



Privacy Risks in Machine Learning

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

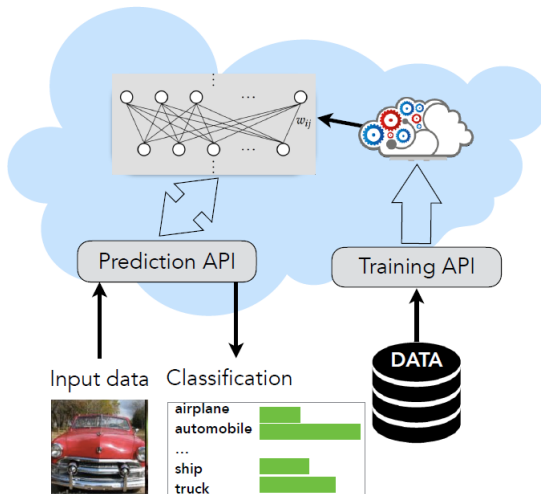
1 Recap

2 White-box Inference Attacks

Recap

Machine Learning as a Service

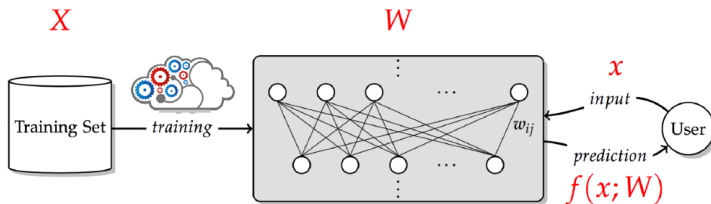
Machine Learning as a Service



(Shokri, 2020)

Privacy Risks in Machine Learning

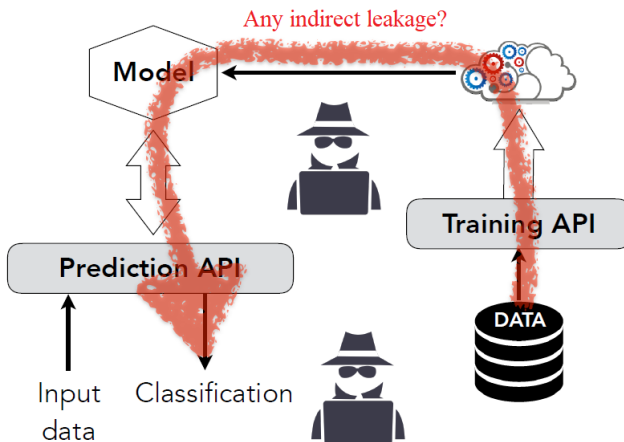
- What is training data leakage? Inferring information about members of X , beyond what can be learned about its underlying distribution.



(Shokri, 2020)

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:

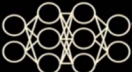

Was  trained
on the example  ?

(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)

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.

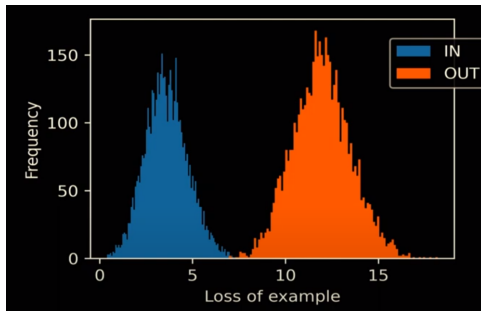
Overview of the attack

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 $(x^{\{i\}}, y^{\{i\}})_{target}$.

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 $Pr\{(x, y) \in D_{target}^{train}\}$.

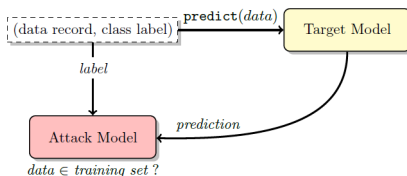


Fig. 1: Membership inference attack in the black-box setting.

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} .

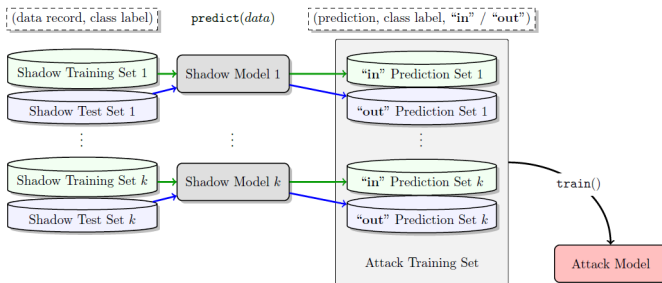


Fig. 3: Training the attack model on the inputs and outputs of the shadow models. For all records in the training dataset of a shadow model, we query the model and obtain the output. These output vectors are labeled “in” and added to the attack model’s training dataset. We also query the shadow model 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

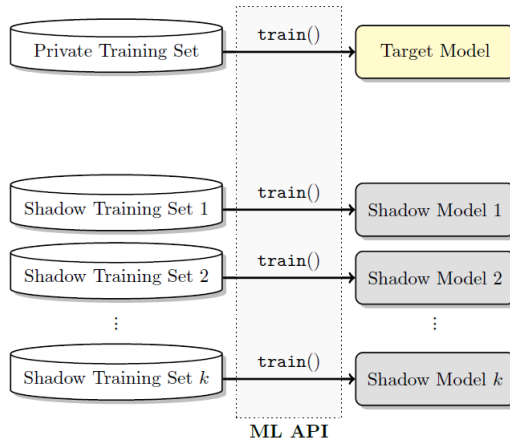


Fig. 2: Training shadow models using the same machine learning platform as was used to train the target model. The training datasets of the target and shadow models have the same format but are disjoint. The training datasets of the shadow models may overlap. All models' internal parameters are trained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

White-box Inference Attacks

White-box Inference Attacks

2019 IEEE Symposium on Security and Privacy

Comprehensive Privacy Analysis of Deep Learning

Passive and Active White-box Inference Attacks against Centralized and Federated Learning

Milad Nasr

University of Massachusetts Amherst
milad@cs.umass.edu

Reza Shokri

National University of Singapore
reza@comp.nus.edu.sg

Amir Houmansadr

University of Massachusetts Amherst
amir@cs.umass.edu

Abstract

The attack measures the privacy leakage through

- The parameters of fully trained models
- The parameter updates of models during training

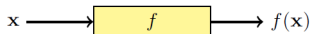
It proposes the inference algorithms for both **centralized** and **federated learning**, with respect to passive and active inference attackers.

White-box inference attacks that exploit the privacy vulnerabilities of the **stochastic gradient descent** (SGD) algorithm.

Black-box vs. White-box Inference

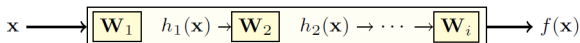
Black-box

- The attacker can **obtain the prediction vector** $f(x)$ on arbitrary input x , but cannot access the model parameters, nor the intermediate computations of $f(x)$.



White-box

- The attacker has access to the full model $f(x; W)$, notably its **architecture and parameters** W , and any hyper-parameter that is needed to use the model for predictions. Thus, he can also **observe the intermediate computations** at hidden layers $h_i(x)$.



Black-box vs. White-box Inference

Let $L(f(x; W), y)$ be the loss function for the classification model f .

- During the training, the SGD algorithm minimizes the empirical expectation of the loss function over the training set \mathcal{D} :

$$\min_W \mathbb{E}_{(x,y) \sim \mathcal{D}} [L(f(x; W), y)]$$

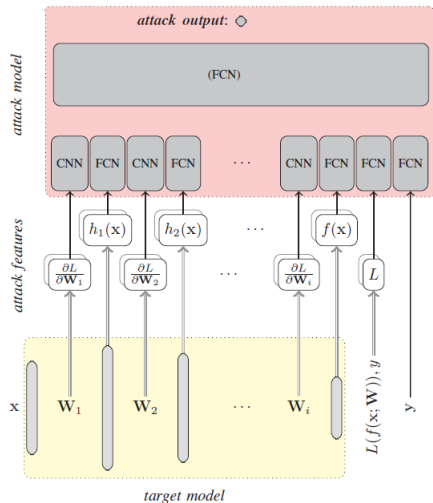
Thus, for any data record in the training dataset, **the gradient of the loss** $\frac{\partial L}{\partial W}$ over the data record **is pushed towards zero**, after each round of training.

- This can be exploited to extract information about a model's training data.

Inference model

The distinct inputs to the attack model

- 1 The set of gradients for different layers
- 2 The set of activation vectors for different layers
- 3 The model output
- 4 The one-hot encoding of the label
- 5 The loss of the model on the target data

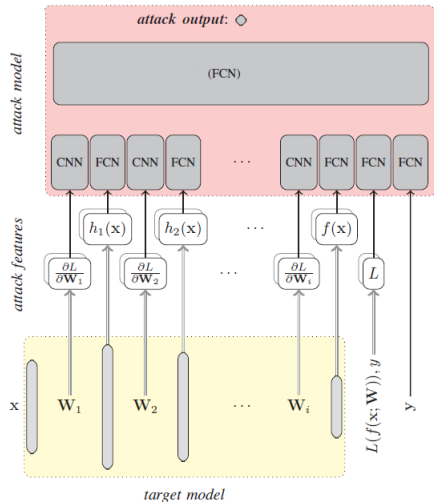


Inference model

The distinct inputs to the attack model

- 1 The set of gradients for different layers
- 2 The set of activation vectors for different layers
- 3 The model output
- 4 The one-hot encoding of the label
- 5 The loss of the model on the target data

Each of these are separately fed into the attack model, and are analyzed separately using independent components. The output of components is **concatenated** together.



Inference model

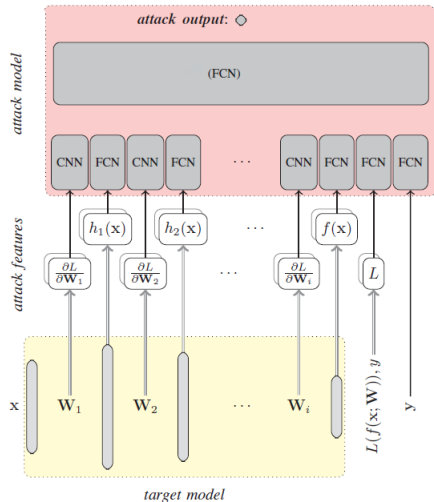
The distinct inputs to the attack model

- 1 The set of gradients for different layers
- 2 The set of activation vectors for different layers
- 3 The model output
- 4 The one-hot encoding of the label
- 5 The loss of the model on the target data

Each of these are separately fed into the attack model, and are analyzed separately using independent components. The output of components is **concatenated** together.

The output of the encoder is a **single score**, which is the output of the attack.

- This score (in the supervised attack training) predicts the membership probability of the input data.



Stand-alone vs. Federated Learning

The training data is available

- 1 All in one place (i.e., stand-alone centralized training)
- 2 Distributed among multiple parties who do not trust each other (i.e., federated learning)

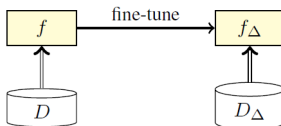
Stand-alone vs. Federated Learning

The training data is available

- 1 All in one place (i.e., stand-alone centralized training)
- 2 Distributed among multiple parties who do not trust each other (i.e., federated learning)

Stand-alone

- The attacker observes the final target model f , after the training is done (e.g., in a centralized manner) using dataset D .
- He might also observe the updated model f_{Δ} after it has been updated (fine-tuned) using a new dataset D_{Δ} .



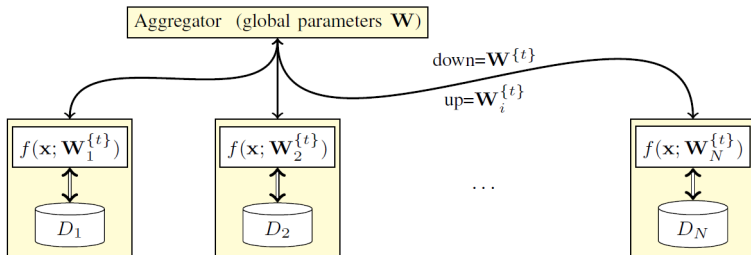
Stand-alone vs. Federated Learning

The training data is available

- 1 All in one place (i.e., stand-alone centralized training)
- 2 Distributed among multiple parties who do not trust each other (i.e., federated learning)

Federated

- The attacker could be the **central aggregator**, who observes individual updates over time and can **control the view of the participants on the global parameters**.
- He could also be **any of the participants** who can observe the global parameter updates, and can control his parameter uploads.



Stand-alone fine-tuning

A model f is trained on dataset \mathcal{D} . At a later stage it is updated to f_{Δ} after being **fine-tuned using a new dataset \mathcal{D}_{Δ}** .

- If the attacker observes the final outcome, we want to measure the information leakage of the final model f_{Δ} about the **whole training set $\mathcal{D} \cup \mathcal{D}_{\Delta}$** .
- The attacker might also be interested only in recovering information about the **new set \mathcal{D}_{Δ}** .
 - This is very relevant in numerous cases where the original model is trained using some unlabeled (and perhaps public) data, and then it is **fine-tuned using sensitive private labeled data**.

Stand-alone fine-tuning

A model f is trained on dataset \mathcal{D} . At a later stage it is updated to f_{Δ} after being **fine-tuned using a new dataset \mathcal{D}_{Δ}** .

- If the attacker observes the final outcome, we want to measure the information leakage of the final model f_{Δ} about the **whole training set $\mathcal{D} \cup \mathcal{D}_{\Delta}$** .
- The attacker might also be interested only in recovering information about the **new set \mathcal{D}_{Δ}** .
 - This is very relevant in numerous cases where the original model is trained using some unlabeled (and perhaps public) data, and then it is **fine-tuned using sensitive private labeled data**.

The model for inference attacks against fine-tuned models is a **special case** of our membership inference model for attacking **federated learning**.

- In both cases, the attacker observes multiple versions of the target model.

Federated Learning

In this setting, N **participants, who have different training sets \mathcal{D}_i** , agree on a single deep learning task and model architecture to **train a global model**.

- A **central server** keeps the latest version of the parameters W for the global model.
- **Each participant has a local model**, hence a local set of parameters W_i .

Federated Learning

In this setting, N **participants, who have different training sets \mathcal{D}_i** , agree on a single deep learning task and model architecture to **train a global model**.

- A **central server** keeps the latest version of the parameters W for the global model.
- **Each participant has a local model**, hence a local set of parameters W_i .

In each epoch of training

- Each participant
 - Downloads the global parameters W
 - Updates them locally using SGD algorithm on their local training data \mathcal{D}_i
 - Uploads new weights W_i back to the server

Federated Learning

In this setting, N **participants, who have different training sets** \mathcal{D}_i , agree on a single deep learning task and model architecture to **train a global model**.

- A **central server** keeps the latest version of the parameters W for the global model.
- **Each participant has a local model**, hence a local set of parameters W_i .

In each epoch of training

- Each participant
 - Downloads the global parameters W
 - Updates them locally using SGD algorithm on their local training data \mathcal{D}_i
 - Uploads new weights W_i back to the server
- The parameter server computes the **average** value for each parameter using the uploaded parameters by all participants.

Federated Learning

In this setting, N **participants, who have different training sets \mathcal{D}_i** , agree on a single deep learning task and model architecture to **train a global model**.

- A **central server** keeps the latest version of the parameters W for the global model.
- **Each participant has a local model**, hence a local set of parameters W_i .

In each epoch of training

- Each participant
 - Downloads the global parameters W
 - Updates them locally using SGD algorithm on their local training data \mathcal{D}_i
 - Uploads new weights W_i back to the server
- The parameter server computes the **average** value for each parameter using the uploaded parameters by all participants.

This collaborative training **continues until the global model converges**.

Federated Learning

There are two possibilities for the position of the attacker in federated learning. The adversary can be

- The centralized parameter server
- One of the participants

Federated Learning

There are two possibilities for the position of the attacker in federated learning. The adversary can be

- The centralized parameter server
- One of the participants

The centralized parameter server

- A curious parameter server can receive updates from each individual participant over time $W_i^{\{t\}}$, and use them to infer information about the training set of each participant.

Federated Learning

There are two possibilities for the position of the attacker in federated learning. The adversary can be

- The centralized parameter server
- One of the participants

The centralized parameter server

- A curious parameter server can receive updates from each individual participant over time $W_i^{\{t\}}$, and use them to infer information about the training set of each participant.

One of the participants

- An adversarial participant can only observe the global parameters over time $W^{\{t\}}$, and craft his own adversarial parameter updates $W_i^{\{t\}}$ to gain more information about the union of the training data of all other participants.

Inference Model

The adversary can try to run an independent membership inference attack on each of these models $W_i^{\{t\}}$, and then combine their results.

- This, however, might not **capture the dependencies between parameter values** over time, which can leak information about the training data.

Inference Model

The adversary can try to run an independent membership inference attack on each of these models $W_i^{\{t\}}$, and then combine their results.

- This, however, might not **capture the dependencies between parameter values** over time, which can leak information about the training data.

Instead, in the design it is used a single inference model, where each attack component (e.g., components for gradients of layer i) **processes all of its corresponding inputs over the observed models at once**.

- For example, for the attack component that analyzes the loss value L , the input dimension can be $1 \times T$, if the adversary observes T versions of the target model over time.

Inference Model

The adversary can try to run an independent membership inference attack on each of these models $W_i^{\{t\}}$, and then combine their results.

- This, however, might not **capture the dependencies between parameter values** over time, which can leak information about the training data.

Instead, in the design it is used a single inference model, where each attack component (e.g., components for gradients of layer i) **processes all of its corresponding inputs over the observed models at once**.

- For example, for the attack component that analyzes the loss value L , the input dimension can be $1 \times T$, if the adversary observes T versions of the target model over time.

The output of the attack component is also T times larger than the case of attacking a stand-alone model.

Inference Model

The adversary can try to run an independent membership inference attack on each of these models $W_i^{\{t\}}$, and then combine their results.

- This, however, might not **capture the dependencies between parameter values** over time, which can leak information about the training data.

Instead, in the design it is used a single inference model, where each attack component (e.g., components for gradients of layer i) **processes all of its corresponding inputs over the observed models at once**.

- For example, for the attack component that analyzes the loss value L , the input dimension can be $1 \times T$, if the adversary observes T versions of the target model over time.

The output of the attack component is also T times larger than the case of attacking a stand-alone model.

Finally, these correlated outputs, of all attack components, are processed all at once by the inference model.

Passive vs. Active Inference Attack

Passive adversary

- The attacker can **only observe** the genuine computations by the training algorithm and the model.

Active adversary

- The adversary, who is participating in the training process, **can actively influence the target model** in order to **extract more information** about its training set.

Passive vs. Active Inference Attack

Passive adversary

- The attacker can **only observe** the genuine computations by the training algorithm and the model.

Active adversary

- The adversary, who is participating in the training process, **can actively influence the target model** in order to **extract more information** about its training set.
 - The attacker could be **one of the participants** in the federated learning, who adversarially modifies his parameter uploads $W_i^{\{t\}}$.
 - The attacker could be the **central aggregator** who adversarially modifies the aggregate parameters $W^{\{t\}}$ which he sends to the target participant(s).

Active attack strategy

Let x be a data record, which is targeted by the adversary to determine its membership, and the adversary is one of the participants.

- The attacker runs a **gradient ascent** on x , and updates its local model parameters in the direction of increasing the loss on x .

$$W \leftarrow W + \lambda \frac{\partial L^x}{\partial W}$$

where λ is the adversarial update rate.

- The adversary then uploads the adversarially computed parameters to the central server, who will aggregate them with the parameter updates from other participants.

Active attack strategy

Let x be a data record, which is targeted by the adversary to determine its membership, and the adversary is one of the participants.

- The attacker runs a **gradient ascent** on x , and updates its local model parameters in the direction of increasing the loss on x .

$$W \leftarrow W + \lambda \frac{\partial L^x}{\partial W}$$

where λ is the adversarial update rate.

- The adversary then uploads the adversarially computed parameters to the central server, who will aggregate them with the parameter updates from other participants.

If the target record x is in the training set of a participant, its local **SGD algorithm abruptly reduces the gradient of the loss on x** .

- This can be detected by the inference model, and be used to distinguish members from non-members.

Supervised vs. Unsupervised Inference

Supervised

- The adversary has a dataset D' that **overlaps** with the target dataset D .
- Given dataset D' , he can train the attack model in a supervised way, and use it to attack the rest of the training dataset.
- In the supervised setting, the (mean square) loss is minimized for predicting the membership of the data points in its training set D' :

$$\sum_{d \in \{D' \cap D\}} (h(d) - 1)^2 + \sum_{d \in \{D' / D\}} (h(d))^2$$

where h is the inference attack model.

Supervised vs. Unsupervised Inference

Supervised

- The adversary has a dataset D' that **overlaps** with the target dataset D .
- Given dataset D' , he can train the attack model in a supervised way, and use it to attack the rest of the training dataset.
- In the supervised setting, the (mean square) loss is minimized for predicting the membership of the data points in its training set D' :

$$\sum_{d \in \{D' \cap D\}} (h(d) - 1)^2 + \sum_{d \in \{D' / D\}} (h(d))^2$$

where h is the inference attack model.

Unsupervised

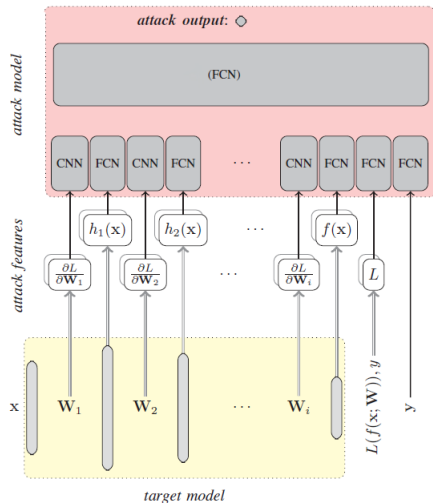
- If the adversary **does not have known samples from the target training set**, there are two possibilities for training the inference attack models
 - supervised training on **shadow models**
 - **Unsupervised training** of inference models

Unsupervised training of inference models

The assumption for this attack is that the attacker has access to a dataset D' which **partially overlaps** with the target training set D , however, **the adversary does not know which data points are in $D' \cap D$.**

- The objective is to find a score for each data point that represents its **embedding in a space**, which helps us easily separating members from non-members (using **clustering** algorithms).
- An encoder-decoder architecture is used to achieve this goal.

Unsupervised training of inference models

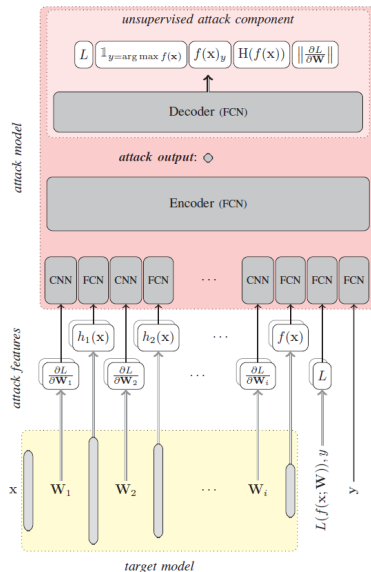


Unsupervised training of inference models

The objective of the decoder is to **reconstruct some key features** of the attack input which are important for membership inference.

- 1 The loss value L
- 2 Whether the target model has predicted the correct label $\mathbb{1}_{y=\arg\max f(x)}$
- 3 The confidence of the model on the correct label $f(x)_y$
- 4 The prediction uncertainty (entropy) of the model $H(f(x))$
- 5 The norm of the gradients $\|\frac{\partial L}{\partial W}\|$

These features are **strong signals** for distinguishing members from non-members.



Unsupervised training of inference models

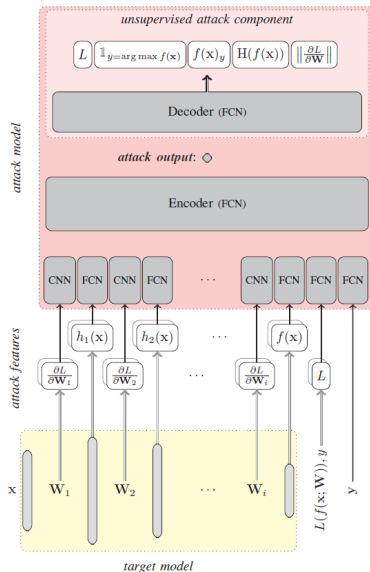
The objective of the decoder is to **reconstruct some key features** of the attack input which are important for membership inference.

- 1 The loss value L
- 2 Whether the target model has predicted the correct label $\mathbb{1}_{y=\arg\max f(x)}$
- 3 The confidence of the model on the correct label $f(x)_y$
- 4 The prediction uncertainty (entropy) of the model $H(f(x))$
- 5 The norm of the gradients $\|\frac{\partial L}{\partial W}\|$

These features are **strong signals** for distinguishing members from non-members.

A **clustering algorithm** is used to cluster each input of the target model in two clusters based on the outputs of the encoder.

- The cluster with the **larger gradient norm as non-members**.



Evaluation

Datasets

- CIFAR100
- Purchase100: contains 600 different products and each user has a binary record which indicates whether she has bought each of the products (a total of 197, 324 data records). It has 100 classes based on the similarity of the purchases.
- Texas100: contains 67, 330 records and 6, 170 binary features.

In federated learning, the same training dataset size is used for all parties, and each party's training data is selected uniformly at random from our available datasets.

TABLE II: Size of datasets used for training and testing the target classification model and the membership inference model

Datasets	Target Model		Inference Attack Model			
	Training	Test	Training Members	Training Non-members	Test Members	Test Non-members
CIFAR100	50,000	10,000	25,000	5,000	5,000	5,000
Texas100	10,000	70,000	5,000	10,000	10,000	10,000
Purchase100	20,000	50,000	10,000	10,000	10,000	10,000

Impact of the outputs of different layers

Among the layer outputs, the last layer (model output) leaks the most membership information about the training data.

- The first layers extract simple features from the input, thus generalize much better compared to the last layers, which are responsible for complex task of finding the relationship between abstract features and the classes.

TABLE III: Attack accuracy using the outputs of individual activation layers. Pre-trained Alexnet on CIFAR100, stand-alone setting.

Output Layer	Attack Accuracy
Last layer (prediction vector)	74.6% (black-box)
Second to last	74.1%
Third to last	72.9%
Last three layers, combined	74.6%

Impact of gradients

The gradient of the later layers leak more membership information

TABLE IV: Attack accuracy when we apply the attack using parameter gradients of different layers. (CIFAR100 dataset with Alexnet architecture, stand-alone scenario)

Gradient w.r.t.	Attack Accuracy
Last layer parameters	75.1%
Second to last layer parameters	74.6%
Third to last layer parameters	73.5%
Forth to last layer parameters	72.6%
Parameters of last four layers, combined	75.15%

Impact of the gradient norm

Last-layer gradient norms over consecutive training epochs for member and non-member instances (for the Purchase100 dataset).

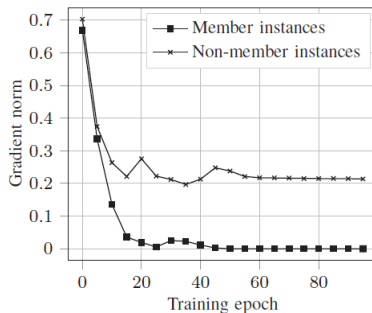


Fig. 3: Gradient norms of the last layer during learning epochs for member and non-member instances (for Purchase100).

Impact of gradients

Gradients leak significantly **more** membership information about the training set, compared to the layer outputs

The **large capacity of the model** which enables it to learn complex tasks and generalize well, leads to also memorizing more information about the training data.

- The total number of the parameters in pre-trained Densenet model is 25.62M , whereas this is only 1.7M parameters for ResNet.

TABLE VIII: The attack accuracy for different datasets and different target architectures using layer outputs versus gradients. This is the result of analyzing the stand-alone scenario, where the CIFAR100 models are all obtained from pre-trained online repositories.

Target Model				Attack Accuracy		
Dataset	Architecture	Train Accuracy	Test Accuracy	Black-box	White-box (Outputs)	White-box (Gradients)
CIFAR100	Alexnet	99%	44%	74.2%	74.6%	75.1%
CIFAR100	ResNet	89%	73%	62.2%	62.2%	64.3%
CIFAR100	DenseNet	100%	82%	67.7%	67.7%	74.3%
Texas100	Fully Connected	81.6%	52%	63.0%	63.3%	68.3%
Purchase100	Fully Connected	100%	80%	67.6%	67.6%	73.4%

Stand-Alone Setting: Unsupervised Attacks

TABLE VII: Accuracy of our unsupervised attack compared to the Shadow models approach [6] for the white-box scenario.

Dataset	Arch	(Unsupervised) Attack Accuracy	(Shadow Models) Attack Accuracy
CIFAR100	Alexnet	75.0%	70.5%
CIFAR100	DenseNet	71.2%	64.2%
CIFAR100	ResNet	63.1%	60.9%
Texas100	Fully Connected	66.3%	65.3%
Purchase100	Fully Connected	71.0%	68.2%

Attacking Fine-Tuned Models

60% of the train dataset as D and the rest for D_{Δ} .

TABLE IX: Attack accuracy on fine-tuned models. D is the initial training set, D_{Δ} is the new dataset used for fine-tuning, and \bar{D} is the set of non-members (which is disjoint with D and D_{Δ}).

Dataset	Architecture	Train Acc.	Test Acc.	Distinguish D from D_{Δ}	Distinguish D from \bar{D}	Distinguish D_{Δ} from \bar{D}
CIFAR100	Alexnet	100.0%	39.8%	62.1%	75.4%	71.3%
CIFAR100	DenseNet	100.0%	64.3%	63.3%	74.6%	71.5%
Texas100	Fully Connected	95.2%	48.6%	58.4%	68.4%	67.2%
Purchase100	Fully Connected	100.0%	77.5%	64.4%	73.8%	71.2%

Federated Learning Settings: The Passive Parameter Aggregator

The adversary (The parameter aggregator) can passively collect all the parameter updates from all of the participants, at each epoch, and can perform the membership inference attack against each of the target participants, separately.

- Using later epochs, substantially increases the attack accuracy.

TABLE XII: The accuracy of the passive global attacker in the federated setting when the attacker uses various training epochs. (CIFAR100-Alexnet)

Observed Epochs	Attack Accuracy
5, 10, 15, 20, 25	57.4%
10, 20, 30, 40, 50	76.5%
50, 100, 150, 200, 250	79.5%
100, 150, 200, 250, 300	85.1%

Federated Learning Settings: The Passive Local Attacker

The goal of the attacker is to learn if a target input has been a member of the training data of **any other participants**.

- A local attack has a lower accuracy compared to the global attack (The parameter aggregator)
- The accuracy of the local attacker **degrades for larger numbers of participants**.

TABLE XIII: The accuracy of the passive local attacker for different numbers of participants. (CIFAR100-Alexnet)

Number of Participants	Attack Accuracy
2	89.0%
3	78.1%
4	76.7%
5	67.2%

Federated Learning

New scenario: the adversary (the parameter aggregator) isolates the target participant and segregates the target participant’s learning process.

- Isolating a target participant, and creating a local view of the network for it.
- Isolating the training of a target model significantly increases the attack accuracy.

TABLE X: Attack accuracy in the federated learning setting. There are 4 participants. A global attacker is the central parameter aggregator, and the local attacker is one of the participants. The global attacker performs the inference against each individual participant, and we report the average attack accuracy. The local attacker performs the inference against all other participants. The passive attacker follows the protocol and only observes the updates. The active attacker changes its updates, or (in the case of a global attack) isolates one participant by not passing the updates of other participants to it, in order to increase the information leakage.

Target Model		Global Attacker (the parameter aggregator)				Local Attacker (a participant)	
		Passive	Active			Passive	Active
Dataset	Architecture		Gradient Ascent	Isolating	Isolating Gradient Ascent		Gradient Ascent
CIFAR100	Alexnet	85.1%	88.2%	89.0%	92.1%	73.1%	76.3%
CIFAR100	DenseNet	79.2%	82.1%	84.3%	87.3%	72.2%	76.7%
Texas100	Fully Connected	66.4%	69.5%	69.3%	71.7%	62.4%	66.4%
Purchase100	Fully Connected	72.4%	75.4%	75.3%	82.5%	65.8%	69.8%