University of Idaho

Department of Computer Science

CS 487/587 Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 11

Privacy Attacks against Machine Learning Models

Lecture Outline

- Privacy attacks in AML
  - Categories of privacy attacks
    - Membership inference attack
    - Feature inference attack
    - Model extraction attack
  - Causes of privacy leaks
- Presentation by Sabiha Jannath Tisha
  - Shokri (2018) Membership Inference Attacks Against Machine Learning Models
- Attacks against distributed learning
- Privacy Attacks against LLMs

Privacy Attacks in AML

Privacy Attacks in AML

- ***Privacy attacks*** are also referred to as inference attacks or confidentiality attacks
- They can broadly be developed against:
  - Training data
    - E.g., reveal the identity of patients whose data was used for training a model
  - ML model
    - E.g., reveal the architecture and parameters of a model that is used by an insurance company for predicting insurance rates
    - E.g., reveal the model used by a financial institution for credit card approval
- Privacy attacks are commonly divided into the following main categories
  - Membership inference attack
  - Feature inference attack
  - Model extraction attack

Membership Inference Attack

Categories of Privacy Attacks

- ***Membership inference attack***
  - Adversarial goal: determine whether or not an individual data instance x^* is part of the training dataset $\mathcal{D}$ for a model
- The attack typically assumes black-box query access to the model
- Attacks on both supervised classification models and generative models (GANs, VAEs) have been demonstrated
- A common approach is to first train several ***shadow models*** that imitate the behavior of the target model, and use the prediction vectors of the shadow models for training a binary classifier (that infers the membership)

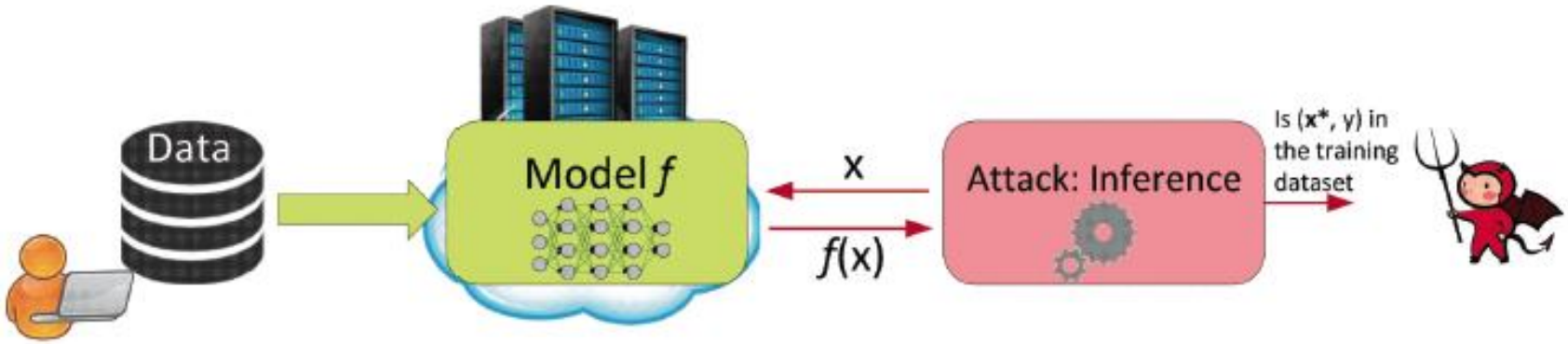


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Feature Inference Attack

Categories of Privacy Attacks

- ***Feature inference attack***
  - Adversarial goal: recreate certain features of a data instance x^* or statistical properties (such as average value for a class) from the training dataset $\mathcal{D}$ for a model
- A.k.a. attribute inference attack, reconstruction attack, or data extraction attack
- Various attacks were developed to either recover partial information about the training data (such as sensitive features of the dataset, or typical representatives of specific classes in the dataset) or full data samples
  - An example of a training data extraction attack is described later in this lecture
- Similarly, recreating dataset properties that were not encoded in the dataset is also referred to as ***property inference attack***
  - E.g., extract information about the ratio of men and women in a patient dataset, despite that gender information was not provided in the training records

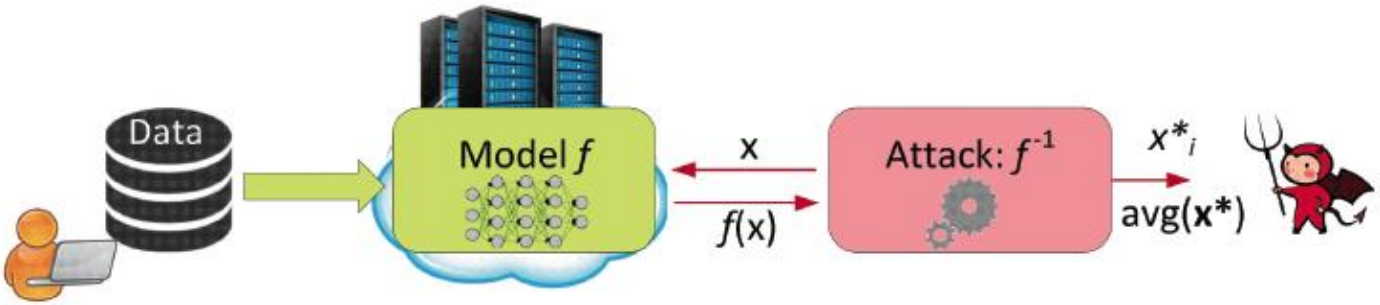


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Model Inversion Attack

Feature Inference Attack

- Fredrickson (2015) Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures
- ***Model inversion attack*** creates prototype examples for the classes in the dataset
  - The authors demonstrated an attack against a DNN model for face recognition
  - Given a person's name and white-box access to the model, the attack reverse-engineered the model and produced an averaged image of that person
    - The obtained averaged image (left image below) makes the person recognizable
  - This attack is limited to classification models where the classes pertain to one type of object (such as faces of the same person)

Recovered image using the model inversion attack

Image of the person used for training the model

Model Inversion Attack

Feature Inference Attack

- The model inversion attack applies gradient descent to start from a given label, and follow the gradient in a trained network to recreate an image for that label
  - In the algorithm, c denotes the cost function, whereas the PROCESS function applies image denoising and sharpening operations to improve the reconstructed image
- Model inversion attack can be used for potential breaches where the adversary, given some access to the model, can infer features that characterize each class

Algorithm 1 Inversion attack for facial recognition models.

function MI-FACE($label, \alpha, \beta, \gamma, \lambda$)
  $c(\mathbf{x}) \stackrel{\text{def}}{=} 1 - \tilde{f}_{label}(\mathbf{x}) + \text{AUXTERM}(\mathbf{x})$
  $\mathbf{x}_0 \leftarrow \mathbf{0}$
  for $i \leftarrow 1 \ldots \alpha$ **do**
    $\mathbf{x}_i \leftarrow \text{PROCESS}(\mathbf{x}_{i-1} - \lambda \cdot \nabla c(\mathbf{x}_{i-1}))$
    if $c(\mathbf{x}_i) \geq \max(c(\mathbf{x}_{i-1}), \ldots, c(\mathbf{x}_{i-\beta}))$ **then**
      break
    if $c(\mathbf{x}_i) \leq \gamma$ **then**
      break
  return $[\arg\min_{\mathbf{x}_i}(c(\mathbf{x}_i)), \min_{\mathbf{x}_i}(c(\mathbf{x}_i))]$

Model Extraction Attack

Categories of Privacy Attacks

- ***Model extraction attack***
  - Adversarial goal: reconstruct an approximated model $f'(x)$ of the target model $f(x)$
- A.k.a. model inference attack
- The approximated function $f'(x)$ will act as a substitute model and produce similar predicted outputs as the target model
  - The adversary has black-box query access to the model
  - The goal is to "steal" the model and use the substitute model for lunching other attacks, such as synthesis of adversarial examples, or membership inference attacks
- Besides creating a substitute model, several works focused on recovering the hyperparameters of the model, such as the number of layers, optimization algorithm, activation types used, etc.

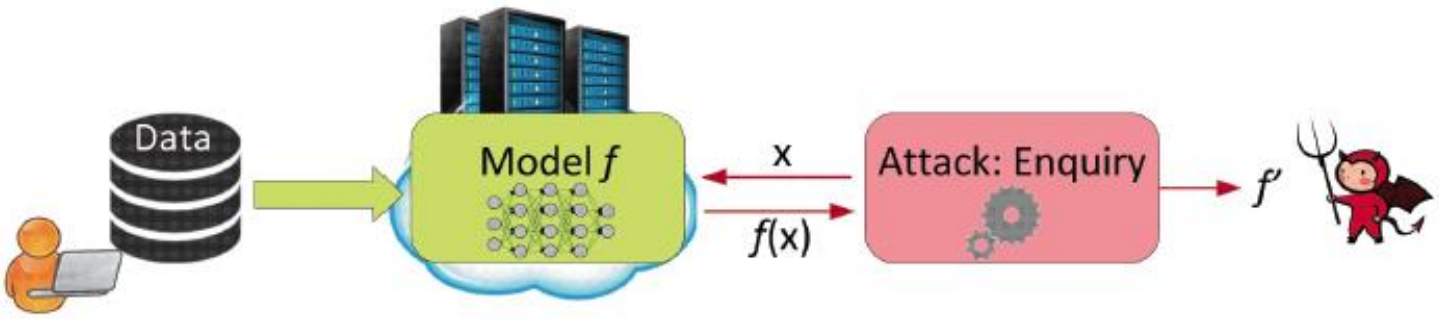


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Causes of Privacy Leaks

Causes of Privacy Leaks

- Overfitting is among the main causes of privacy leakage
  - It leads to poor generalization and memorization of the training data
  - Although adversarial training is often applied for increasing to model robustness, it reduces the accuracy on clean data, due to the trade-off between the model accuracy and robustness
    - The reduced accuracy can lead to increased sensitivity to data leakage
- Datasets that are more diverse and with larger number of class labels are more susceptible to attacks
  - I.e., binary classifiers are safer than multiclass models
  - Input samples that are out-of-distribution (i.e., are considered outliers with respect to the distribution of the training data) are more susceptible to privacy leakage
- Model complexity can also impact the vulnerability
  - Complex models with large number of parameters memorize more sensitive information about the training data

Membership Inference Attacks Against Machine Learning Models

Reza Shokri, Marco Strontia, Congzheng Song, Vitaly Shmtikov

Machine Learning Privacy

- **Machine Learning (ML) is behind many of the** technologies we use every day. It powers things like:.
- Online image and speech recognition
- Language translation tools
- Personalized ads and product recommendations

To make accurate predictions, ML models are trained on **real user data**, which often includes:

•Medical records, financial transactions, personal photos, voice commands, and location history

Privacy Issue?

- These models can unintentionally **leak information** about the data they were trained on
- This is especially dangerous when the model is available through a **black-box API**—the attacker doesn't need to know the internal workings to exploit it
- Such leaks can violate **user trust, ethical standards, and data protection laws**

Privacy Attacks In AML

- Privacy attacks — also called **inference attacks** or **confidentiality attacks** — aim to extract sensitive information from machine learning systems.

They typically target two key areas:

- These attacks attempt to reveal whether a specific person's data (such as medical records) was included in the model's training set. Example: Identifying patients whose data was used to train a disease prediction model.
- ML model- these attacks focus on uncovering the internal structure or parameters of the model itself.

Examples:

- Revealing the algorithm an insurance company uses to calculate premium rates
- Stealing the model a financial institution uses for credit card approval decisions

What is Membership Inference?

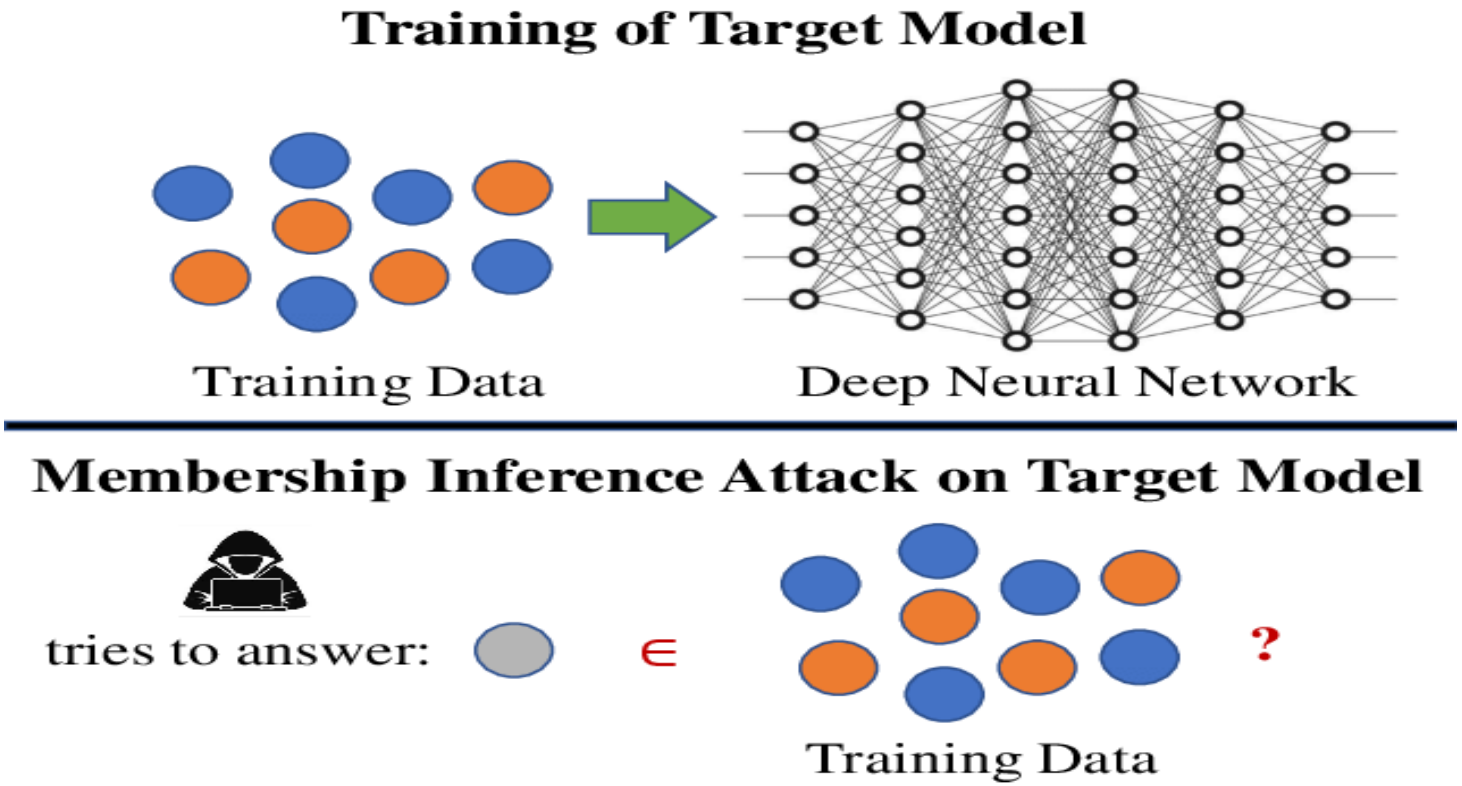
A type of privacy attack where the goal is to determine whether a specific data record was part of the training dataset used to train a machine learning model.

Example Scenario – Black-Box Attack:

•Attacker only has **black-box access**: can input data into the model and observe its predictions.

•By analyzing the model's output confidence or probability scores, the attacker tries to infer whether the input data was part of the training set.

What is Membership Inference?



Objective of the paper

Quantify Infomation Leakage
Measure how much private information about the training data can be inferred just from a model's outputs

Develop Attack Techniques
Propose a general and practical method for performing membership inference attacks using shadow models and attack models.

- **Evaluate in Real-World Settings**

Test these attacks on real services like:

- **Google Prediction API**
- **Amazon ML**

Threat Model

Adversary's Knowledge and Capabilities:

The attacker knows **the type of model** (e.g., neural network, decision tree) but **not its architecture or parameters**.

- Has **no access to the training data** used by the target model.
- May have access to a **similar dataset** from the same distribution (used to train shadow models).
- Can observe **prediction vectors** (confidence scores or probabilities) returned by the model.
- **Black-Box Access Assumption**:
- The model is accessed like an API: input goes in, predictions come out.
- No visibility into how predictions are made internally.
- This reflects **real-world ML-as-a-service** scenarios (e.g., Google Prediction API, Amazon ML).
- **Goal of the Adversary**:
- Given a data record and a prediction vector from the target model, determine whether that record was part of the training data (i.e., perform **membership inference**).

Membership Inference as a Classification Problem

•The attacker's goal is to find out if a specific data record was used to train a machine-learning model.

•To do this, they build a separate **attack model** that treats the problem as a **yes/no (binary) classification task**:

→ **Was this data point in the training set?**

→ **Yes (Member)** or **No (Non-member)**

•The attack model learns to detect patterns in the **prediction scores** (also called prediction vectors) returned by the target model.

•It finds that:

•**Training data (members)** usually receive **high confidence** predictions.

•**Unseen data (non-members)** tend to produce **less confident or uncertain** predictions.

- The attacker doesn't have the real training data, so they create and train **shadow models** using similar data they do have.
- Since they know which data went into their shadow models, they can label it as "member" or "non-member."
- This labeled data is used to **train the attack model**, so it learns how to spot the difference

Attack overview

Step 1: The attacker sends a data record to the target model (black-box access).
- **Step 2:** The target model returns a prediction vector (confidence scores for each class).
- **Step 3:** The attacker feeds this predictior vector (and optionally the true label) into the attack model.
- **Step 4:** The attack model analyzes the input and outputs either:

→ **"Member"** – if the record was likely par of the training data

→ **"Non-member"** – if the record was not used during training

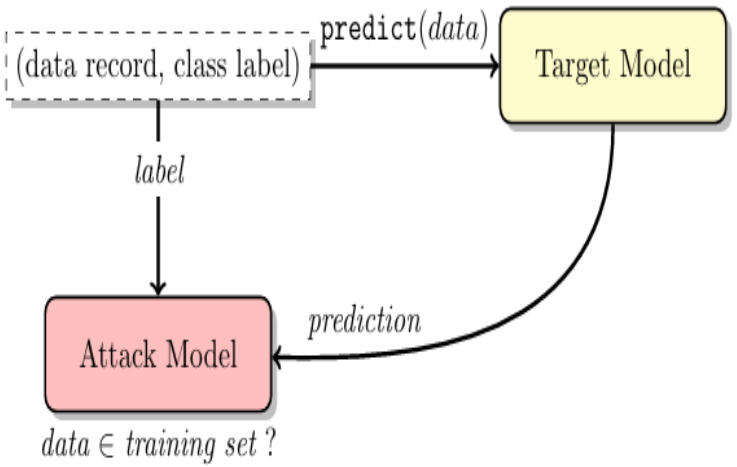
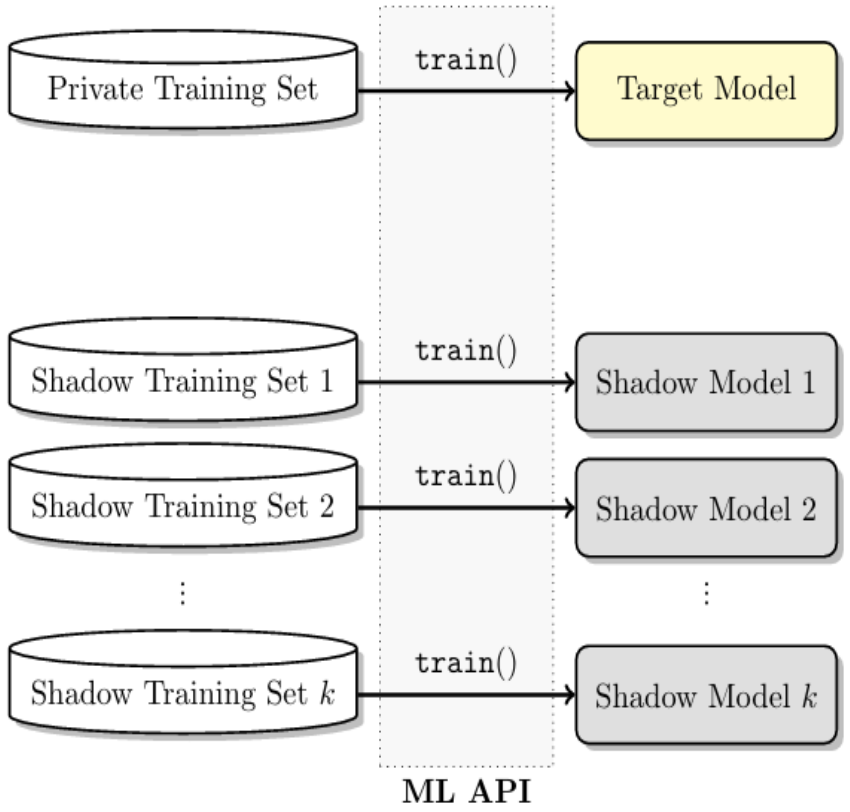


Fig. 1: Membership inference attack in the black-box setting. The attacker queries the target model with a data record and obtains the model's prediction on that record. The prediction is a vector of probabilities, one per class, that the record belongs to a certain class. This prediction vector, along with the label of the target record, is passed to the attack model, which infers whether the record was *in* or *out* of the target model's training dataset.

Shadow Training Attack

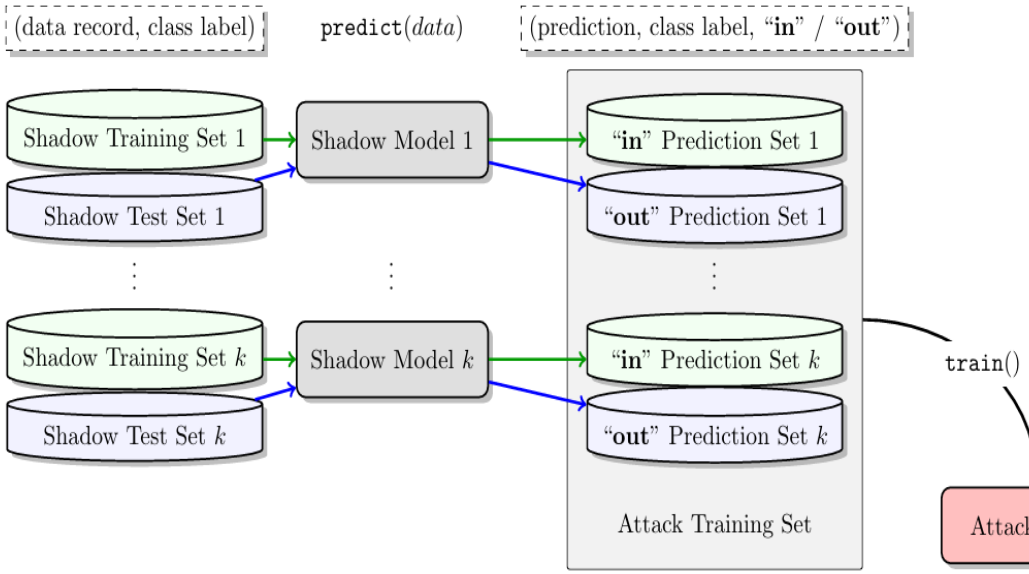
- The attacker does **not have access** to the target model's training data.
- To simulate the target model's behavior, the attacker trains **shadow models** on data from a **similar distribution**.
- These shadow models help the attacker understand how a model behaves on:
  → **Training data**
  → **Unseen data**
- Create several shadow models to substitute the target model
- Each shadow model is trained on a dataset that has a similar distribution as the private training dataset of the target model
- Same input instances are used to create shadow training sets for training multiple shadow models



Shadow Training attack

The output probability vectors from the shadow models are next used as inputs training attack models (as binary classifiers) for each class.

- An attack model is trained on these inputs to perform binary classification (in or out)
- A separate attack model is trained for each celebrity person in the shadow training sets



Shadow Training Attack

- The attack models for each class are afterward used to predict whether individual inputs instances were members of the private training set of the target model
- The assumption in this attack is that the output probability vectors of the shadow models are different for samples that are members of the shadow training sets, in comparison to samples that are members of the shadow test sets
- Experiments showed that increasing the number of shadow models improves the accuracy of membership inference, but it also increases the computational recourses

Real Data

- Uses datasets from the same domain as the target model.
- Example: If the target model is trained on shopping data, use publicly available purchase reco
- **Pros:** High realism
- **Cons:** May not always be available

2. Noisy Data

- Modify known real data by introducing random noise to simulate variation.
- Example: Flip 10–20% of feature bits randomly.
- **Pros:** Easy to generate, robust to imperfect knowledge
- **Cons:** Slight reduction in attack precision

3. Synthetic Data

Two techniques:

- **Model-based synthesis:**
  - Use the target model to find inputs it classifies with **high confidence**.
  - Employ a **hill-climbing search** to generate such inputs.
  - Produces data statistically similar to training set without prior knowledge.
- **Statistics-based synthesis:**
  - Sample each feature independently from known **marginal distributions**.
  - Requires some statistical knowledge of the dataset.

Synthetic Data Generation Method

When real or noisy data is unavailable, attackers can generate training data for shadow models using the **target model itself**. This approach is known as **model-based synthesis**.

- The attacker generates synthetic records that the target model classifies with **high confidence**.
- This is achieved using an **iterative hill-climbing algorithm** that tweaks input features to maximize the model's prediction confidence.
- .

Experimental Setup — Datasets Used

CIFAR-10	Image dataset with 60,000 32×32 color images across 10 object categories, Commonly used for object recognition tasks Evaluates membership inference in visual models with moderate class diversity
CIFAR-100	Similar to CIFAR-10 but with 100 fine-grained classes. Increases complexity and class overlap. Tests the effect of class count on attack performance
Purchases (Kaggle)	Tabular dataset with binary feature, 100-class classification task based on clustering. Models real-world e-commerce data commonly used in ML systems.
Texas Hospital Stay Dataset	Real medical discharge data from over 67,000 patients, Each record includes demographic, diagnostic, and procedural codes. Sensitive health domain used to evaluate privacy threats in healthcare.
Location Dataset (Foursquare)	Spatial-temporal check-in data of mobile users at different locations, Spatial-temporal check-in data of mobile users at different locations. Ideal for studying membership inference on behavioral datasets
MNIST	28×28 grayscale images of handwritten digits (0–9), Includes 60,000 training and 10,000 test samples, Serves as a simple visual baseline to examine vulnerability in low-complexity models
Adult Dataset	U.S. Census income prediction, Includes personal attributes like age, education, occupation, and marital status

Experimental Setup — ML Platforms

- **Google Prediction API**
  - Fully hosted, black-box machine learning service
  - No control over the internal model architecture or training parameters
  - Only allows input queries and returns prediction scores
- **Amazon Machine Learning (Amazon ML)**
  - Black-box API with limited control over model configuration
  - User can adjust a few parameters (e.g., passes over data, regularization)
  - Used to test inference attack success under varying configurations
- **Local Neural Networks (Torch7)**
  - Fully configurable neural network models trained locally
  - Complete control over architecture, training method, and data
  - Provided a baseline to validate the attack model's behavior and effectiveness before applying it to MLaaS platforms.

Results — Attack Precision on CIFAR-10 and CIFAR-100

- **Attack precision improves as the training set size decreases**
  - Smaller training sets → more overfitting → higher risk of leakage
- The target model becomes more confident on its training data when overfitted
- **Recall** of the attack remains close to **1.0**, regardless of training set size

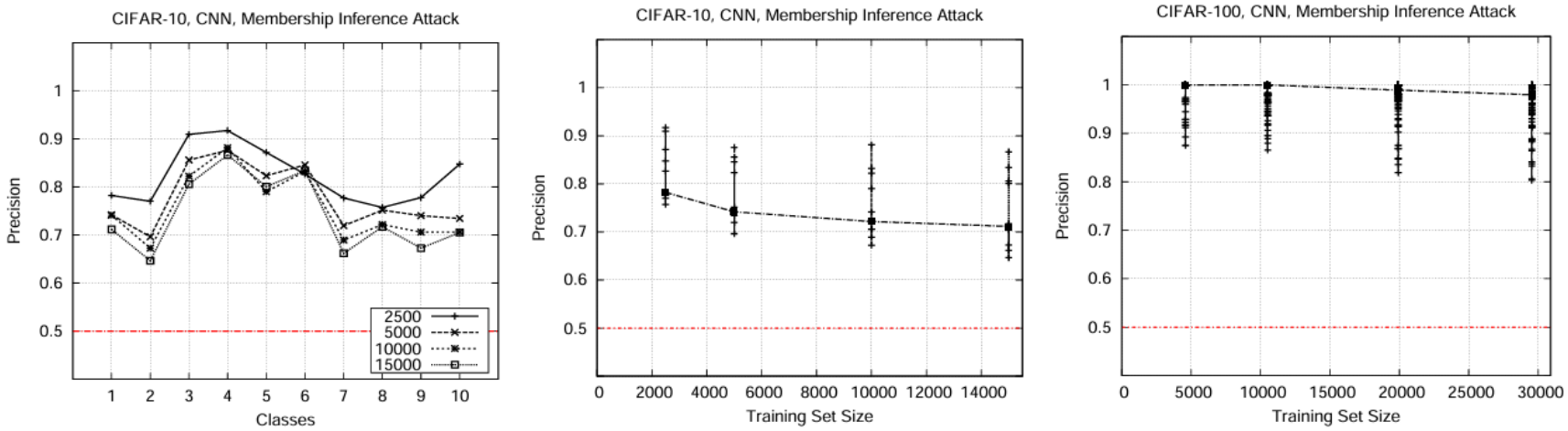


Fig. 4: Precision of the membership inference attack against neural networks trained on CIFAR datasets. The graphs show precision for different classes while varying the size of the training datasets. The median values are connected across different training set sizes. The median precision (from the 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.

Purchase Dataset — Google vs. Amazon Platforms

- Binary vectors representing user purchase history. Used to evaluate inference attacks on **Google Prediction API** and **Amazon ML**
- Precision Breakdown by Class**- CDF plot** shows per-class attack precision across Google and Amazon setups. Google yields **consistently higher per-class precision.** Amazon precision improves with **increased training and regularization.**
- **Both platforms are vulnerable**, even in black-box settings
- **Google** shows greater leakage under default config
- **Per-class precision** supports overall attack reliability

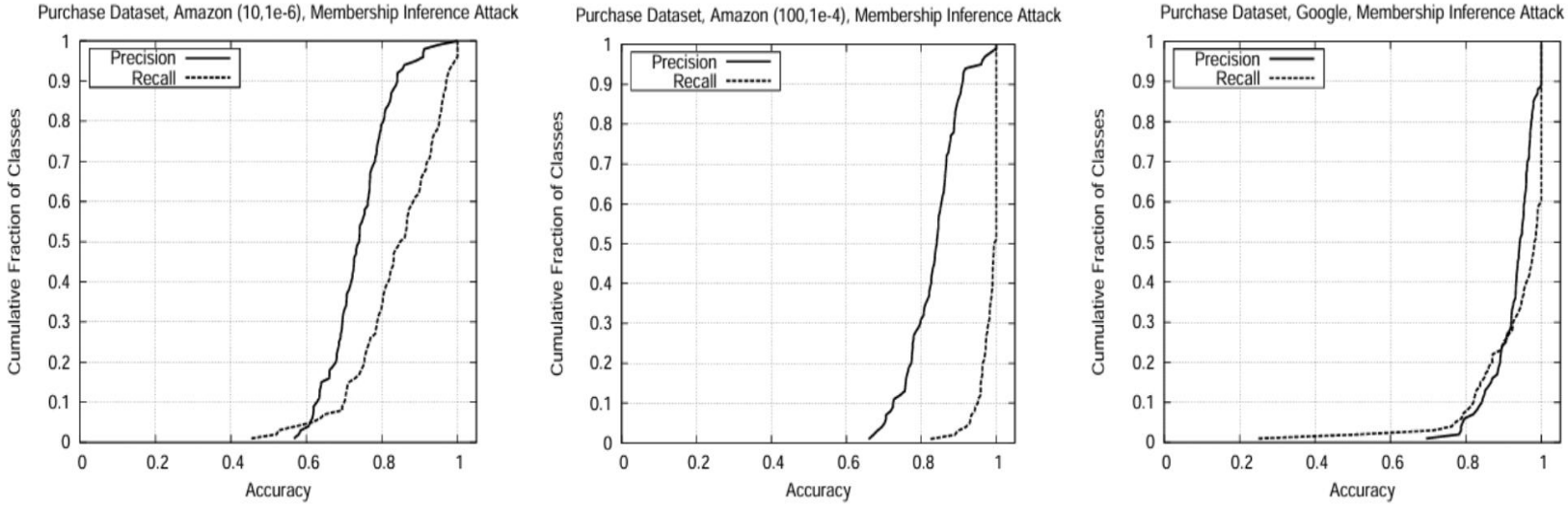


Fig. 5: Empirical CDF of the precision and recall of the membership inference attack against different classes of the models trained using Amazon ML (in two different configurations) and Google Prediction API on $10,000$ purchase records. $50, 75, 90$-percentile of precision is $0.74, 0.79, 0.84$ on Amazon $(10, 1e-6)$, $0.84, 0.88, 0.91$ on Amazon $(100, 1e-4)$, and $0.94, 0.97, 1$ on Google, respectively. Recall is close to 1.

Results — Texas Hospital Stay Dataset

Real-world dataset with **67,000+ patient records.** Represents a **highly sensitive domain**: healthcare

- Platform: **Google Prediction API**
- **Attack precision:**
- **Above 0.6** for the majority of classes
- **Above 0.85** for over **20 different classes**
- Precision varies by class but remains significantly high

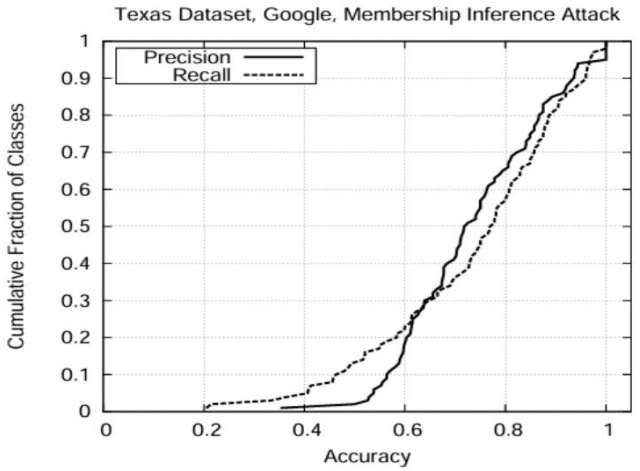


Fig. 6: Precision and recall of the membership inference attack against the classification model trained using Google Prediction API on the Texas hospital-stay dataset.

Location Dataset & Shadow Data Comparison

Dataset: Foursquare Check-in Records (Location history)
Shadow Training Types Tested:
- **Real Data**
- **Noisy Data** (10% and 20%)
- **Synthetic Data** (Model-based and Marginal Distribution)

Shadow Data Type	Precision	Recall
Real Data	0.678	~1.0
10% Noisy	0.666	~1.0
20% Noisy	0.613	~1.0
Synthetic (Marginal)	0.795	0.991
Synthetic (Model-based)	0.896	~1.0

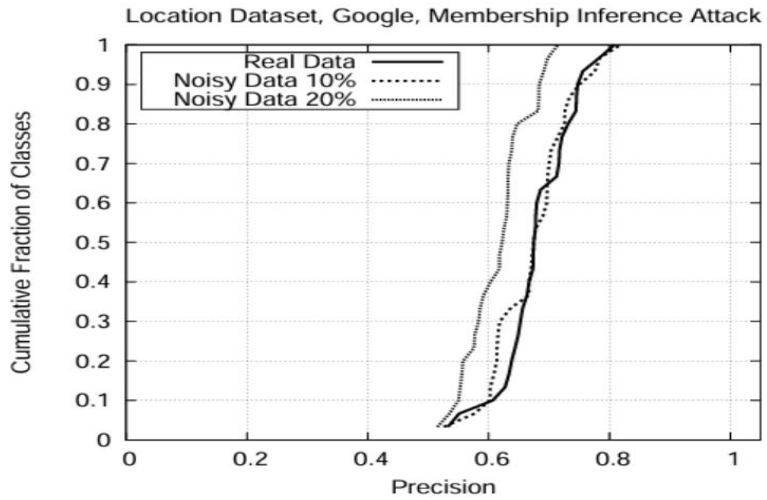


Fig. 8: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the location dataset. Results are shown for the shadow models trained on real data and for the shadow models trained on noisy data with 10% and 20% noise (i.e., $x\%$ of features are replaced with random values). Precision of the attack over all classes is 0.678 (real data), 0.666 (data with 10% noise), and 0.613 (data with 20% noise). The corresponding recall of the attack is 0.98, 0.99, and 1.00, respectively.

Synthetic Data Impact — Marginal vs. Model-Based Generation

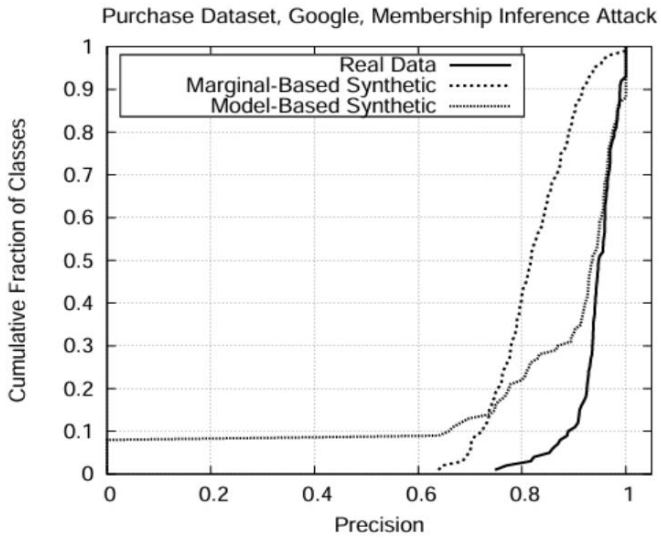


Fig. 9: Empirical CDF of the precision of the membership inference attack against the Google-trained model for the purchase dataset. Results are shown for different ways of generating training data for the shadow models (real, synthetic generated from the target model, synthetic generated from marginal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

Impact of Number of Classes & Training Data Per Class

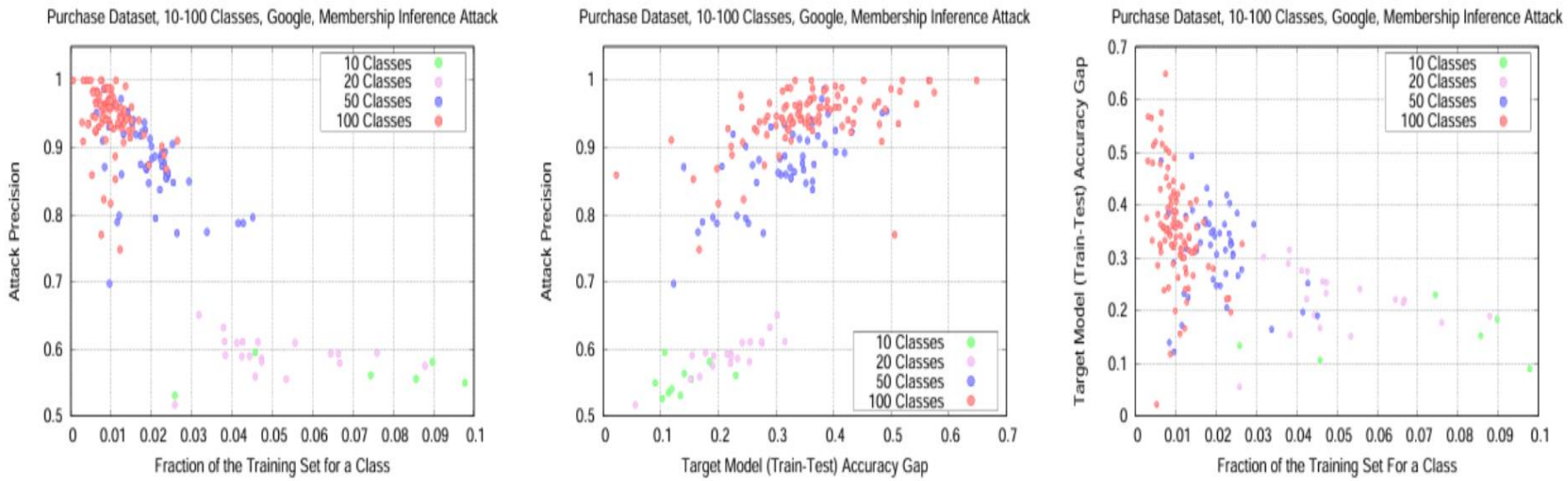


Fig. 11: Relationship between the precision of the membership inference attack on a class and the (train-test) accuracy gap of the target model, as well as the fraction of the training dataset that belongs to this class. Each point represent the values for one class. The (train-test) accuracy gap is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

- **Overfitting** occurs when a model performs much better on training data than on unseen data.
- This results in the model **memorizing specific training instances**, which leads to higher vulnerability to membership inference.

Why Attacks Work — Prediction Accuracy & Uncertainty

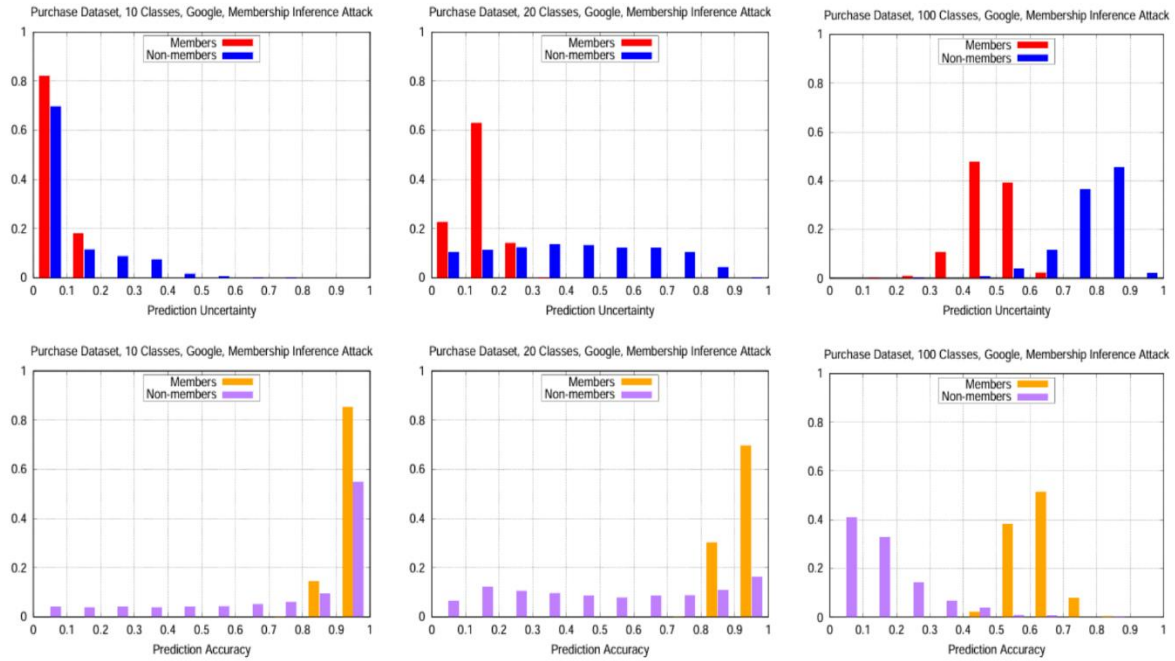


Fig. 12: Classification uncertainty (top row) and prediction accuracy (bottom row) of the target model for the members of its training dataset vs. non-members, visualized for several sample classes. The difference between the member and non-member output distributions is among the factors that our attack exploits to infer membership. The accuracy of our attack is higher for the models where the two distributions are more distinguishable (See Table II).

Dataset	Training Accuracy	Test Accuracy	Attack Precision
MNIST	0.984	0.928	0.517
Adult	0.848	0.842	0.503
Location	1.000	0.673	0.678
Purchase (100)	0.999	0.659	0.935
TX Hospital	0.668	0.517	0.657

Mitigation Techniques

To **reduce information leakage** from ML models without significantly degrading their predictive performance.

- **Restricting Prediction Vectors**
  I. Limit model outputs to top-*k* classes (e.g., top-1, top-3)
  II. Removes less confident probabilities to reduce leakage
- **Rounding Prediction Scores**
  I. Round confidence scores to *d* decimal places
  II. Lower precision = less information for attacker
- **Temperature Scaling**
  I. Increases softmax temperature to raise prediction entropy
  II. More uniform outputs = harder to differentiate members
- **Regularization**
  I. L2 regularization to reduce overfitting
  II. Improves generalization and reduces attack success

Mitigation Techniques

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.

Differential Privacy as a Defense Overview of Differential Privacy

What Is Differential Privacy?

•Differential Privacy (DP) is a **mathematical framework** that adds **random noise** to the training process of ML models.

•It ensures that the model's output **does not reveal whether any single data point** (like a patient's health record) was included in the training data.

•The presence or absence of any individual record **barely changes** the outcome.

How Differential Privacy Blocks Membership Inference

•**Membership inference attacks** work by spotting small differences in how a model responds to:

- **Member data** (used during training)
- **Non-member data** (never seen before)

•**Differential Privacy (DP)** disrupts this by:

- Adding controlled **random noise** during training
- Making the model's responses **statistically similar** for all data — whether it was used or not

Recommendations for ML-as-a-Service Providers

1. **Be Transparent About Privacy Risks**
   i. Clearly communicate to users that **models can leak training data**, even when accessed via black-box APIs.
   ii. Share known vulnerabilities (e.g., overfitting, confidence score exposure).
   iii. Offer documentation on **attack surfaces** and privacy implications.

2. **Provide Built-in Privacy Controls**
   i. Include settings that **limit prediction exposure**:
   i. Top-*k* class outputs, Confidence rounding, Temperature scaling
   ii. Offer **regularization defaults** and alerts for overfitted models.

3. **Enable Differential Privacy Options**
   i. Support **training with differential privacy** for sensitive domains (e.g., healthcare, finance).
   ii. Allow developers balance privacy and accuracy using tunable parameters (like ε).

4. **Educate and Empower Users**
   i. Provide **tools to audit models** for overfitting and data leakage.
   ii. Offer tutorials and dashboards that explain **privacy trade-offs**.

Conclusion

- Membership inference attacks can expose private training data, even in black-box models.
  Overfitting and detailed prediction outputs increase vulnerability.
  Simple defenses help, but trade-offs with accuracy remain.
  Differential Privacy offers the strongest protection.
  Privacy must be considered during model design.

- Explore stronger, adaptive defenses and hybrid mitigation techniques.
  Expand beyond membership inference to other attack types.
  Improve platform-level privacy tools for developers.
  Encourage transparent ML practices in sensitive domains.
  Balance model performance with ethical data protection.

Attacks against Distributed Learning

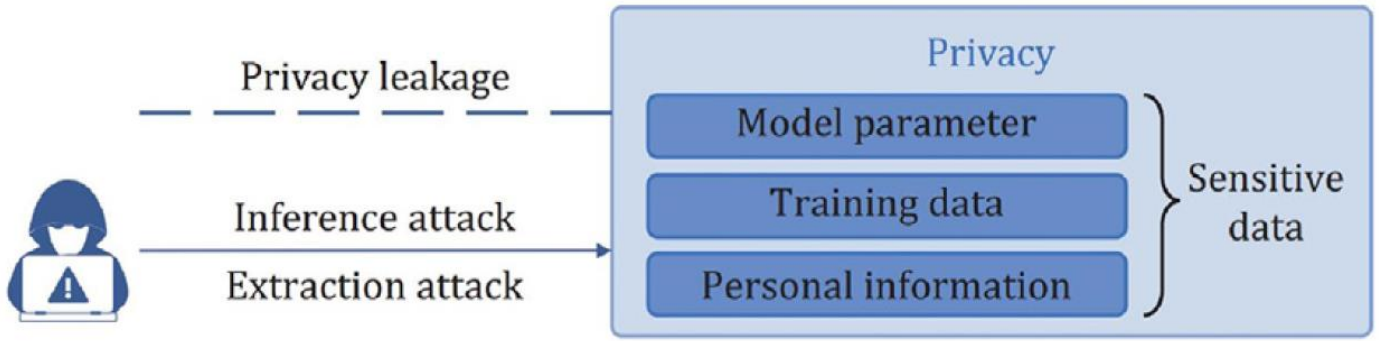
Attacks against Distributed Learning

- ***Privacy attacks against federated learning*** and related distributed learning models have also been demonstrated
- The attacks can be *passive* (the adversary collects the updates) and *active* (the adversary shares information to impact the training procedure)
  - A malicious attacker who participates in federated learning can perform membership inference attack to reveal if other participants used a data record for training
    - Nasr (2018) Machine Learning with Membership Privacy Using Adversarial Regularization
  - Property inference attacks were developed to reveal whether training data with certain properties were used by the other participants
    - Melis (2019) Exploiting Unintended Feature Leakage in Collaborative Learning
  - Training data reconstruction attack was accomplished by using an additional GAN model for reconstructing class representative samples from the local dataset used by the other participants
    - Hitaj (2017) Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning

Privacy Attacks against LLMs

Privacy Attacks against LLMs

- LLMs face privacy issues that can undermine trust and reliability of their applications
  - ***Privacy leakage*** occurs when sensitive or personal information is inadvertently disclosed through the LLM's responses, due to the model's training on data containing such information
  - ***Inference attacks*** involve the exploitation of model responses to infer sensitive information about the training data or the underlying algorithms, potentially compromising LLM's privacy or security
  - ***Extraction attacks*** are attempts to reverse-engineer or obtain proprietary information, such as model parameters, by interacting with the LLM



Zhang et al. (2025) – On Large Language Models Safety, Security, and Privacy: A Survey

Privacy Leakage

Privacy Attacks against LLMs

- ***Privacy leakage*** is an important concern due to the inherent potential of LLMs to inadvertently disclose sensitive information
  - For example, Samsung Electronics had sensitive corporate information unintentionally disclosed through interactions with ChatGPT
  - Also, Gemini Flash has been reported to collect excessive user data beyond what is necessary and has anonymization failures when sharing data with third parties
- Privacy leakage can arise from three primary causes
  - Private information in training data
    - The training datasets may contain sensitive or personally identifiable information where despite efforts at anonymization, there remains a risk of reidentification
  - Data memorization
    - LLMs can unintentionally retain and later reproduce specific pieces of training data, including sensitive personal information
  - Inference leakage
    - Even without explicit exposure to private data during interactions, LLMs can infer sensitive details through contextual clues
    - User prompts can trigger responses that reveal more than intended, especially when the model has learned correlations between different types of data during training

Inference Attack

Privacy Attacks against LLMs

- ***Inference attacks*** leverage the models' ability to infer sensitive information from their training datasets
  - Membership inference attack (MIA) – finds whether a particular data point was included in the model's training set
  - Shi (2023) Detecting pretraining data from large language models
    - Introduced a MIA detection technique called MIN-K% PROB
    - This method is based on the assumption that an unseen example is more likely to contain outlier words with low probabilities under LLM, whereas a seen example would exhibit fewer such low-probability terms
  - Kaneko (2024) Sampling-based pseudo-likelihood for membership inference attacks
    - SaMIA (Sampling-based pseudo-likelihood MIA) attack uses ROUGE-N score to quantify the n-gram match between the target text and multiple LLM outputs, thus inferring the text's membership in the training data

Inference Attack

Privacy Attacks against LLMs

- Attribute inference attack
  - Infer specific attributes or characteristics of the data contained within the training set of LLMs
    - E.g., reveal general information about the dataset, such as users demographic information
  - Staab (2023) Beyond Memorization: Violating Privacy via Inference with Large Language Models
    - Assessed the capabilities of nine LLMs to infer eight distinct personal attributes, including: location, income, age, sex, education, occupation, place of birth, relationship state
    - Constructed a dataset consisting of real Reddit profiles, and achieved up to 85% top-1 and 95% top-3 accuracy
    - The authors show that common mitigations, such as text anonymization and model alignment, are ineffective at protecting user privacy against LLM inference
- Data reconstruction attack
  - A more intrusive form of data compromise, aiming to reconstruct entire sequences or datasets from the model's knowledge base
  - Carlini (2020) Extracting training data from large language models
    - Explained later in this lecture

Model Extraction Attack

Privacy Attacks against LLMs

- ***Model extraction attacks*** are designed to retrieve information about the LLM, such the model parameters, gradients, architecture, etc.
  - This is typically achieved through an analysis of the model's outputs or observed behaviors
  - Attackers often employ methods such as optimization algorithms to fine-tune substitute models that emulate the behavior of the target model
  - Carlini (2024) Stealing Part of a Production Language Model
    - Explained later in the lecture

Training Data Extraction from GPT-2

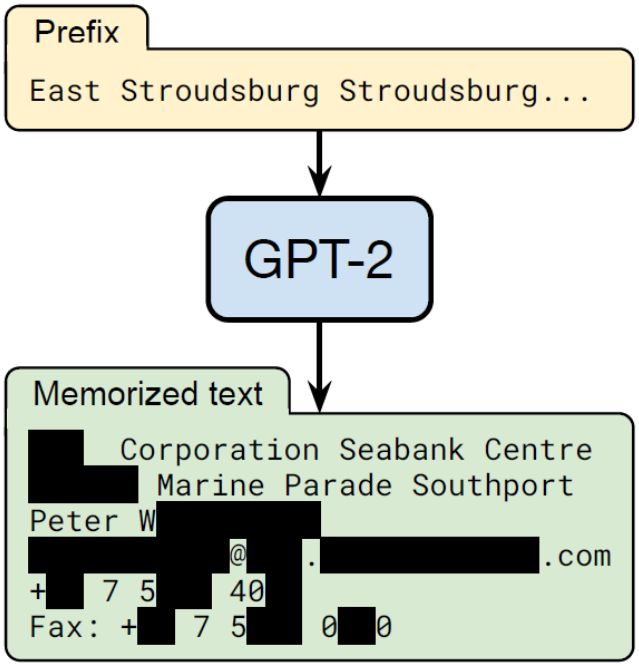
Training Data Extraction Attack against GPT-2

- ***Training data extraction attack***
  - Carlini (2020) Extracting training data from large language models
- Attack on ***GPT-2*** language model (LM)
  - GPT-2 has 1.5 billion parameters, it is trained on public data collected from the Internet
- The goal of the attack is to analyze output text sequences from GPT-2 and identify text that has been memorized by the model
  - The authors had black-box query access to the GPT-2 model
- Successfully extracted data examples include:
  - Personally identifiable information (PII): names, phone numbers, e-mail addresses
  - News headlines, log files, Internet forum conversations, code
- The extracted information was present in just one document in the training data

Training Data Extraction from GPT-2

Training Data Extraction Attack against GPT-2

- Example of training data extraction
  - The authors query GPT-2 by entering the ***prefix***: "East Stroudsburg Stroudsburg..."
  - The model outputted a block of text, which included the full name, phone number, email address, and physical address of the person
  - This information was included in the training data for GPT-2, it was memorized by the model, and extracted by using the training data extraction attack



Attack Threat Model

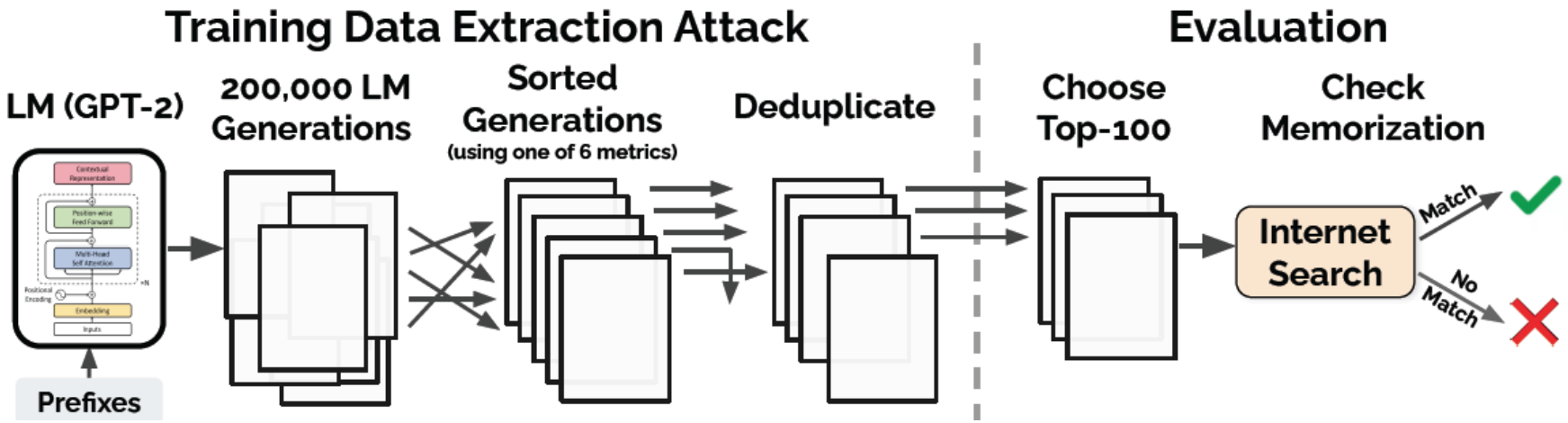
Training Data Extraction Attack against GPT-2

- The authors had black-box query access to GPT-2
- Objective: extract memorized training data
  - The strength of the attack is measured based on the number of documents in which the text appeared
    - Memorizing one word that occurred in many training examples (documents) is not severe
  - Stronger attack extract text that occurred in one single document
    - This is referred to as "unintended" memorization
- The training data for GPT-2 was collected by OpenAI from public sources
  - OpenAI didn't release the training dataset
    - But they released a document on the data collection process
  - The authors downloaded the public data by following the documentation
    - They didn't have access to the actual dataset used by OpenAI

Attack Approach

Training Data Extraction Attack against GPT-2

- Attack approach
  - Generate many text samples by using prefix prompts to GPT-2
    - Build 3 datasets of 200,000 generated samples, each sample if 256 tokens long
  - Sort the generated output text using 6 metrics (see next page)
    - LM has high confidence when the text is taken directly from the training data
  - Remove duplicate outputs
  - For the top 100 outputs, perform an Internet search to confirm whether the generated text is an exact match to a web document
  - Check with OpenAI to confirm if the extracted text occurred in their training dataset



Attack Approach

Training Data Extraction Attack against GPT-2

- ***Metrics for sorting*** the predicted text
  1. Perplexity, $P = \exp(-(1/n)\sum_{i=1}^{n} \log f_\theta(x_i|x_1, x_2, \ldots, x_{i-1}))$
  2. Comparison to the predictions by GPT-Small and GPT-Medium models
     - It is less likely that the different models will memorize the same data
  3. Text entropy when the output is compressed using zlib compression
  4. Perplexity when the text is switched from uppercase to lowercase letters
  5. Averaged perplexity using a sliding window of 50 tokens
- Other strategies to improve the attack:
  - Use prompts based on Internet text
- The authors used 3 datasets of 200,000 generated samples
  - For the 6 metrics above, this resulted in 3×6 configurations, or 1,800 top samples

Results

Training Data Extraction Attack against GPT-2

- The authors manually inspected 1,800 generated samples from GPT-2
- They identified 604 memorized training examples (about 33% of the samples)
  - The categories of memorized training examples are shown in the table

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Training Data Extraction from GPT-2

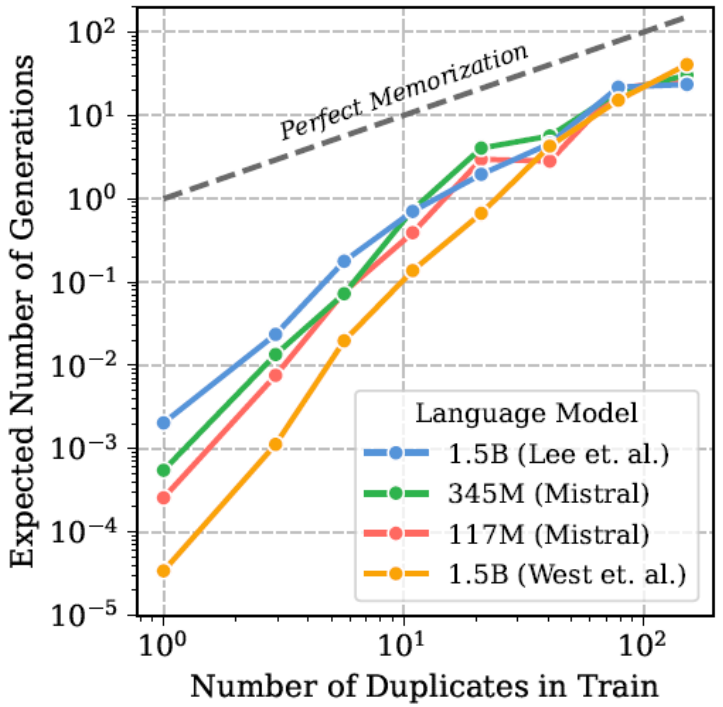
Training Data Extraction Attack against GPT-2

- The main findings of the study are:
  - Most of the memorized samples were found in only 1 document in the dataset
    - However, the samples were repeated multiple times in the document
  - Out of 1,800 candidate sequences that were manually analyzed by the authors, GPT-2 memorized 600 from the public training data
  - Larger language models are more vulnerable to data extraction than smaller models
  - Although LMs are trained on large datasets and therefore they exhibit little overfitting, they can still memorize the training data
- Implications:
  - Training data extraction attacks have previously been limited to small LMs trained on small datasets
  - It was also believed that LMs do not memorize the data, because they exhibit little overfitting
  - Recent LMs are increasingly larger, thus, such vulnerabilities can become more significant

Data Deduplication as Mitigation Strategy

Data Deduplication as Mitigation Strategy

- ***Defense against training data extraction using data deduplication***
  - Kandpal (2022) Deduplicating Training Data Mitigates Privacy Risks in Language Models
- This work showed the success of training data extraction attack against GPT-2 was due to the duplication of data in the training set
- Deduplicating data samples can significantly reduce the attack success
- The figure shows the expected number a text sequence can occur in generated text versus the number of occurrence of that text sequence in the training dataset
  - There is a linear relationship between these quantities
  - Perfect Memorization refers to generating a sequence at the same frequency as it appears in the training data



Stealing Part of LLM Attack

Stealing Part of LLM Attack

- ***Model extraction attack***
  - Carlini (2024) Stealing Part of a Production Language Model
- Model-stealing attack extracts information from black-box production LLMs, such as Open AI's ChatGPT or Google's PaLM-2
  - The attack recovers the embedding projection layer (i.e., the last layer) of a transformer model
  - Black-box attack, the authors have typical API access to the LLMs
  - For under $20 USD, the attack extracts the entire projection matrix of the OpenAI's GPT-3 LLMs ada and babbage

Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Sizes of layers in transformer networks

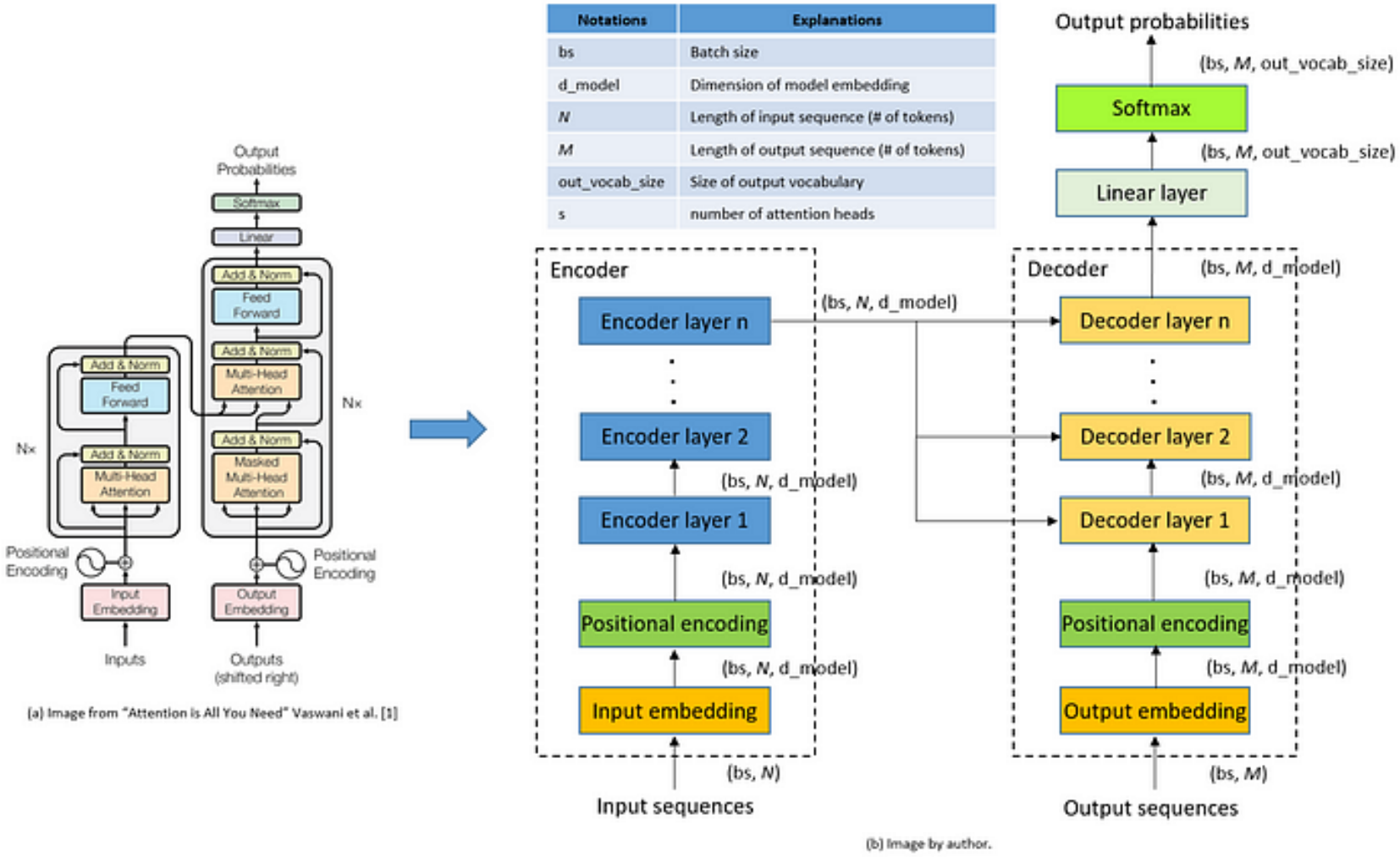
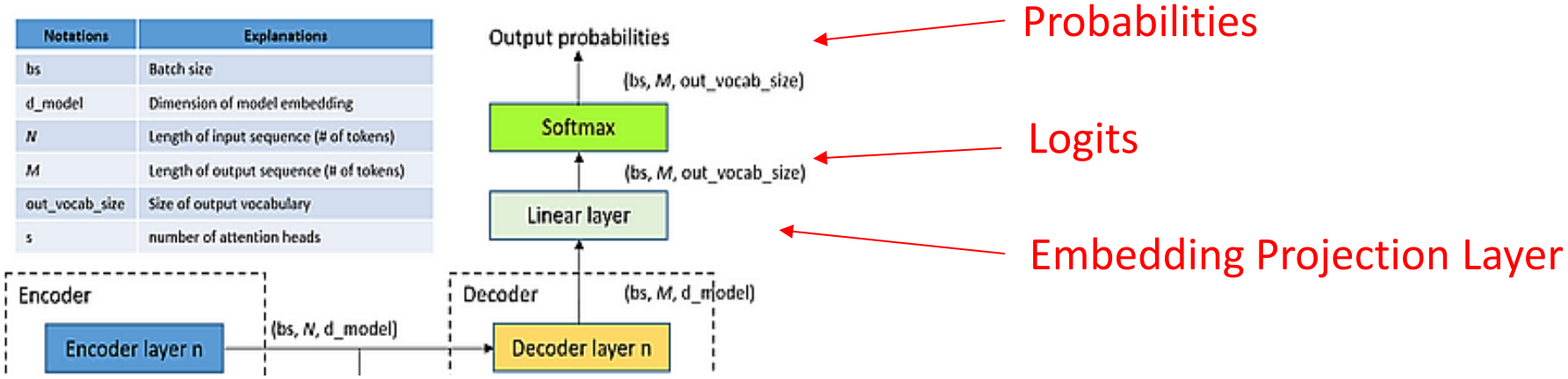


Figure from: https://medium.com/@freshtechyy/transformer-a-detailed-explanation-from-perspectives-of-tensor-shapes-and-pytorch-implementation-74fdff967de6

Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Embedding projection layer
  - For each token, the layer takes a vector of size d_model and outputs a logits vector of size out_vocab_size
    - d_model is the size of vector embeddings (a.k.a. hidden dimension), i.e., it is the dimensionality of all internal vectors processed throughout the model (see previous page)
    - Out_vocab_size is the size of the vocabulary (typical LLM has about 50,000 words vocabulary)
  - During training the model predicts the next token at every position
    - Logits of tokens at position *t* are used to predict the token at position *t*+1
      - E.g., the logits for the tokens "Mary had a little" are used to predict the logit vector for the next token
      - The logit vector has 50,000 values, with the highest value having the word "lamb"



Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Attack assumption: querying the LLM via API returns the logits values for each token (having the size out_vocab_size)
  - The embedding dimension d_model is not known for proprietary LLMs
  - Note that d_model is much smaller than the vocabulary size
- Recovering the size of the embedding dimension
  - Intuition: although the logits have out_vocab_size, they lie in a smaller dimensional space with size d_model
  - By querying the LLM many times with random prompts and applying Singular Value Decomposition, the dimensionality of the reduced space can be estimated
  - Algorithm:

Algorithm 1 Hidden-Dimension Extraction Attack

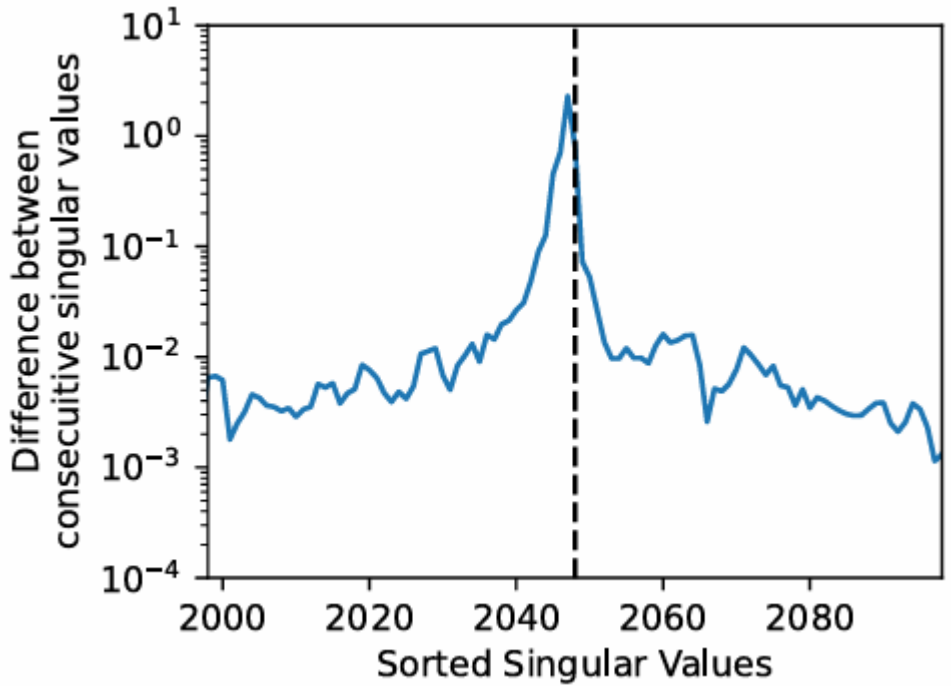
Require: Oracle LLM $\mathcal{O}$ returning **logits**

Initialize n to an appropriate value greater than h
Initialize an empty matrix $\mathbf{Q} = \mathbf{0}^{n \times l}$
for $i = 1$ to n **do**
    $p_i \leftarrow$ `RandPrefix()` ▷ Choose a random prompt
    $\mathbf{Q}_i \leftarrow \mathcal{O}(p_i)$
end for
$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \leftarrow \text{SingularValues}(\mathbf{Q})$
count $\leftarrow \arg\max_i \log\|\lambda_i\| - \log\|\lambda_{i+1}\|$
return count

Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Recovering the size of the embedding dimension
  - The figure shows the sorted singular values for the Pythia 1.4B model
    - The largest SV is at 2,048, which is the embedding dimension for this model
  - The table shows the recovered embedding dimensions for several LLMs



Model	Hidden Dim
GPT-2 Small (fp32)	768
GPT-2 XL (fp32)	1600
Pythia-1.4 (fp16)	2048
Pythia-6.9 (fp16)	4096
LLaMA 7B (fp16)	4096
LLaMA 65B (fp16)	8192

Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Recovering the weights of the embedding projection layer
  - The authors used the SVD decomposition to further recover the entire projection matrix of the last layer
- The table below shows the attack success rate and the cost for querying LLMs from OpenAI for dimension extraction and weight matric extraction

	Dimension Extraction			Weight Matrix Extraction		
Model	Size	# Queries	Cost (USD)	RMS	# Queries	Cost (USD)
OpenAI ada	1024 ✓	$< 2 \cdot 10^6$	$1	$5 \cdot 10^{-4}$	$< 2 \cdot 10^7$	$4
OpenAI babbage	2048 ✓	$< 4 \cdot 10^6$	$2	$7 \cdot 10^{-4}$	$< 4 \cdot 10^7$	$12
OpenAI babbage-002	1536 ✓	$< 4 \cdot 10^6$	$2	†	$< 4 \cdot 10^6$ †+	$12
OpenAI gpt-3.5-turbo-instruct	* ✓	$< 4 \cdot 10^7$	$200	†	$< 4 \cdot 10^8$ †+	$2,000†+
OpenAI gpt-3.5-turbo-1106	* ✓	$< 4 \cdot 10^7$	$800	†	$< 4 \cdot 10^8$ †+	$8,000†+

✓ Extracted attack size was exactly correct; confirmed in discussion with OpenAI.
* As part of our responsible disclosure, OpenAI has asked that we do not publish this number.
† Attack not implemented to preserve security of the weights.
+ Estimated cost of attack given the size of the model and estimated scaling ratio.

Stealing Part of LLM Attack

Stealing Part of LLM Attack

- Usefulness of this attack
  - It reveals the width of the LLM, which is often correlated with the number of total parameters
  - It reduces the degree of black-box knowledge about the model
  - Raises concerns that advanced attacks may recover more information about the model
- Responsible disclosure
  - The authors shared the attack with developers of LLMs
  - In response, OpenAI and Google modified their APIs to introduce mitigations and defenses against the attack

Mitigation Strategies for LLMs

Privacy Attacks against LLMs

- Mitigation strategies for privacy attacks on LLMs include
  - Selecting the training data for LLMs
    - Apply methods that limit the amount of sensitive content (e.g., filter personal information)
    - De-duplicate content in the training data
  - Training with differential privacy
    - DP can reduce, but cannot prevent, memorization of content in the dataset
  - Federated learning
    - Enables models to be trained on decentralized data, reducing the need to centralize and expose sensitive user data
  - Cryptography methods
    - Encryption techniques, secure communication protocols to prevent exposing model parameters or user's inputs
  - Watermarking
    - Embeds unique identifiers into data, serving as a traceable source against unauthorized use or access (e.g., by integrating special tokens for sensitive information)
  - Auditing LLMs
    - Apply audits to determine the level of memorization in LLMs
- Limitations: reduced accuracy, longer training times

References

1. Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook (link)
2. Rigaki and Carcia (2021) A Survey of Privacy Attacks in Machine Learning (link)
3. Cristofaro (2020) An Overview of Privacy in Machine Learning (link)
4. Zhang (2025) On Large Language Models Safety, Security, and Privacy: A Survey (link)