an Attack

Baseline Attack: LOSS

ℓ < **threshold** $\rightarrow$ ✓ In training data
ℓ > **threshold** $\rightarrow$ ✕ Not in training data

$\rightarrow$ good at identifying **non**-training examples, bad at identifying training examples

Constructing an Attack

Baseline Attack: LOSS → Scores badly on proposed metric:

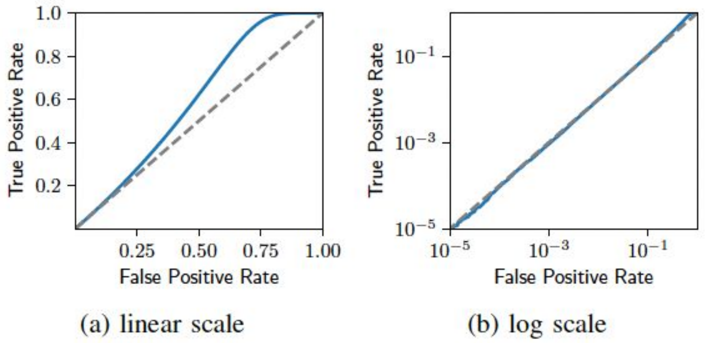


Fig. 2: ROC curve for the LOSS baseline membership inference attack, shown with both linear scaling (left), also and log-log scaling (right) to emphasize the low-FPR regime.

Carlini et al., (2022), Figure 2, page 4)

Constructing an Attack

Proposed Attack: Likelihood Ratio Attack (LiRA):

1. Train shadow models on datapoints
2. Compute Likelihood-ratio test to determine datapoint label

Constructing an Attack

1. Train shadow models on datapoints

For a given datapoint ***d**:*

- $\mathcal{Q}_{\text{in}}$: distribution of models trained on ***d*** (50% of models)
- $\mathcal{Q}_{\text{out}}$: distribution of models not trained on ***d*** (50% of models)

Constructing an Attack

2. Compute Likelihood-ratio test to determine datapoint label

$$\Lambda(f; x, y) = \frac{p(f \mid \mathbb{Q}_{\text{in}}(x, y))}{p(f \mid \mathbb{Q}_{\text{out}}(x, y))}, \qquad (2)$$

(Carlini et al., (2022), Equation 2, page 4)

Constructing an Attack

Some Issues:

1. Can’t compute these distributions

Constructing an Attack

Some Issues:

1. Can’t compute these distributions → use distributions over model losses:

$$p(\ell(f(x), y) \mid \tilde{\mathbb{Q}}_{\text{in/out}}(x, y)). \qquad (3)$$

(Carlini et al., (2022), Equation 3, page 4)

Constructing an Attack

Some Issues:

1. Can’t compute these distributions → use distributions over model losses:

$$p(\ell(f(x), y) \mid \tilde{\mathbb{Q}}_{\text{in/out}}(x, y)). \qquad (3)$$

(Carlini et al., (2022), Equation 3, page 4)

2. May need many shadow models to estimate all distribution parameters

Constructing an Attack

Some Issues:

1. Can’t compute these distributions → use distributions over model losses:

$$p(\ell(f(x), y) \mid \tilde{\mathbb{Q}}_{\text{in/out}}(x, y)). \qquad (3)$$

(Carlini et al., (2022), Equation 3, page 4)

2. May need many shadow models to estimate all distribution parameters → assume $\boldsymbol{Q}_{\text{in}}$ and $\boldsymbol{Q}_{\text{out}}$ are Gaussian distributions

Constructing an Attack

Some Issues:

1. Can’t compute these distributions → use distributions over model losses:

$$p(\ell(f(x), y) \mid \tilde{\mathbb{Q}}_{\mathbf{in/out}}(x, y)). \qquad (3)$$

(Carlini et al., (2022), Equation 3, page 4)

2. May need many shadow models to estimate all distribution parameters → assume $\boldsymbol{Q}_{\text{in}}$ and $\boldsymbol{Q}_{\text{out}}$ are Gaussian distributions

3. Models’ losses/confidence are not inherently Gaussian

Constructing an Attack

Some Issues:

1. Can't compute these distributions → use distributions over model losses:

$$p(\ell(f(x), y) \mid \tilde{\mathbb{Q}}_{\text{in/out}}(x, y)). \qquad (3)$$

(Carlini et al., (2022), Equation 3, page 4)

2. May need many shadow models to estimate all distribution parameters → assume $\mathcal{Q}_{\text{in}}$ and $\mathcal{Q}_{\text{out}}$ are Gaussian distributions

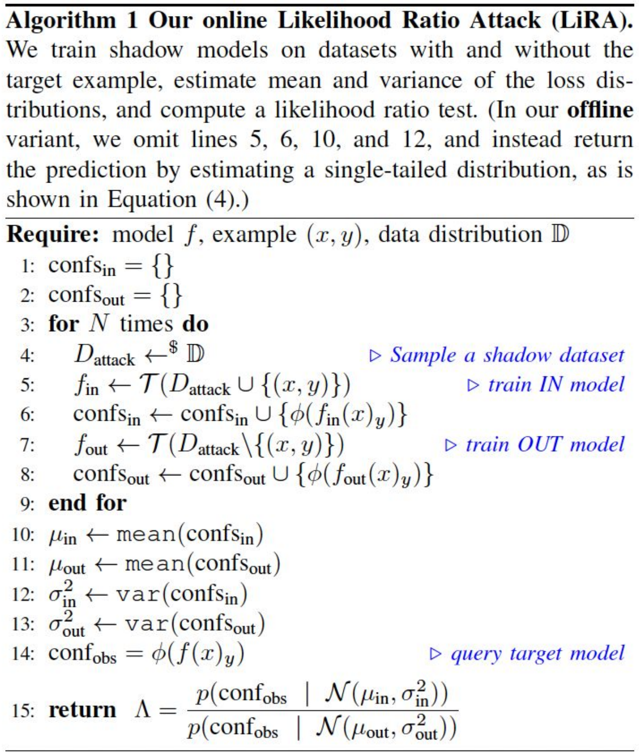
3. Models' losses/confidence are not inherently Gaussian → try to model them as Gaussian:

$$\phi(p) = \log\left(\frac{p}{1-p}\right), \quad \text{for } p = f(x)_y$$

(Carlini et al., (2022), page 5)

Constructing an Attack

Online Attack:



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).)

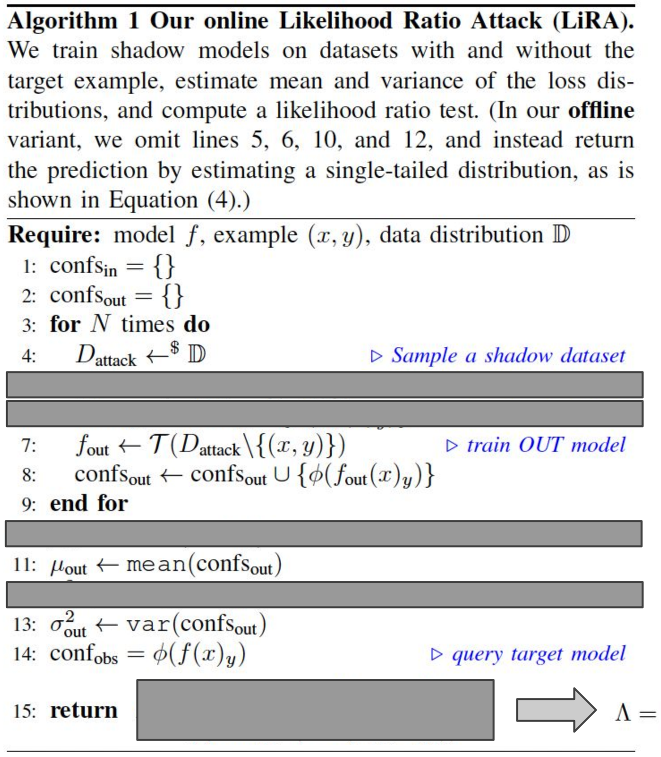
```
Require: model $f$, example $(x, y)$, data distribution $\mathbb{D}$
 1: $\text{confs}_{\text{in}} = \{\}$
 2: $\text{confs}_{\text{out}} = \{\}$
 3: for $N$ times do
 4:     $D_{\text{attack}} \xleftarrow{\$} \mathbb{D}$                      ▷ Sample a shadow dataset
 5:     $f_{\text{in}} \leftarrow \mathcal{T}(D_{\text{attack}} \cup \{(x, y)\})$          ▷ train IN model
 6:     $\text{confs}_{\text{in}} \leftarrow \text{confs}_{\text{in}} \cup \{\phi(f_{\text{in}}(x)_y)\}$
 7:     $f_{\text{out}} \leftarrow \mathcal{T}(D_{\text{attack}} \backslash \{(x, y)\})$         ▷ train OUT model
 8:     $\text{confs}_{\text{out}} \leftarrow \text{confs}_{\text{out}} \cup \{\phi(f_{\text{out}}(x)_y)\}$
 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 $\Lambda = \frac{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}}))}{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}}))}$
```

(Carlini et al., (2022), Algorithm 1, page 6)

Constructing an Attack

Offline Attack:

- No training “in” models
- Assume *(x, y)* is not in the training data
- Compare model confidences relative to confs$_{out}$ (further away = less more likely to be in the training data)



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:     $D_{\text{attack}} \leftarrow^{\$} \mathbb{D}$            ▷ Sample a shadow dataset
[redacted]
[redacted]
 7:     $f_{\text{out}} \leftarrow \mathcal{T}(D_{\text{attack}} \backslash \{(x, y)\})$            ▷ train OUT model
 8:     $\text{confs}_{\text{out}} \leftarrow \text{confs}_{\text{out}} \cup \{\phi(f_{\text{out}}(x)_y)\}$
 9: end for
[redacted]
11: $\mu_{\text{out}} \leftarrow \texttt{mean}(\text{confs}_{\text{out}})$
[redacted]
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 [redacted]
```

(Carlini et al., (2022), Algorithm 1, page 6)

$$\Rightarrow \quad \Lambda = 1 - \Pr[Z > \phi(f(x)_y)], \text{where } Z \sim \mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}}) \; . \quad (4)$$

(Carlini et al., (2022), Equation 4, page 6)

Constructing an Attack

- Addition to the method: query multiple datapoint augmentations (e.g., image augmentations) to improve attack accuracy

Constructing an Attack

Proposed attack's advantage over LOSS: accounts for more nuances in loss/membership distribution

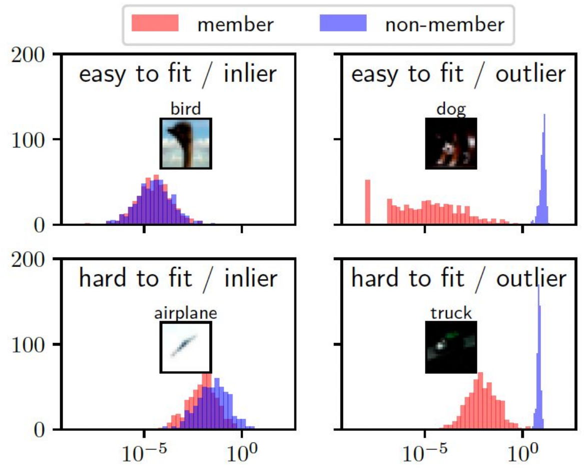


Fig. 3: Some examples are easier to fit than others, and some have a larger separability between their losses when being a member of the training set or not. We train 1024 models on random subsets of CIFAR-10 and plot the losses for four examples when the example is a member of the training set ($\tilde{\mathbb{Q}}_{\text{in}}(x,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

(Carlini et al., (2022), Figure 3, page 5)

Findings

Online and offline attack results are similar

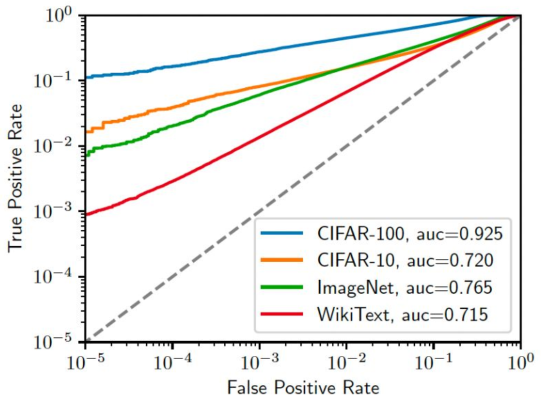


Fig. 5: **Success rate of our attack on CIFAR-10, CIFAR-100, ImageNet, and WikiText.** All plots are generated with 256 shadow models, except ImageNet which uses 64.

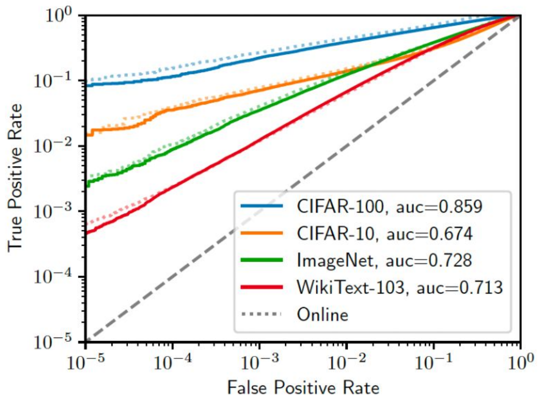
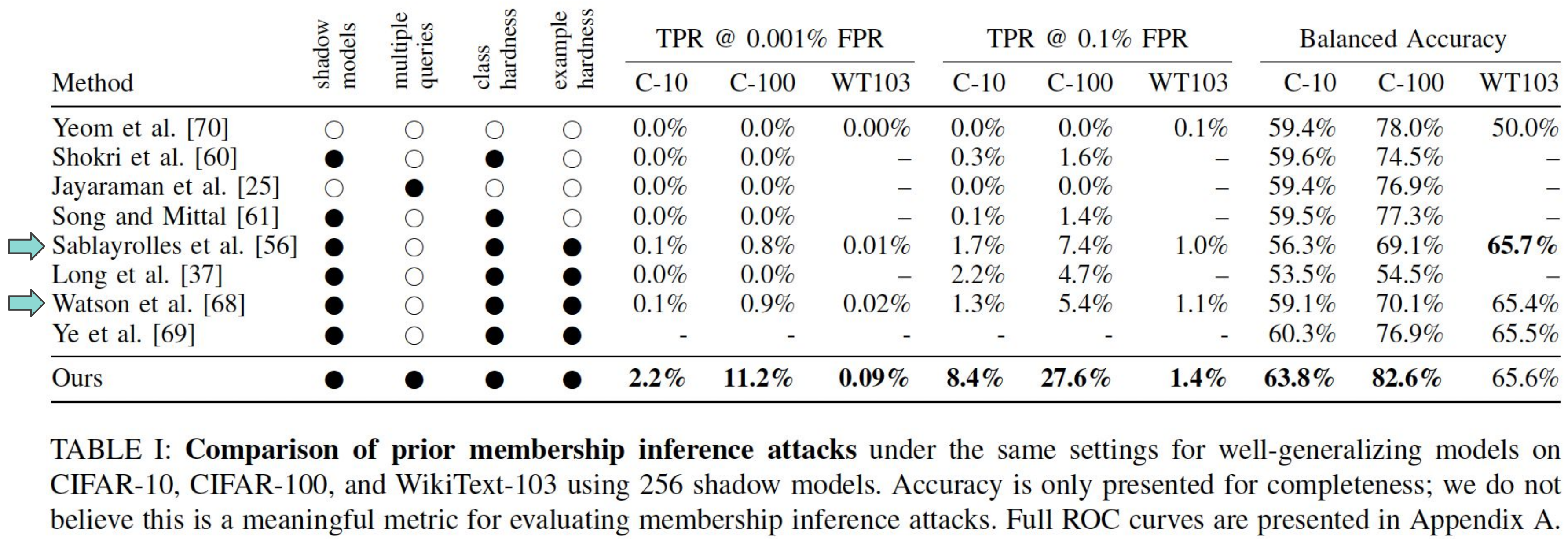


Fig. 6: **Success rate of our offline attack on CIFAR-10, CIFAR-100, ImageNet, and WikiText.** All plots are generated with 128 OUT shadow models, except ImageNet which uses 32. For each dataset, we also plot our online attack with the same number of shadow models (half IN, half OUT).

(Carlini et al., (2022), Figures 5 and 6, page 7)

Findings

Scoring prior work using the proposed method:



Method	shadow models	multiple queries	class hardness	example hardness	TPR @ 0.001% FPR C-10	TPR @ 0.001% FPR C-100	TPR @ 0.001% FPR WT103	TPR @ 0.1% FPR C-10	TPR @ 0.1% FPR C-100	TPR @ 0.1% FPR WT103	Balanced Accuracy C-10	Balanced Accuracy C-100	Balanced Accuracy 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.

(Carlini et al., (2022), Table 1, page 8)

Findings

Testing the proposed attack on other pretrained models - same architecture is helpful, but not necessary:

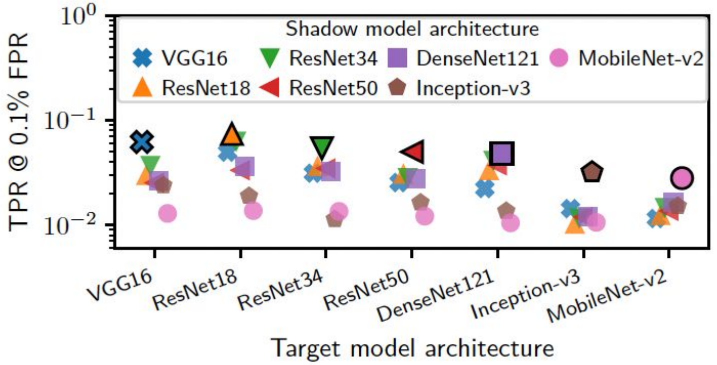


Fig. 12: Our attack succeeds against real state-of-the-art CIFAR-10 models [51]. The attacker trains shadow models on a random subset of 50,000 points from the entire CIFAR-10 dataset. The attack performs best when the shadow models have the same architecture as the target model, but training different models still leads to a strong attack.

(Carlini et al., (2022), Figure 12, page 12)

Findings

Examples that are more out of distribution are more easily detectable in models (higher privacy score = more detectable)

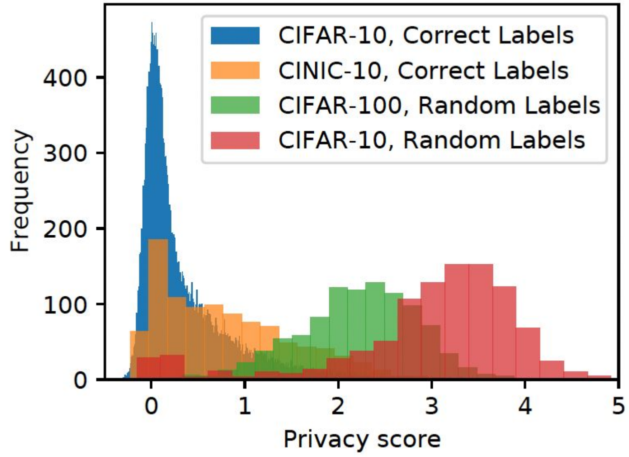


Fig. 13: Out-of-distribution training examples are less private.

(Carlini et al., (2022), Figure 13, page 13)

Ablation Study

Logit Scaling

Number of Shadow Models

Queries

Data Overlap

Model Architectures and Hyperparameters

Ablation Study

Logit Scaling:

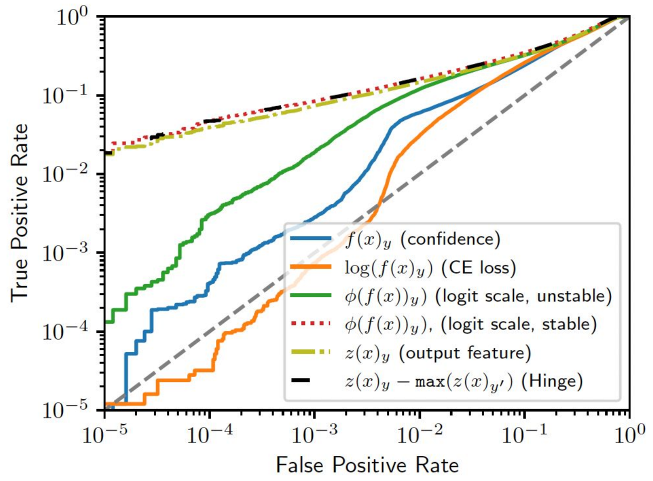


Fig. 8: The best scoring metrics ensure the output distribution is approximately Gaussian, and the worst metrics are not easily modeled with a standard distribution (see Figure 4).

(Carlini et al., (2022), Figure 8, page 10)

Number of Shadow Models:

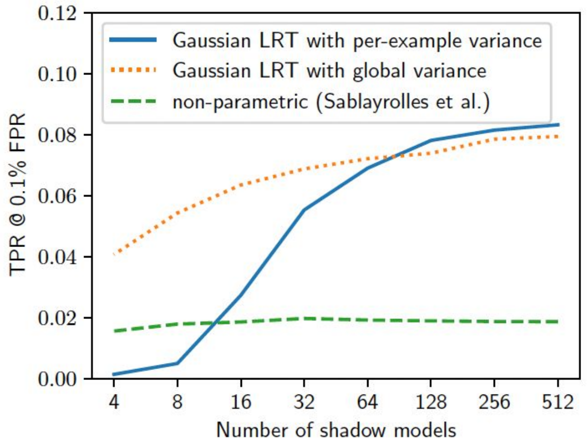
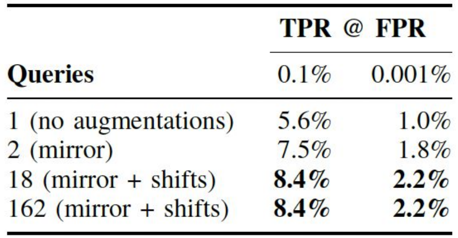


Fig. 9: Attack success rate increases as the number of shadow models increases, with the benefit eventually tapering off. When fewer than 64 models are used, it is better to estimate the variance of the model confidence as a global parameter instead of computing it on a per-example basis.

(Carlini et al., (2022), Figure 9, page 10)

Ablation Study

Queries:



	TPR @ FPR	
Queries	0.1%	0.001%
1 (no augmentations)	5.6%	1.0%
2 (mirror)	7.5%	1.8%
18 (mirror + shifts)	**8.4%**	**2.2%**
162 (mirror + shifts)	**8.4%**	**2.2%**

TABLE III: Querying on augmented versions of the image doubles the true-positive rate at low false-positive rates, with most benefits given by just two queries.

(Carlini et al., (2022), Table III, page 11)

Data Overlap:

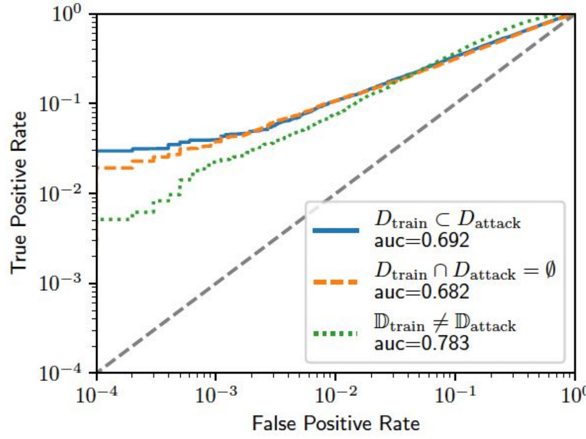
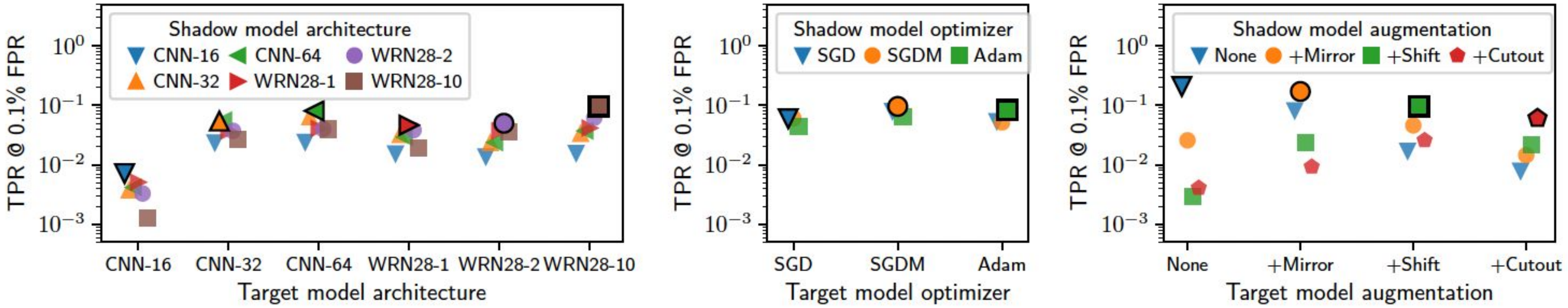


Fig. 10: The attack's success rate on CINIC-10 remains unchanged when the training sets of shadow models are sampled from a dataset D_{attack} that is disjoint from the target model's training set D_{train}. The attack's performance does decrease when the two datasets are sampled from different *distributions*.

(Carlini et al., (2022), Figure 10, page 11)

Ablation Study

Architecture and Hyperparameters:



(a) Vary model architecture.
(b) Vary training optimizer.
(c) Vary data augmentation.

Fig. 11: Our attack succeeds when the adversary is uncertain of the target model's training setup. We vary the target model's architecture (a), the training optimizer (b) and the data augmentation (c), as well as the adversary's guess of each of these properties when training shadow models. The attack performs best when the adversary guesses correctly (black-lined markers).

(Carlini et al., (2022), Figure 11, page 12)

Discussion (10 minutes)

- Much of the work performed in Carlini et al., (2022) assumed Gaussian distributions for models and model confidences (and, for model confidences, transformed their values to approximate a Gaussian distribution). Do you think this is a fair assumption to make, and/or might there be negative effects of this assumption?

- Having reliable evaluation metrics for AI models is just as important as finding techniques to make models safer and more reliable. What could future research into safe/private/responsible AI do to ensure that both model development and evaluation are robust/reliable?

How much “privacy” is provided by our defenses?

Selection of Privacy Parameter ε

- An acceptable amount of inferable information (ε) should be selected carefully in real-world situations
- Privacy vs accuracy tradeoff
  - High ε: weaker privacy guarantees, more data utility
  - Low ε: stronger privacy guarantees, less data utility

- In reality, this parameter is often set as large as possible while still guaranteeing some reasonable level of privacy

ε Derivation: Provable vs Empirical Privacy Guarantees

- Privacy techniques with provable guarantees (generally Differential Privacy) ensures an upper bound on ε through theoretical analysis

- Sometimes, theoretical analysis is not representative of the real-world, leading to unreasonably high privacy constraints
    - Tends to sacrifice significant data utility to reach some sufficiently low ε

ε Derivation: Provable vs Empirical Privacy Guarantees

- Otherwise, techniques have empirical guarantees - measuring the success rate of various attacks on our method to estimate an ε upper bound
- More realistic and representative of real-world applications, but provides weaker guarantees
    - As attacks adapt and evolve over time, estimated privacy guarantees through empirical analysis may no longer hold
- With a good empirical strategy, we can provide a deeper analysis of the level of privacy provided (estimate ε upper bound, demonstrate ε lower bound)

Combining Provable and Empirical ε Techniques

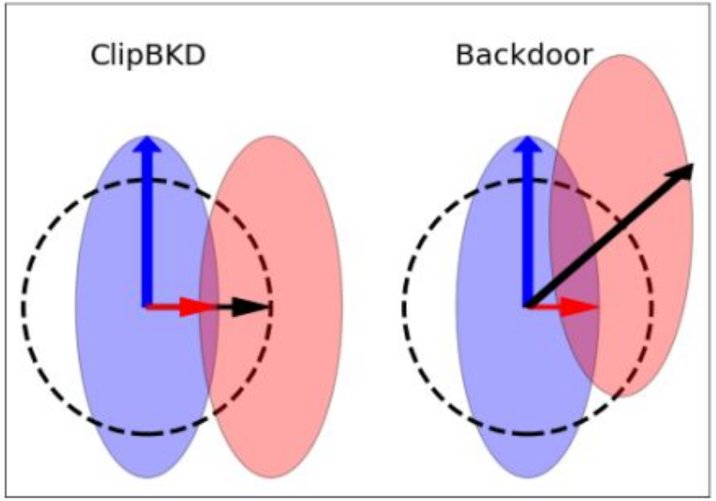
- We can empirically evaluate the privacy provided by differential privacy - DP-SGD - to find a more realistic level of privacy provided
  - Is there a disparity between the theoretical and real-world amount of privacy?

Evaluation experiment:

- If we have two nearly identical datasets, can we distinguish between DP-SGD models trained on each?

Constructing the Datasets

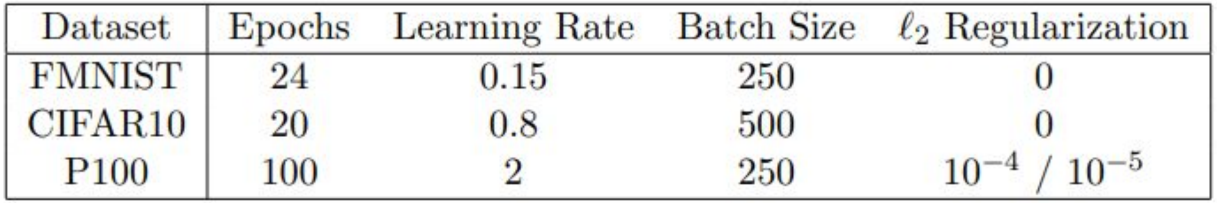
- With a base dataset, we can construct a similar alternate through data poisoning
- However, standard data poisoning attacks generally have poor performance on DP-SGD due to gradient clipping
    - ClipBKD pushes points in the direction of least variance to maximize distance



Experimental Setup

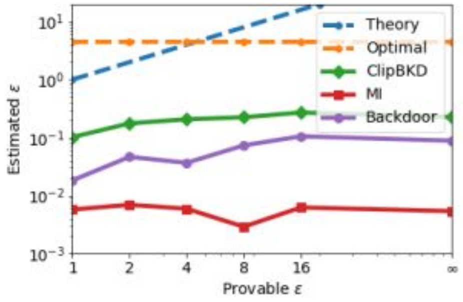
- Fairly standard experimental setup to ensure applicability to alternative dataset-model pairs
- Accuracy is maintained at 96-98% for consistency

- Two model types:
  - Two layer feed-forward neural network
  - Logistic Regression model
- Three datasets:
  - Fashion MNIST
  - CIFAR10
  - Purchase100

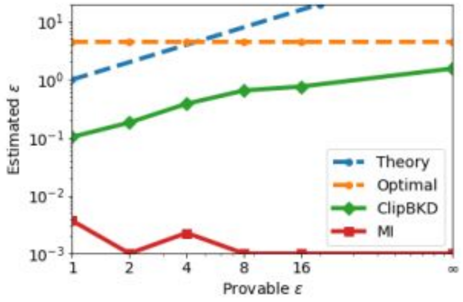


Dataset	Epochs	Learning Rate	Batch Size	ℓ_2 Regularization
FMNIST	24	0.15	250	0
CIFAR10	20	0.8	500	0
P100	100	2	250	10^{-4} / 10^{-5}

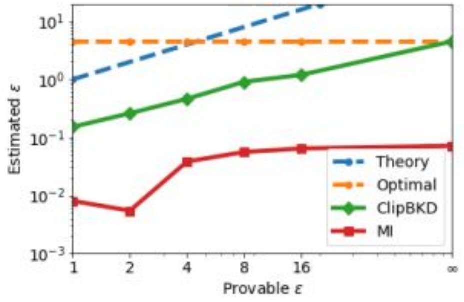
Results



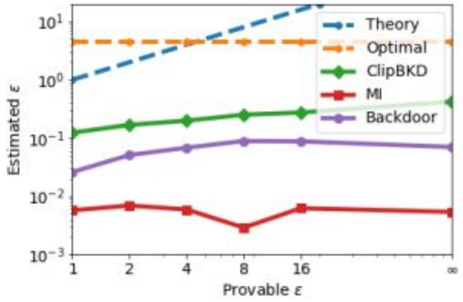
(a) FMNIST, LR



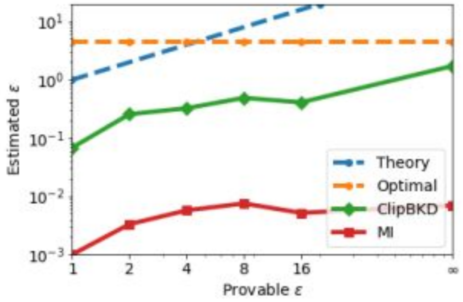
(b) CIFAR10, LR



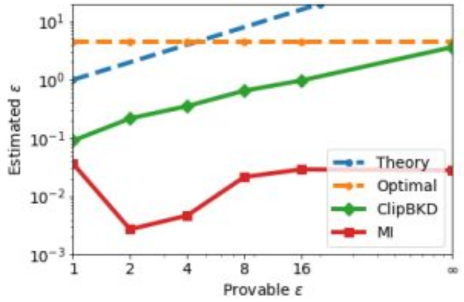
(c) P100, LR



(d) FMNIST, FNN



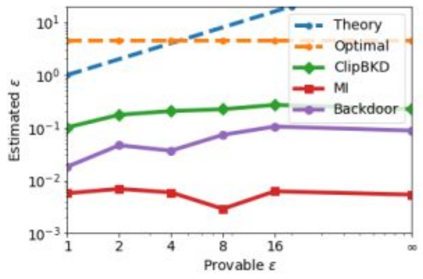
(e) CIFAR10, FNN



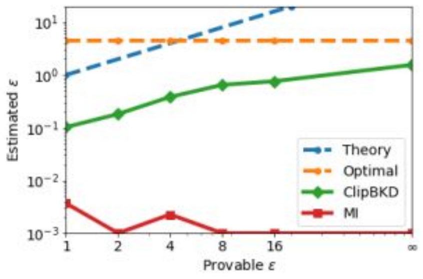
(f) P100, FNN

Upper and Lower Bounds of Privacy

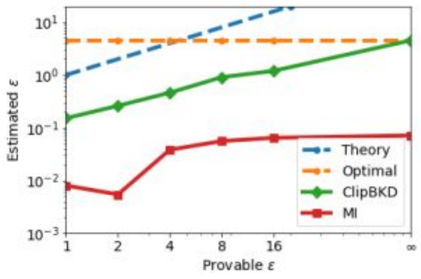
- Shrinking gap between the lower and worst-case upper bounds - there might not be much more room for improvement in the amount of privacy that can be realistically provided
- Empirical analyses can complement theoretical evaluation to find the “true” level of privacy provided



(a) FMNIST, LR



(b) CIFAR10, LR



(c) P100, LR

Discussion

- Should organizations that deploy DP systems be required to perform empirical privacy audits to avoid inaccurate privacy guarantees?
  - Are there downsides to this type of analysis?

- Assuming we can get an accurate estimation on the level of privacy provided by our techniques, how transparent should companies be with these privacy guarantees?

References

Carlini, Nicholas, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. “Membership Inference Attacks From First Principles,” 2022. https://arxiv.org/abs/2112.03570.

Carlini, Nicholas, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. “The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks,” 2019. https://arxiv.org/abs/1802.08232.

Jagielski, Matthew, Jonathan Ullman, and Alina Oprea. “Auditing Differentially Private Machine Learning: How Private Is Private SGD?,” 2020. https://arxiv.org/abs/2006.07709.

Shokri, Reza, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. “Membership Inference Attacks against Machine Learning Models,” 2017. https://arxiv.org/abs/1610.05820.