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
  - Attacks against distributed learning
- Carlini (2020) – Training data extraction attack against GPT-2
- Yu (2023) – Training data extraction attack against GPT-Neo

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)

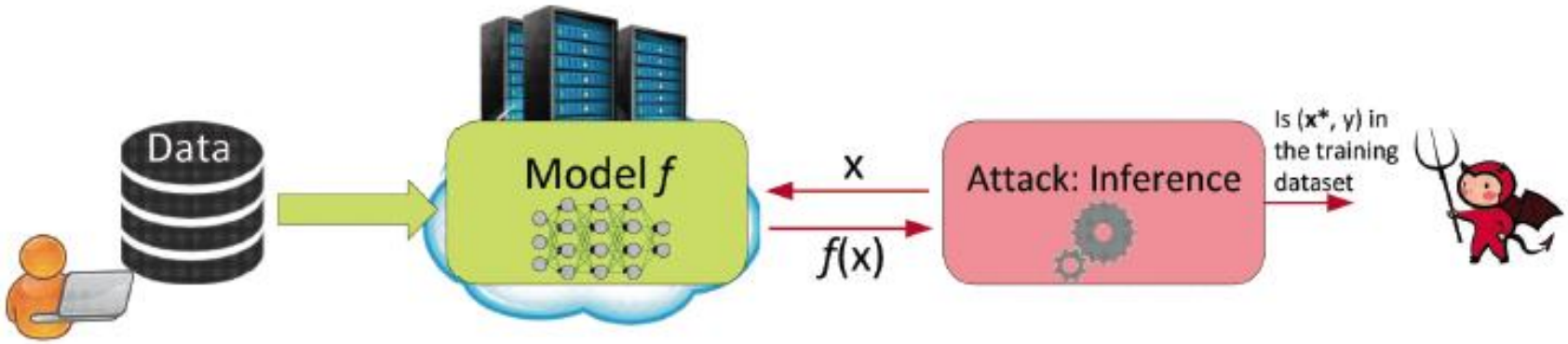
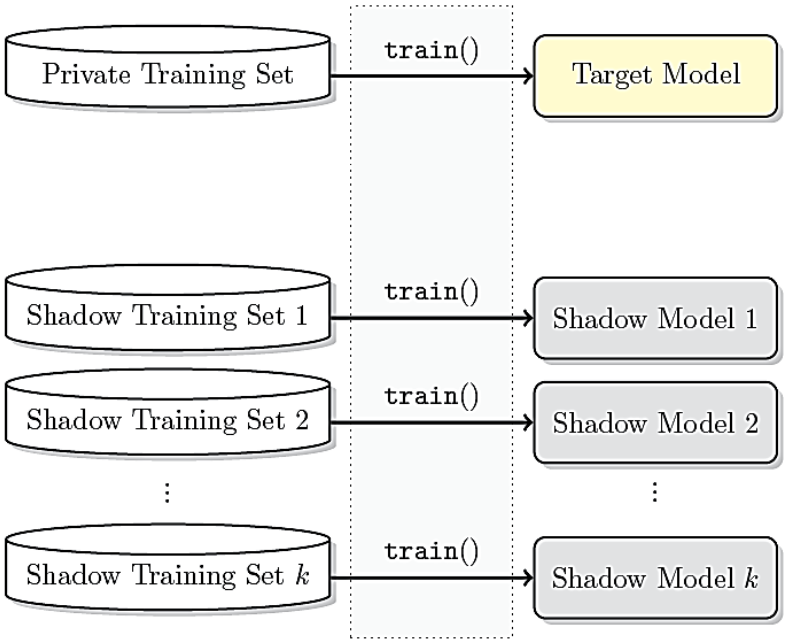


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

Shadow Training Attack

Membership Inference Attack

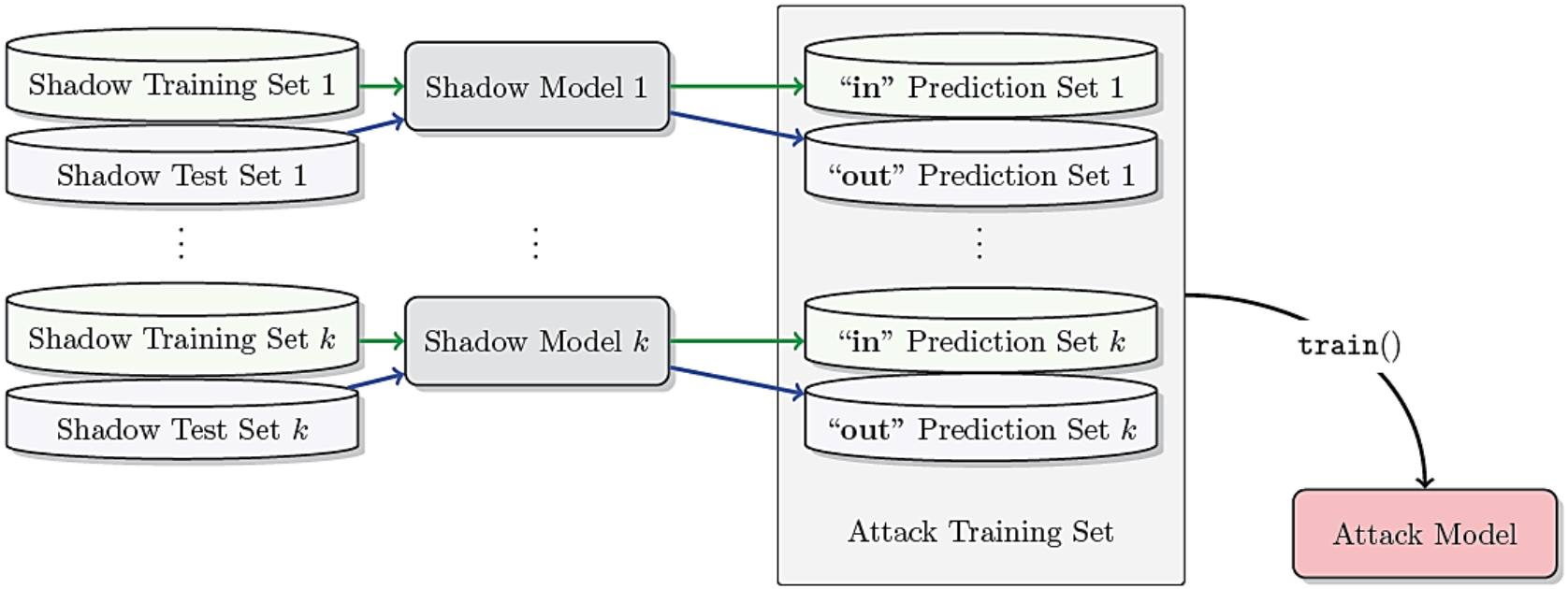
- Shokri (2016) Membership Inference Attacks Against Machine Learning Models
- Threat model:
  - The adversary has back-box query access to the target model
  - The goal is to infer whether input samples were part of its private training set
- ***Shadow training*** approach:
  - Create several shadow 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
    - E.g., if the target model performs celebrity face recognition, the attacker can collect images of celebrities from the Internet
      - Then, query the target model with images of Brad Pitt, and if the confidence of the target model is high, then probably the private training set contains images of Brad Pitt: use those images for the shadow training sets
    - Same input instances are used to create shadow training sets for training multiple shadow models



Shadow Training Attack

Membership Inference Attack

- The output probability vectors from the shadow models are next used as inputs for training attack models (as binary classifiers) for each class
  - E.g., the probability vectors for all input images of Brad Pitt from all shadow training sets are labeled with 1 (meaning 'in' the training set)
  - The probability vectors for all input images of Brad Pitt from all shadow test sets are labeled with 0 (meaning 'out' or not in the training set)
  - 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

Membership Inference 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

Shadow Training Attack

Membership Inference Attack

- The table shows the accuracy of a target model on training and testing sets, and the success of the attack for several models
  - One can note that the larger the overfitting (difference between the training and testing accuracy), the more successful the membership inference attack is
    - Conclusively, overfitting not only reduces the generalization of a model, but also makes the model more likely to leak sensitive information about the training data
  - In addition, the attack was more successful for training datasets that are more diverse and have larger number of classes (e.g., compare Purchase model with 100 classes to Purchase with 2 classes)

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

Shadow Training Attack

Membership Inference Attack

- If the adversary cannot get access to input samples for creating shadow training sets or to any other statistics about the target data distribution, the authors developed an algorithm for creating synthetic samples by querying the target model
  - First, for each class, start with a random sample, query the target model and change each input feature until the modified samples are classified with high confidence
  - Next, randomly change a set of input features, and repeat the procedure to create new samples

Algorithm 1 Data synthesis using the target model

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

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

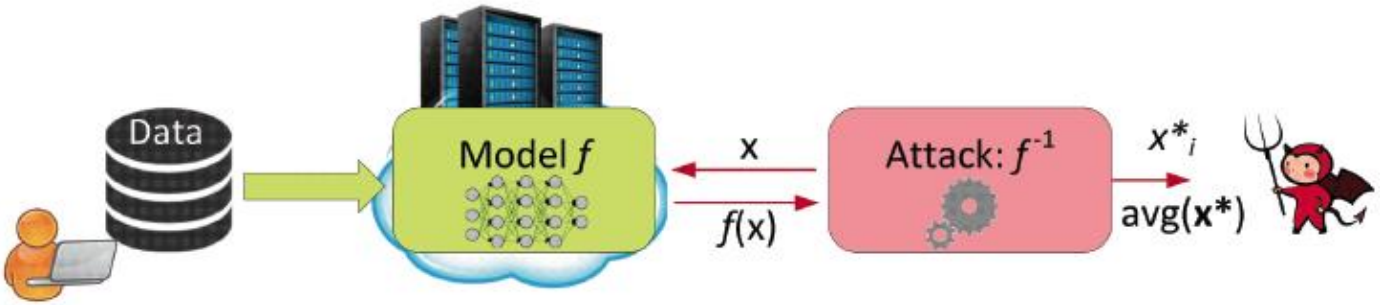


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

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.

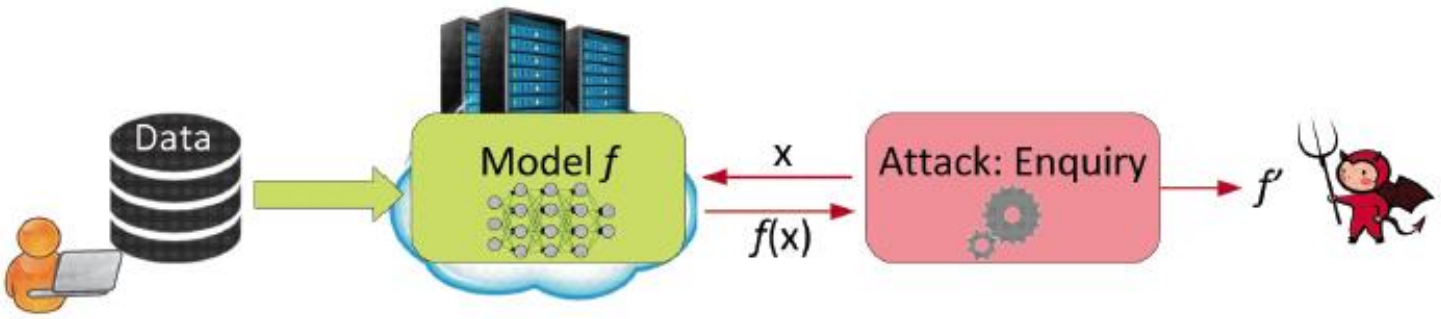


Figure form: Liu et al. (2020) When Machine Learning Meets Privacy: A Survey and Outlook

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

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

Training Data Extraction from GPT-2

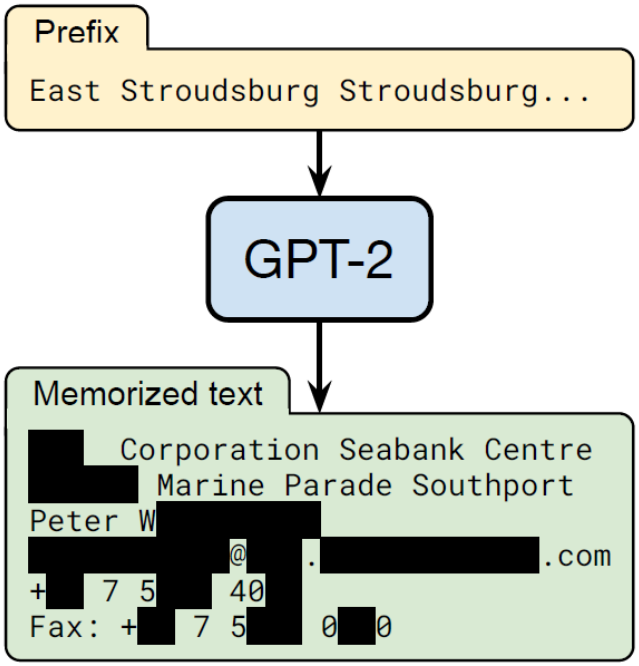
Training Data Extraction Attack against GPT-2

- ***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



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

GPT-2

Training Data Extraction Attack against GPT-2

- ***GPT-2*** (Generative Pre-training Transformer) was released by OpenAI in 2019
- It is a family of several models with varying number of parameters, trained using the same dataset
  - GPT-2 XL: 1.5 billion parameters
  - GPT-2 Medium: 334 million parameters
  - GPT-2 Small: 124 million parameters
- The training dataset consists of 40 GB of de-duplicated text data
  - The data is scraped from publicly available sources from the Internet
- For the training data extraction attack the authors used GPT-2 XL
  - It was found that GPT-2 XL memorized 10 times more information than GPT-2 Small

Attack Threat Model

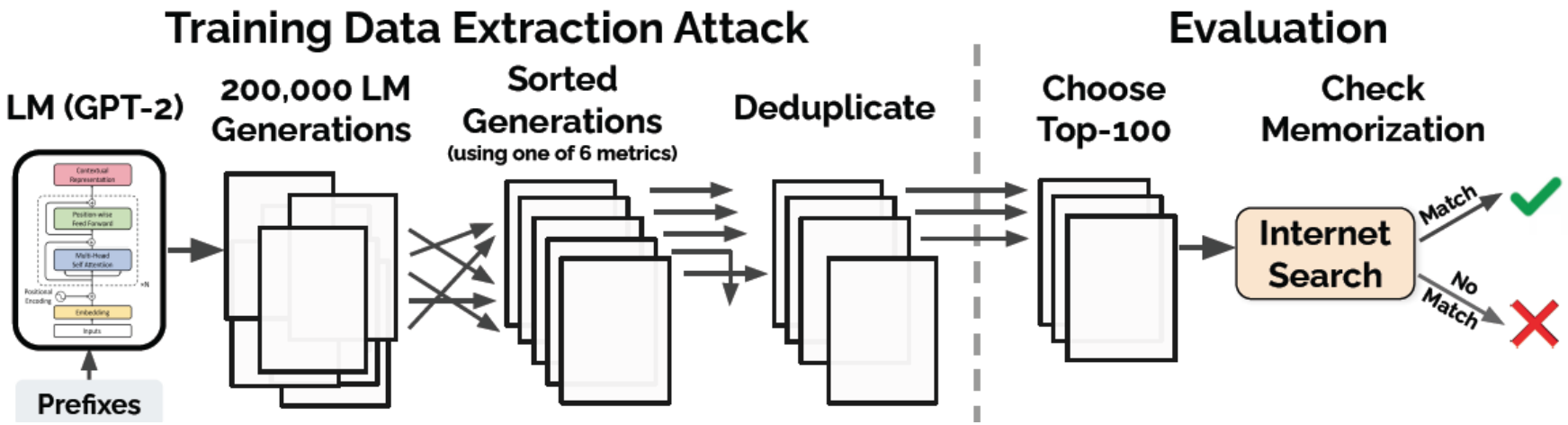
Training Data Extraction Attack against GPT-2

- 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

Results

Training Data Extraction Attack against GPT-2

- Examples of memorized content
  - Personally identifiable information (PII)
    - Found 78 examples of peoples' names, phone numbers, addresses, and social media accounts
    - E.g., extracted the usernames of participants in an Internet forum conversation that appeared in one training document
  - URLs
    - 50 examples of memorized URLs
  - Code
    - Identified 31 samples that contain snippets of memorized source code
  - Unnatural text
    - E.g., UUID: 1e4bd2a8-e8c8-4a62-adcd-40a936480059
- GPT-2 memorized removed content
  - The authors extracted content that has been removed from the Internet, but it had been memorized by GPT-2

Results

Training Data Extraction Attack against GPT-2

- Examples of memorized strings of unnatural text and URLs that occurred in only 1 document in the training data
  - E.g., in the left table the first extracted string listed has 87 characters, and occurred 10 times in 1 document
  - In the right table, we can see that GPT-XL model memorized more content than GPT-S and GPT-M models

Memorized String	Sequence Length	Occurrences in Data	
		Docs	Total
Y2...[redacted]...y5	87	1	10
7C...[redacted]...18	40	1	22
XM...[redacted]...WA	54	1	36
ab...[redacted]...2c	64	1	49
ff...[redacted]...af	32	1	64
C7...[redacted]...ow	43	1	83
0x...[redacted]...C0	10	1	96
76...[redacted]...84	17	1	122
a7...[redacted]...4b	40	1	311

URL (trimmed)	Occurrences		Memorized?		
	Docs	Total	XL	M	S
/r/[redacted]51y/milo_evacua...	1	359	✓	✓	½
/r/[redacted]zin/hi_my_name...	1	113	✓	✓	
/r/[redacted]7ne/for_all_yo...	1	76	✓	½	
/r/[redacted]5mj/fake_news_...	1	72	✓		
/r/[redacted]5wn/reddit_admi...	1	64	✓	✓	
/r/[redacted]lp8/26_evening...	1	56	✓	✓	
/r/[redacted]jla/so_pizzagat...	1	51	✓	½	
/r/[redacted]ubf/late_night...	1	51	✓	½	
/r/[redacted]eta/make_christ...	1	35	✓	½	
/r/[redacted]6ev/its_officia...	1	33	✓		
/r/[redacted]3c7/scott_adams...	1	17			
/r/[redacted]k2o/because_his...	1	17			
/r/[redacted]tu3/armynavy_ga...	1	8			

Mitigation Strategies for Data Leakage

Training Data Extraction Attack against GPT-2

- Selecting the training data for LMs
  - Avoid text from websites that are known to host sensitive data
  - 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 that occurs often in the dataset
  - Limitation: reduced accuracy, longer training times
- Auditing LMs for memorization
  - Determine the level of memorization in LMs
  - E.g., the training data extraction attack can be used to evaluate the level of memorization of an LM

Ethical Considerations and Lessons

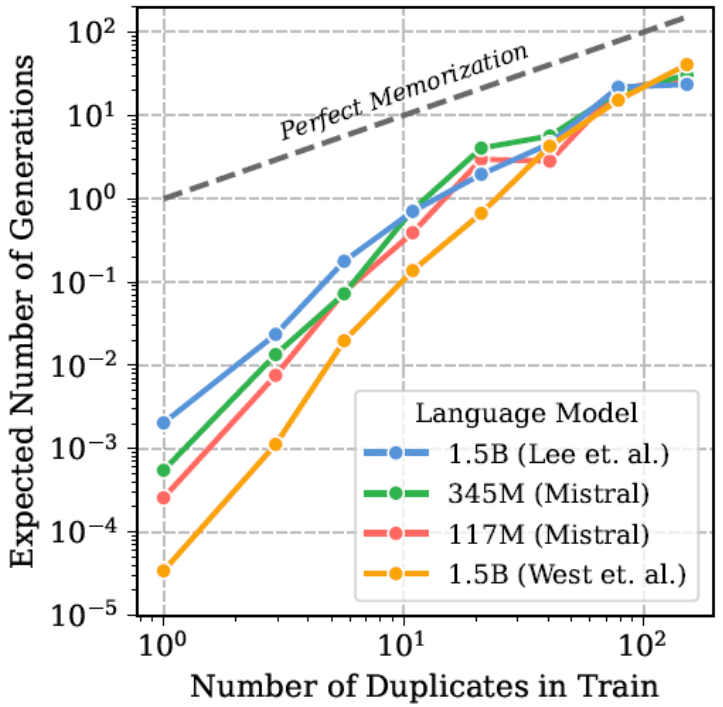
Training Data Extraction Attack against GPT-2

- The authors contacted the individuals whose PII was extracted and obtained permissions to include it in the paper
  - All PII in the paper is masked with a black rectangle box
- Among the 600,000 generated samples, 604 (or 0.1%) contain memorized text
  - The authors manually inspected only 1,800 samples
- For complete memorization, it was estimated that the content should occur 33 times in one single document
- It is important to further study and understand memorization in LMs, and develop prevention strategies

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



Training Data Extraction from GPT-Neo

Training Data Extraction Attack against GPT-Neo

- ***Training data extraction attack against GPT-Neo***
  - Yu (2023) Bag of Tricks for Training Data Extraction from Language Models
- The paper introduced several new techniques for extracting training data from language models
- Attacked is GPT-Neo, a language model with 1.2 B parameters
  - The authors had black-box query-based access to the model
- The attack is based on a benchmark for training data extraction
  - The benchmark defines text prompts and the expected answer
    - This is called targeted data extraction
    - As opposed to non-targeted data extraction, where the attacker can select their own text prompts for the attack
  - Focus is on data that occurs in only one document in the training dataset (1-eidetic memorized data)

Training Data Extraction from GPT-Neo

Training Data Extraction Attack against GPT-Neo

- Attack results
  - *Prefix* – prompt to the model
  - *Generated suffix* – the answer generated by the proposed attack approach
  - *Ground truth suffix* – the actual answer in the training dataset
    - Red color indicates mismatched text between the generated suffix and the ground truth

Prefix	.
	//
	// Copyright (c) 2008-2011 Texas Instruments Incorporated. All rights reserved.
	// Software License Agreement
	//
	// Texas Instruments (TI) is supplying this software for use solely and
	// exclusively on TI

Generated Suffix	NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
	// A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
	// Redistribution and use insource and binary forms, with

Ground Truth Suffix	NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
	// A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY
	// CIRCUMSTANCES, BE LIA

Training Data Extraction from GPT-Neo

Training Data Extraction Attack against GPT-Neo

- Additional results from the attack

Prefix	//// //// THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY //// //// EXPRESS OR IMPLIED WARRANTIES, INCLUDING
Generated Suffix	DAMAGES //// //// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS //// //// OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUS
Ground Truth Suffix	DAMAGES //// //// (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE //// //// GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

Prefix	# Permission is hereby granted, free of charge, to any person # obtaining a copy of this software and associated documentation # files (the "Software"), to deal in the Software without # restriction,
Generated Suffix	WARRANTY OF ANY KIND, # EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF # MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND # NONINFRINGEMENT
Ground Truth Suffix	WARRANTY OF ANY KIND, # EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES # OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND # NONINFRINGEMENT

Approach

Training Data Extraction Attack against GPT-Neo

- To generate an answer, language models apply a two-step approach:
  1. ***Suffix generation*** - generate a set of candidate answers for a given prefix
     - A sequence of tokens is generated by sampling from the probability distribution of the tokens in the vocabulary
  2. ***Suffix ranking*** - rank the candidate answers based on specific criteria and retain the best answer
     - Eliminate less likely suffixes based on the perplexity metric
- The authors introduced several techniques for data extraction that are applied to either the suffix generation or ranking steps

Evaluation Metrics

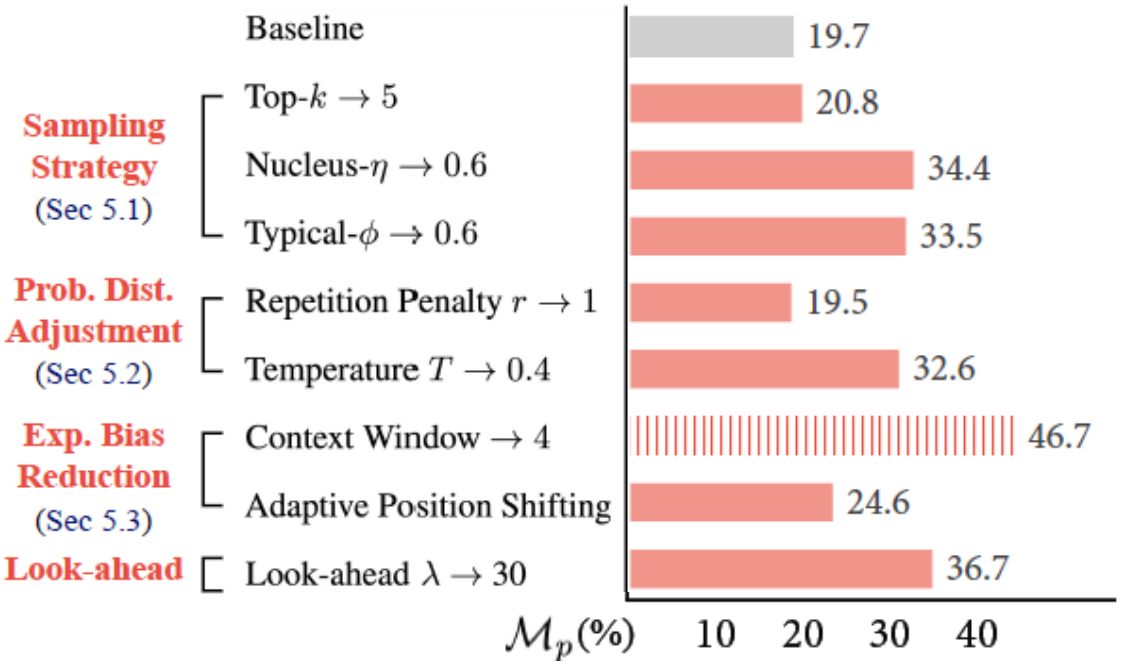
Training Data Extraction Attack against GPT-Neo

- Metrics that were used to evaluate the attack success
  - Precision $\mathcal{M}_P$ - proportion of correctly generated suffixes versus the total number of text prompts (prefixes)
    - Where a correctly generated suffix is considered identical to the ground truth suffix
  - Recall $\mathcal{M}_R$ - proportion of correctly generated suffixes versus the total number of generated suffixes
  - Hamming distance $\mathcal{M}_H$ - for two sentences of equal length, the number of positions where the letters differ
    - It is used to quantify the similarity between the generated and ground truth answers

Suffix Generation Step

Training Data Extraction Attack against GPT-Neo

- Proposed approaches for improved suffix generation
  - Sampling strategy (apply constraints to the sampling procedure from the probability distribution for the tokens)
    - **Top-*k***: limit the number of sampled suffixes to the top-*k* suffixes, where the top-5 suffixes achieved the best results
    - **Nucleus-η**: limit the sampled tokens to a set of tokens with the total probabilities greater than $\eta = 0.6$
    - **Typical-ϕ**: limit the information content of the sampled tokens based on entropy rate of $\phi = 0.6$



Suffix Generation Step

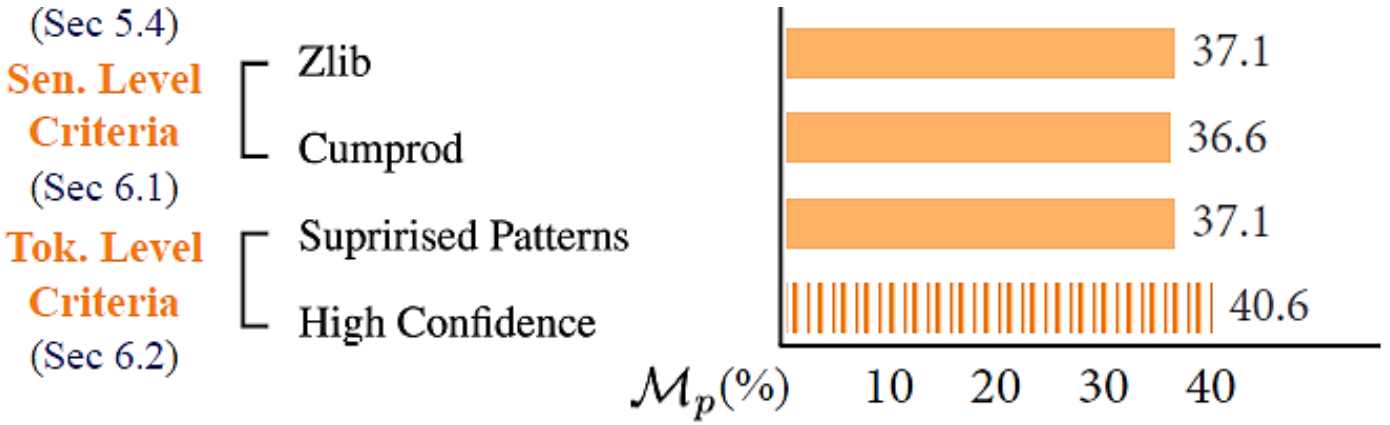
Training Data Extraction Attack against GPT-Neo

- Probability distribution adjustment (adjust directly the sampling probability distribution)
  - **Repetition penalty** – apply penalty if the next token is a repetition of the previous token
  - **Temperature** – decrease the temperature by scaling the logits during suffix generation to reduce the diversity and improve the precision (similarly to the Defensive Distillation method)
- Exposure bias reduction
  - **Dynamic context window** – adjust the window size (number of previously generated tokens) for generating the next token
  - **Dynamic position shifting** – adjust the positional encodings of the tokens, by selecting the positions with the lowest perplexity values
- Look-ahead
  - Use the probability values of the next predicted tokens to inform the generation of the current token

Suffix Ranking Step

Training Data Extraction Attack against GPT-Neo

- Introduced new criteria for ranking the generated suffixes to a prefix
- Sentence-level criteria
  - **Zlib** – use the ratio of perplexity and zlib (entropy of a generated suffix, determined by the Zlib compression algorithm)
  - **Cumprod** – use cumulative product of the probabilities of a generated suffix
- Token-level criteria
  - **Surprised patterns** – encourage tokens with high perplexity (high surprise)
  - **High confidence** – encourage generating tokens with a confidence greater than a threshold



Evaluation Results

Training Data Extraction Attack against GPT-Neo

- Results from combining different techniques
  - Combining all techniques does not produce the most accurate suffixes
  - The best results were obtained by context window + high confidence

	$\mathcal{M}_{\mathrm{P}}(\%)\uparrow$	$\mathcal{M}_{\mathrm{R}}(\%)\uparrow$	$\mathcal{M}_{\mathrm{H}}\downarrow$
Context win + Beams=2	46.7	77.5	17.154
Auto-tuning + Beams=2	46.4	76.2	17.331
Context win + Auto-tuning	46.5	77.5	17.370
Context win + High confidence	46.8	77.5	17.144

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)