



## Privacy Risks

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology  
Computer Engineering Department (CE)  
Data and Network Security Lab (DNSL)



December 29, 2024

# Today's Agenda

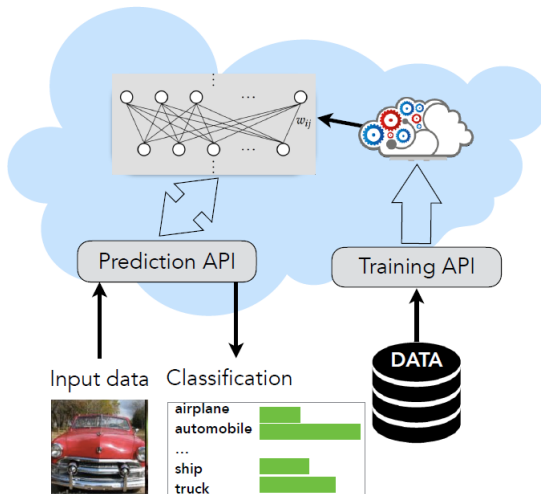
1 Recap

2 Membership Inference Attacks

## Recap

# Machine Learning as a Service

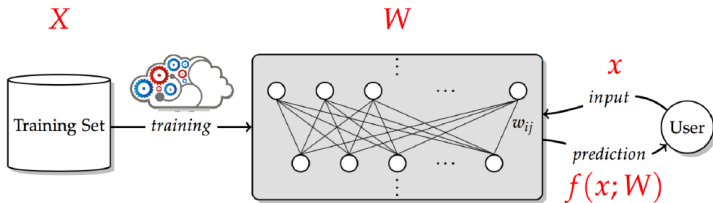
## Machine Learning as a Service



(Shokri, 2020)

# Privacy Risks in Machine Learning

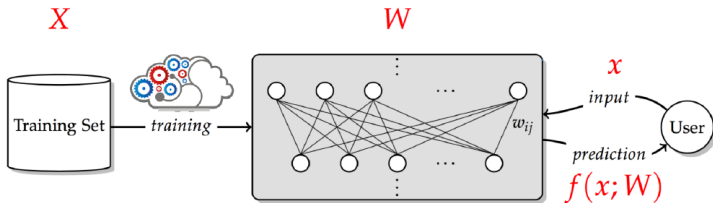
- What is training data leakage? Inferring information about members of  $X$ , beyond what can be learned about its underlying distribution.



(Shokri, 2020)

# Privacy Risks in Machine Learning

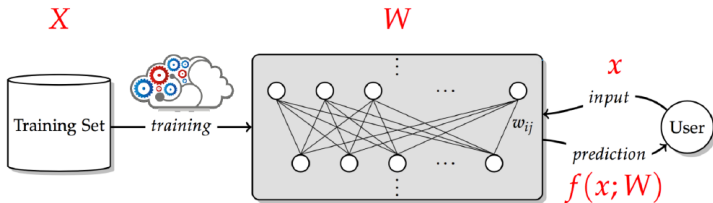
- What is training data leakage? Inferring information about members of  $X$ , beyond what can be learned about its underlying distribution.
- Valuable things: Training set  $X$ , user's data  $x$ , parameters  $W$ , prediction, etc.



(Shokri, 2020)

# Privacy Risks in Machine Learning

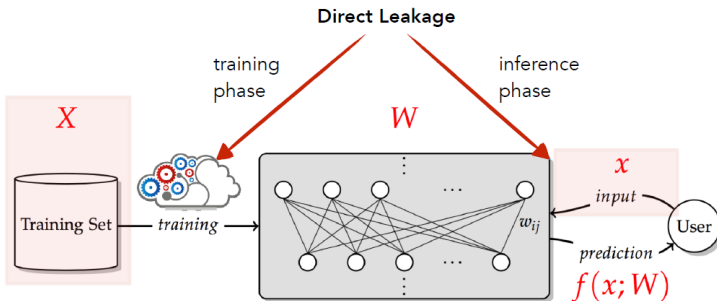
- What is training data leakage? Inferring information about members of  $X$ , beyond what can be learned about its underlying distribution.
- Valuable things: Training set  $X$ , user's data  $x$ , parameters  $W$ , prediction, etc.
- Adversary: malicious cloud, malicious user, the malicious data owner, etc.



(Shokri, 2020)

# Privacy Risks in Machine Learning

- How to prevent direct leakage? Secure multi-party computation, federated learning, homomorphic encryption, trusted hardware

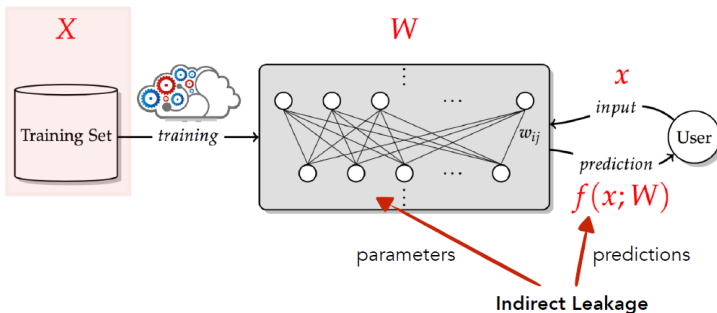


(Shokri, 2020)



# Privacy Risks in Machine Learning

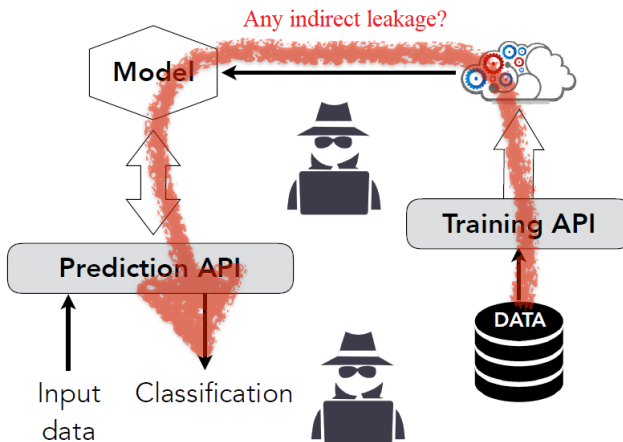
- How to prevent direct leakage? Secure multi-party computation, federated learning, homomorphic encryption, trusted hardware
- How to mitigate the indirect leakage? Differential privacy



(Shokri, 2020)

# Membership Inference Attack

- Given a model, can an adversary infer whether data point  $x$  is part of its training set?





(Shokri, 2020)

# Membership Inference Attack

- Given a model, can an adversary infer whether data point  $x$  is part of its training set?

Membership Inference:

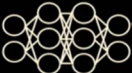

Was  trained  
on the example  ?

(Carlini, 2022)

# Membership Inference Attack

- Given a model, can an adversary infer whether data point  $x$  is part of its training set?

## Membership Inference:

$$A = \Pr(\text{ \text{ was trained on } \text{})$$

(Carlini, 2022)

# Model Inversion Attacks

- The attack uses a trained classifier in order to extract representations of the training data.



Original face image (right) and restored one through model inversion (left)

# Model Inversion Attacks

- The attack uses a trained classifier in order to extract representations of the training data.

---

**Algorithm 1** Inversion attack for facial recognition models.

---

```
1: function MI-FACE(label,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ )
2:    $c(\mathbf{x}) \stackrel{\text{def}}{=} 1 - \tilde{f}_{\text{label}}(\mathbf{x}) + \text{AUXTERM}(\mathbf{x})$ 
3:    $\mathbf{x}_0 \leftarrow \mathbf{0}$ 
4:   for  $i \leftarrow 1 \dots \alpha$  do
5:      $\mathbf{x}_i \leftarrow \text{PROCESS}(\mathbf{x}_{i-1} - \lambda \cdot \nabla c(\mathbf{x}_{i-1}))$ 
6:     if  $c(\mathbf{x}_i) \geq \max(c(\mathbf{x}_{i-1}), \dots, c(\mathbf{x}_{i-\beta}))$  then
7:       break
8:     if  $c(\mathbf{x}_i) \leq \gamma$  then
9:       break
10:  return  $[\arg \min_{\mathbf{x}_i} (c(\mathbf{x}_i)), \min_{\mathbf{x}_i} (c(\mathbf{x}_i))]$ 
```

---

# Generative Sequence Models



WHEN YOU TRAIN PREDICTIVE MODELS  
ON INPUT FROM YOUR USERS, IT CAN  
LEAK INFORMATION IN UNEXPECTED WAYS.

<https://xkcd.com/2169/>

# Generative Sequence Models

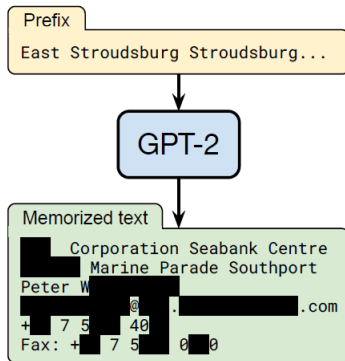


Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

*(Extracting Training Data from Large Language Models, Carlini, 2021)*



## Membership Inference Attacks

# Membership Inference Attacks

## Membership Inference Attacks Against Machine Learning Models

Reza Shokri  
Cornell Tech

shokri@cornell.edu

Marco Stronati\*

marco@stronati.org

Congzheng Song  
Cornell

cs2296@cornell.edu

Vitaly Shmatikov  
Cornell Tech

shmat@cs.cornell.edu

# Abstract

## membership inference attack

- Given a machine learning model and a record, determine whether this record was used as part of the model’s training dataset or not.

### Threat Model

- It is investigated in the most difficult setting, where the adversary’s access to the model is limited to **black-box** queries that return the model’s output on a given input.

### Approach

- Train an **inference model** to recognize **differences in the target model’s predictions** on the inputs that it trained on versus the inputs that it did not train on.

# Threat Model

- The adversary has query access to the model and can obtain the model's **prediction vector** on any data record.
- The attacker may have some background knowledge about the population from which the target model's training dataset was drawn.
  - For example, he may have **independently drawn samples from the population**, disjoint from the target model's training dataset.

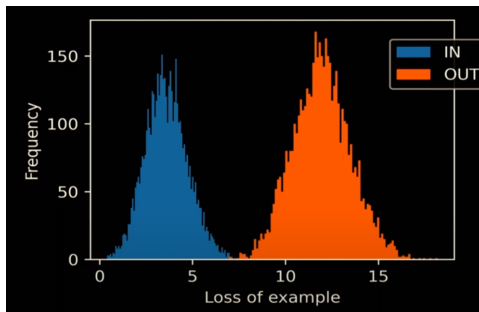
# Overview of the attack

Membership inference attack exploits the observation that

- Machine learning models often **behave differently** on the data that they **were trained** on versus the data that they **see for the first time**.

The objective of the attacker is

- Construct an **attack model** that can **recognize such differences** in the target model's behavior and use them to distinguish members from non-members of the target model's training dataset based solely on the target model's output.



(Carlini, 2022)

# Overview of the attack

Let  $f_{target}$  be the target model, and let  $D_{target}^{train}$  be its private training dataset which contains labeled data records  $(x^{\{i\}}, y^{\{i\}})_{target}$ .

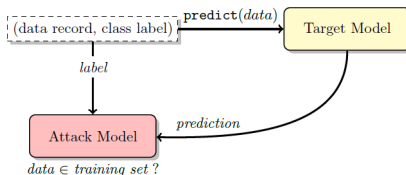


Fig. 1: Membership inference attack in the black-box setting. The

# Overview of the attack

Let  $f_{target}$  be the target model, and let  $D_{target}^{train}$  be its private training dataset which contains labeled data records  $(x^{\{i\}}, y^{\{i\}})_{target}$ .

Let  $f_{attack}()$  be the attack model, and its input  $x_{attack}$  is composed of

- A correctly labeled record
- A prediction vector of size  $c_{target}$

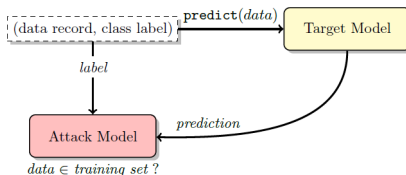


Fig. 1: Membership inference attack in the black-box setting. The

# Overview of the attack

Let  $f_{target}$  be the target model, and let  $D_{target}^{train}$  be its private training dataset which contains labeled data records  $(x^{\{i\}}, y^{\{i\}})_{target}$ .

Let  $f_{attack}()$  be the attack model, and its input  $x_{attack}$  is composed of

- A correctly labeled record
- A prediction vector of size  $c_{target}$

Since the goal of the attack is decisional membership inference, the attack model is a **binary classifier** with two output classes, "in" and "out" or  $Pr\{(x, y) \in D_{target}^{train}\}$ .

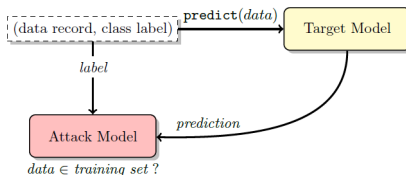


Fig. 1: Membership inference attack in the black-box setting. The



# Shadow Models

To create the attack model's training set, the attack uses **shadow models**.

- Shadow models are created to **behave similarly to the target model**.
- The adversary creates  $k$  shadow models  $f_{shadow}^i()$ .
  - Each shadow model  $i$  is trained on a dataset  $D_{shadow\ i}^{train}$ .

# Shadow Models

To create the attack model's training set, the attack uses **shadow models**.

- Shadow models are created to **behave similarly to the target model**.
- The adversary creates  $k$  shadow models  $f_{shadow}^i()$ .
  - Each shadow model  $i$  is trained on a dataset  $D_{shadow\ i}^{train}$ .
- In contrast to the target model, **we know the ground truth** for each shadow model, i.e., whether a given record was in its training dataset or not.

# Shadow Models

To create the attack model's training set, the attack uses **shadow models**.

- Shadow models are created to **behave similarly to the target model**.
- The adversary creates  $k$  shadow models  $f_{shadow}^i()$ .
  - Each shadow model  $i$  is trained on a dataset  $D_{shadow i}^{train}$ .
- In contrast to the target model, **we know the ground truth** for each shadow model, i.e., whether a given record was in its training dataset or not.
- Approach
  - We query each shadow model with its own training dataset and with a disjoint test set of the same size.
  - The outputs on the training dataset are labeled "in", the rest are labeled "out".
  - The attacker has a dataset  $D_{train}$  of records, the corresponding outputs of the shadow models, and the in/out labels.
  - Split  $D_{train}$  attack into  $c_{target}$  partitions, each associated with a different class label.
  - For each label  $y$ , train a separate attack model.

# Shadow Models

To create the attack model's training set, the attack uses **shadow models**.

- Shadow models are created to **behave similarly to the target model**.
- The adversary creates  $k$  shadow models  $f_{shadow}^i()$ .
  - Each shadow model  $i$  is trained on a dataset  $D_{shadow i}^{train}$ .
- In contrast to the target model, **we know the ground truth** for each shadow model, i.e., whether a given record was in its training dataset or not.
- Approach
  - We query each shadow model with its own training dataset and with a disjoint test set of the same size.
  - The outputs on the training dataset are labeled "in", the rest are labeled "out".
  - The attacker has a dataset  $D_{train}$  of records, the corresponding outputs of the shadow models, and the in/out labels.
  - Split  $D_{train}$  attack into  $c_{target}$  partitions, each associated with a different class label.
  - For each label  $y$ , train a separate attack model.

The shadow models must be **trained in a similar way** to the target model.

- This is easy if the target's training algorithm and model structure are known.
- In MLaaS, the attacker can **use exactly the same service** (e.g., Google Prediction API) to train the shadow model as was used to train the target model.

# Training the attack model

The attack model training set  $D_{attack}^{train}$

- For all  $(x, y) \in D_{shadow_i}^{train}$ , compute the prediction vector  $\hat{y} = f_{shadow}^i(x)$  and add the record  $((\hat{y}, y), in)$  to the attack training set  $D_{attack}^{train}$ .
- For all  $(x, y) \in D_{shadow_i}^{test}$ , compute the prediction vector  $\hat{y} = f_{shadow}^i(x)$  and add the record  $((\hat{y}, y), out)$  to the attack training set  $D_{attack}^{train}$ .

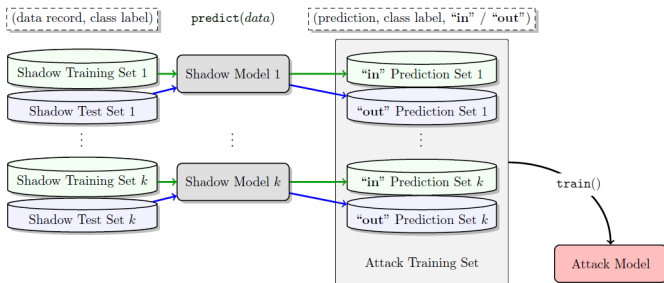


Fig. 3: Training the attack model on the inputs and outputs of the shadow models. For all records in the training dataset of a shadow model, we query the model and obtain the output. These output vectors are labeled “in” and added to the attack model’s training dataset. We also query the shadow model with a test dataset disjoint from its training dataset. The outputs on this set are labeled “out” and also added to the attack model’s training dataset. Having constructed a dataset that reflects the black-box behavior of the shadow models on their training and test datasets, we train a collection of  $c_{target}$  attack models, one per each output class of the target model.

# Shadow Models

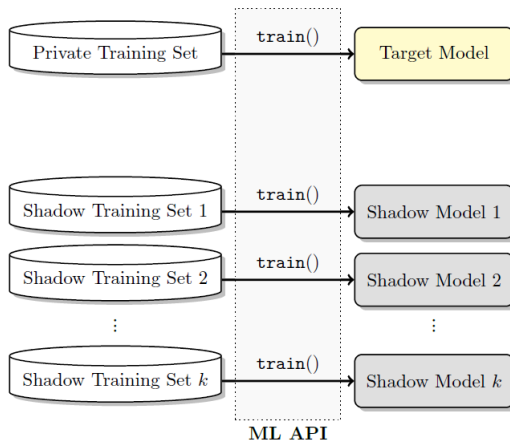


Fig. 2: Training shadow models using the same machine learning platform as was used to train the target model. The training datasets of the target and shadow models have the same format but are disjoint. The training datasets of the shadow models may overlap. All models' internal parameters are trained independently.

# Generating training data for shadow models

## Generating training data for shadow model

- 1 Model-based synthesis
- 2 Statistics-based synthesis
- 3 Noisy real data

## Model-based synthesis

If the attacker does not have real training data nor any statistics about its distribution, he can generate synthetic training data for the shadow models **using the target model** itself.



# Model-based synthesis

If the attacker does not have real training data nor any statistics about its distribution, he can generate synthetic training data for the shadow models **using the target model** itself.

- The intuition is that records that are classified by the target model with **high confidence** should be statistically similar to the target's training dataset.
- The synthesis process runs in two phases
  - 1 search, using a hill-climbing algorithm, the space of possible data records to find inputs that are classified by the target model with high confidence
  - 2 sample synthetic data from these records.

# Model-based synthesis

If the attacker does not have real training data nor any statistics about its distribution, he can generate synthetic training data for the shadow models **using the target model** itself.

- The intuition is that records that are classified by the target model with **high confidence** should be statistically similar to the target's training dataset.
- The synthesis process runs in two phases
  - 1 search, using a hill-climbing algorithm, the space of possible data records to find inputs that are classified by the target model with high confidence
  - 2 sample synthetic data from these records.

It may **not work** if the inputs are **high resolution** images and the target model performs a complex image classification task.

# Model-based synthesis

---

**Algorithm 1** Data synthesis using the target model

---

```
1: procedure SYNTHESIZE(class :  $c$ )
2:    $\mathbf{x} \leftarrow \text{RANDRECORD}()$    ▷ initialize a record randomly
3:    $y_c^* \leftarrow 0$ 
4:    $j \leftarrow 0$ 
5:    $k \leftarrow k_{max}$ 
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27: end procedure
```

---

# Model-based synthesis

---

**Algorithm 1** Data synthesis using the target model
 

---

```

1: procedure SYNTHESIZE(class :  $c$ )
2:    $\mathbf{x} \leftarrow \text{RANDRECORD}()$   $\triangleright$  initialize a record randomly
3:    $y_c^* \leftarrow 0$ 
4:    $j \leftarrow 0$ 
5:    $k \leftarrow k_{max}$ 
6:   for iteration =  $1 \dots iter_{max}$  do
7:      $\mathbf{y} \leftarrow f_{\text{target}}(\mathbf{x})$   $\triangleright$  query the target model
8:     if  $y_c \geq y_c^*$  then  $\triangleright$  accept the record
9:       if  $y_c > \text{conf}_{min}$  and  $c = \arg \max(\mathbf{y})$  then
10:        if  $\text{rand}() < y_c$  then  $\triangleright$  sample
11:          return  $\mathbf{x}$   $\triangleright$  synthetic data
12:        end if
13:      end if
14:
15:
16:
17:   else
18:
19:
20:
21:
22:
23:
24:    $\mathbf{x} \leftarrow \text{RANDRECORD}(\mathbf{x}^*, k)$   $\triangleright$  randomize  $k$  features
25: end for
26: return  $\perp$   $\triangleright$  failed to synthesize
27: end procedure

```

---

# Model-based synthesis

---

**Algorithm 1** Data synthesis using the target model
 

---

```

1: procedure SYNTHESIZE(class :  $c$ )
2:    $\mathbf{x} \leftarrow \text{RANDRECORD}()$   $\triangleright$  initialize a record randomly
3:    $y_c^* \leftarrow 0$ 
4:    $j \leftarrow 0$ 
5:    $k \leftarrow k_{max}$ 
6:   for iteration =  $1 \dots iter_{max}$  do
7:      $\mathbf{y} \leftarrow f_{\text{target}}(\mathbf{x})$   $\triangleright$  query the target model
8:     if  $y_c \geq y_c^*$  then  $\triangleright$  accept the record
9:       if  $y_c > \text{conf}_{min}$  and  $c = \arg \max(\mathbf{y})$  then
10:        if  $\text{rand}() < y_c$  then  $\triangleright$  sample
11:          return  $\mathbf{x}$   $\triangleright$  synthetic data
12:        end if
13:      end if
14:       $\mathbf{x}^* \leftarrow \mathbf{x}$ 
15:       $y_c^* \leftarrow y_c$ 
16:       $j \leftarrow 0$ 
17:    else
18:
19:
20:
21:
22:
23:
24:     $\mathbf{x} \leftarrow \text{RANDRECORD}(\mathbf{x}^*, k)$   $\triangleright$  randomize  $k$  features
25:  end for
26:  return  $\perp$   $\triangleright$  failed to synthesize
27: end procedure

```

---

# Model-based synthesis

---

**Algorithm 1** Data synthesis using the target model
 

---

```

1: procedure SYNTHESIZE(class :  $c$ )
2:    $\mathbf{x} \leftarrow \text{RANDRECORD}()$   $\triangleright$  initialize a record randomly
3:    $y_c^* \leftarrow 0$ 
4:    $j \leftarrow 0$ 
5:    $k \leftarrow k_{\max}$ 
6:   for iteration =  $1 \dots \text{iter}_{\max}$  do
7:      $\mathbf{y} \leftarrow f_{\text{target}}(\mathbf{x})$   $\triangleright$  query the target model
8:     if  $y_c \geq y_c^*$  then  $\triangleright$  accept the record
9:       if  $y_c > \text{conf}_{\min}$  and  $c = \arg \max(\mathbf{y})$  then
10:        if  $\text{rand}() < y_c$  then  $\triangleright$  sample
11:          return  $\mathbf{x}$   $\triangleright$  synthetic data
12:        end if
13:      end if
14:       $\mathbf{x}^* \leftarrow \mathbf{x}$ 
15:       $y_c^* \leftarrow y_c$ 
16:       $j \leftarrow 0$ 
17:    else
18:       $j \leftarrow j + 1$ 
19:      if  $j > \text{rej}_{\max}$  then  $\triangleright$  many consecutive rejects
20:         $k \leftarrow \max(k_{\min}, \lceil k/2 \rceil)$ 
21:         $j \leftarrow 0$ 
22:      end if
23:    end if
24:     $\mathbf{x} \leftarrow \text{RANDRECORD}(\mathbf{x}^*, k)$   $\triangleright$  randomize  $k$  features
25:  end for
26:  return  $\perp$   $\triangleright$  failed to synthesize
27: end procedure

```

---

## Statistics-based synthesis

The attacker may have some **statistical information** about the population from which the target model's training data was drawn.

- Generating synthetic training records for the shadow models by **independently sampling the value of each feature** from its own marginal distribution.

## Noisy real data

The attacker may have access to some data that is **similar to the target model's training data** and can be considered as a “noisy” version.

- Flipping the (binary) values of 10% or 20% randomly selected features, then training our shadow models on the resulting noisy dataset.



# Evaluation

## Datasets

- CIFAR-10 and CIFAR-100
- **Purchases:** The classification task is to predict the purchase style of a user given the 600-feature vector. 10000 randomly records are used to train the target model. {2; 10; 20; 50; 100} classes.
- Location: 446 binary features and 1600 training data into 30 classes.
- Texas hospital stays: The resulting dataset has 67330 records and 6170 binary features. 10000 randomly records are selected to train the target model.
- MNIST
- UCI Adult: This dataset includes 48842 records with 14 attributes. The (binary) classification task is to predict if a person makes over \$50K a year. 10000 randomly records are used to train the target model.

# Evaluation

## Datasets

- CIFAR-10 and CIFAR-100
- **Purchases:** The classification task is to predict the purchase style of a user given the 600-feature vector. 10000 randomly records are used to train the target model. {2; 10; 20; 50; 100} classes.
- Location: 446 binary features and 1600 training data into 30 classes.
- Texas hospital stays: The resulting dataset has 67330 records and 6170 binary features. 10000 randomly records are selected to train the target model.
- MNIST
- UCI Adult: This dataset includes 48842 records with 14 attributes. The (binary) classification task is to predict if a person makes over \$50K a year. 10000 randomly records are used to train the target model.

## Target models

- Two implemented on cloud-based **machine learning as a service** platforms, Google Prediction API and Amazon ML
  - Purchases, Texas hospital stays, locations, Adult, and MNIST
- One implemented locally
  - Purchases, CIFAR10, and CIFAR100

# Shadow Models

The number of shadow models is

- 100 for the CIFAR datasets
- 20 for the purchase dataset
- 10 for the Texas hospital stay dataset
- 60 for the location dataset
- 50 for the MNIST dataset
- 20 for the Adult dataset

Increasing the number of shadow models would **increase the accuracy** of the attack but also its **cost**.

## Accuracy of the attack

The attack is evaluated by executing it on randomly reshuffled records from **the target's training and test datasets** and they have the same size.

- Hence the baseline accuracy is  $\frac{1}{2}$ .

# Evaluation

- The test accuracy of our target neural-network models with the largest training datasets (15000 and 29540 records, respectively) is 60% and 20% for CIFAR-10 and CIFAR-100, respectively.
- Use real data to train shadow models

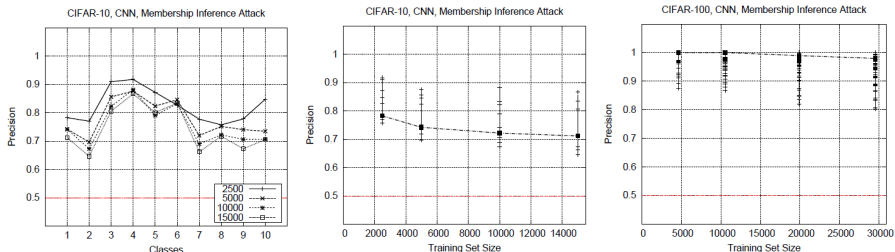
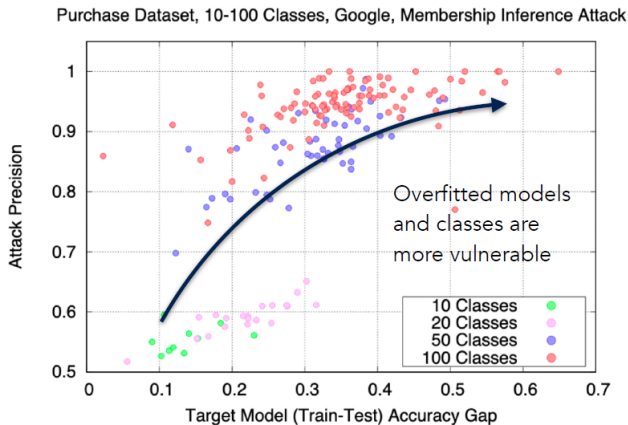


Fig. 4: Precision of the membership inference attack against neural networks trained on CIFAR datasets. The graphs show precision for different classes while varying the size of the training datasets. The median values are connected across different training set sizes. The median precision (from the smallest dataset size to largest) is 0.78, 0.74, 0.72, 0.71 for CIFAR-10 and 1, 1, 0.98, 0.97 for CIFAR-100. Recall is almost 1 for both datasets. The figure on the left shows the per-class precision (for CIFAR-10). Random guessing accuracy is 0.5.

# Privacy Leakage due to Overfitting



# Purchase Dataset

<i>ML Platform</i>	<i>Training</i>	<i>Test</i>
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

TABLE I: Training and test accuracy of the models constrained by different ML-as-a-service platforms on the purchase dataset (classes).

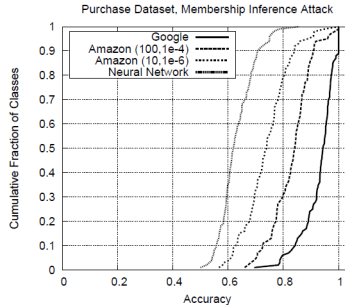


Fig. 7: Precision of the membership inference attack against models trained on the same datasets but using different platforms. The attack model is a neural network.

**Overfitting is not the only factor** that causes a model to be vulnerable to membership inference. The structure and type of the model also contribute to the problem.

# Purchase Dataset

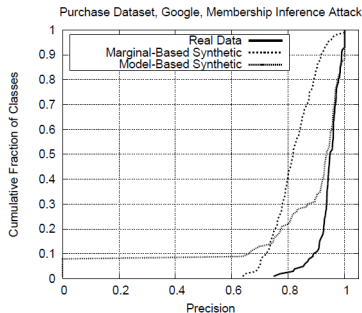


Fig. 9: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the purchase dataset. Results are shown for different ways of generating training data for the shadow models (real, synthetic generated from the target model, synthetic generated from marginal 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.



# Overfitting vs. attack success

<i>Dataset</i>	<i>Training Accuracy</i>	<i>Testing Accuracy</i>	<i>Attack Precision</i>
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

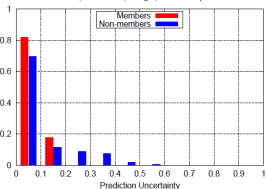
TABLE II: Accuracy of the Google-trained models and the corresponding attack precision.

# Classification uncertainty

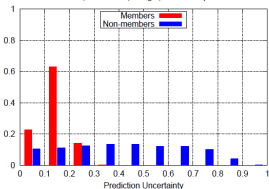
Prediction uncertainty is the **normalized entropy** of the model's prediction vector

- $\frac{-1}{\log(n)} \sum_i p_i \log(p_i)$  where  $p_i$  is the probability that the input belongs to class  $i$ , and  $n$  is the number of classes.

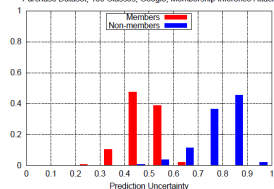
Purchase Dataset, 10 Classes, Google, Membership Inference Attack



Purchase Dataset, 20 Classes, Google, Membership Inference Attack



Purchase Dataset, 100 Classes, Google, Membership Inference Attack



# The accuracy of the target models with different mitigation techniques

<b>Purchase dataset</b>	<i>Testing Accuracy</i>	<i>Attack Total Accuracy</i>	<i>Attack Precision</i>	<i>Attack Recall</i>
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

- Label only: The attack can still exploit the mislabeling behavior of the target model because members and non-members of the training dataset are mislabeled differently (assigned to different wrong classes).

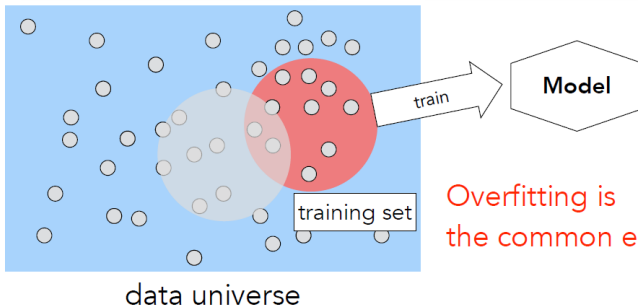
# Privacy vs. Learning

## Privacy

Does the model leak information about data in the training set?

## Learning

Does the model generalize to data outside the training set?



Overfitting is the common enemy!

(Shokri, 2020)