Perfectly Privacy-Preserving AI
What is it and how do we achieve it?

Patricia Thaine
University of Toronto
pthaine@cs.toronto.edu

Gerald Penn
University of Toronto
gpenn@cs.toronto.edu

ABSTRACT

Many AI applications need to process huge amounts of sensitive information for model training, evaluation, and real-world integration. These tasks include facial recognition, speaker recognition, text processing, and genomic data analysis. Unfortunately, one of the following two scenarios occur when training models to perform the aforementioned tasks: either models end up being trained on sensitive user information, making them vulnerable to malicious actors, or their evaluations are not representative of their abilities since the scope of the test set is limited. In some cases, the models never get created in the first place.

There are a number of approaches that can be integrated into AI algorithms in order to maintain various levels of privacy. Namely, differential privacy, secure multi-party computation, homomorphic encryption, federated learning, secure enclaves, and automatic data de-identification. We will briefly explain each of these methods and describe the scenarios in which they would be most appropriate.

Recently, several of these methods have been applied to machine learning models. We will cover some of the most interesting examples of privacy-preserving ML, including the integration of differential privacy with neural networks to avoid unwanted inferences from being made of a network's training data.

Finally, we will discuss how the privacy-preserving machine learning approaches that have been proposed so far would need to be combined in order to achieve perfectly privacy-preserving machine learning.

1 MOTIVATION

Data privacy has been called "the most important issue in the next decade,"[1] and has taken center stage thanks to legislation like the European Union's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). Companies, developers, and researchers are scrambling to keep up with the requirements[2]. In particular, "Privacy by Design"[3] is integral to the GDPR and will likely only gain in popularity this decade. When using privacy preserving techniques, legislation suddenly becomes less daunting, as does ensuring data security which is central to maintaining user trust. Data privacy is a central issue to training and testing AI models, especially ones that train and infer on sensitive data. Yet, to our knowledge, there have been no guides published regarding what it means to have perfectly privacy-preserving AI. We introduce the four pillars required to achieve perfectly privacy-preserving AI and discuss various technologies that can help address each of the pillars. We back our claims up with relatively new research in the quickly growing subfield of privacy-preserving machine learning.

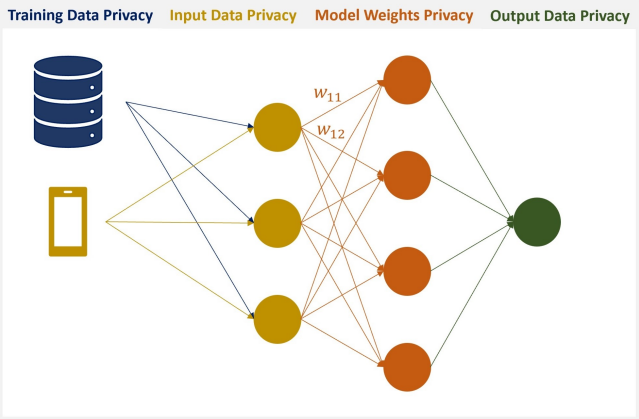

Figure 1: The Four Pillars of perfectly privacy-preserving AI.

2 THE FOUR PILLARS OF PERFECTLY-PRIVACY PRESERVING AI

During our research, we identified four pillars of privacy-preserving machine learning (Figure 1). These are:

(1) **Training Data Privacy:** The guarantee that a malicious actor will not be able to reverse-engineer the training data.
(2) **Input Privacy:** The guarantee that a user's input data cannot be observed by other parties, including the model creator.
(3) **Output Privacy:** The guarantee that the output of a model is not visible by anyone except for the user whose data is being inferred upon.
(4) **Model Privacy:** The guarantee that the model cannot be stolen by a malicious party.

While 1–3 deal with protecting data creators, 4 is meant to protect the model creator.

3 TRAINING DATA PRIVACY

While it may be slightly more difficult to gather information about training data and model weights than it is from plaintext (the technical term for unencrypted) input and output data, recent research has demonstrated that reconstructing training data and reverse-engineering models is not as huge challenge as one would hope.

[1] https://www.forbes.com/sites/marymeehan/2019/11/26/data-privacy-will-be-the-most-important-issue-in-the-next-decade/#3211e2821882
[2] https://www.theverge.com/2019/12/31/21039228/california-ccpa-facebook-microsoft-gdpr-privacy-law-consumer-data-regulation
[3] https://www.ipc.on.ca/wp-content/uploads/resources/7foundationalprinciples.pdf

PrivateNLP '20, February 7, 2020, Houston, TX, USA

Evidence

In [**?**], Carlini and Wagner calculate just how quickly generative sequence models (e.g., character language models) can memorize rare information within a training set. Carlini and Wagner train a character language model on the Penn Treebank with a "secret" inserted into it once: "the random number is ooooooooo" where ooooooooo is a meant to be a (fake) social security number. They show how the exposure of a secret which they hide within their copy of the Penn Treebank Dataset (PTD). They train a character language model on 5% of the PTD and calculate the network's amount of memorization. Memorization peaks when the test set loss is lowest. This coincides with peak exposure of the secret.

Metrics

So how can we quantify how likely it is that a secret can be reverse-engineered from model outputs? [**?**] develops a metric known as exposure:

$$\textbf{exposure}_\theta(s[r]) = \log_2(|R|) - \log_2(\textbf{rank}_\theta(s[r]))$$

Given a canary $s[r]$, a model with parameters θ, and the randomness space R, the exposure s[r] is and the rank is the index at which the true secret (or canary) is among all possible secrets given the model's perplexity for the inputs. The smaller the index, the greater the likelihood that the sequence appears in the training data, so the goal is to minimize the exposure of a secret, which is something that Carlini and Wagner. achieve by using differentially private gradient descent (see ***Solutions*** below). Another exposure metric is presented in [**?**], in which the authors calculate how much information can be leaked from a latent representation of private data sent over an insecure channel. While this paper falls more into the category of input data privacy analysis, it's still worth looking at the metric proposed to compare it with the one presented in [**?**]. In fact, they propose two privacy metrics. One for demographic variables such as sentiment analysis and blog post topic classification, and one for named entities such as news topic classification. Their privacy metrics are:

(1) Demographic variables: "$1 - X$, where X is the average of the accuracy of the attacker on the prediction of gender and age,"
(2) Named entities: "$1 - F$, where F is an F-score computed over the set of binary variables in z that indicate the presence of named entities in the input example," where "z is a vector of private information contained in a [natural language text]."

When looking at the evidence, it's important to keep in mind that this subfield of AI (privacy-preserving AI) is brand-spanking new, so there are likely a lot of potential exploits that either have not been analyzed or even haven't been thought of yet.

Solutions

There are two main proposed solutions for the problem of training data memorization which not only guarantee privacy, but also improve the generalizability of machine learning models:

(1) **Differentially Private Stochastic Gradient Descent (DPSGD) [? ?]:** While differential privacy was originally created to allow one to make generalizations about a dataset without revealing any personal information about any individual within the dataset, the theory has been adapted to preserve training data privacy within deep learning systems.
(2) **Papernot's PATE [?]:** Professor Papernot created PATE as a more intuitive alternative to DPSGD. PATE can be thought of as an ensemble approach and works by training multiple models on iid subsets of the dataset. At inference, if the majority of the models agree on the output, then the output doesn't reveal any private information about the training data and can therefore be shared.

4 INPUT AND OUTPUT PRIVACY

Input user data and resulting model outputs inferred from that data should not be visible to any parties except for the user in order to comply with the four pillars of perfectly privacy-preserving AI. Preserving user data privacy is not only beneficial for the users themselves, but also for the companies processing potentially sensitive information. Privacy goes hand in hand with security. Having proper security in place means that data leaks are much less likely to occur, leading to the ideal scenario: no loss of user trust and no fines for improper data management.

Evidence

This is important to ensure that private data do not:

- get misused (e.g., location tracking as reported in the NYT[4])
- fall into the wrong hands due to, say, a hack, or
- get used for tasks that a user had either not expected or explicitly consented to (e.g., Amazon admits employees listen to Alexa conversations[5]).

While it is standard for data to be encrypted in transit and (if a company is responsible) at rest as well, data is vulnerable when it is decrypted for processing.

Solutions

(1) **Homomorphic Encryption:** homomorphic encryption allows for non-polynomial operations on encrypted data. For machine learning, this means training and inference can be performed directly on encrypted data. Homomorphic encryption has successfully been applied to random forests, naive Bayes, and logistic regression [**?**]. [**?**] designed low-degree polynomial algorithms that classify encrypted data. More recently, there have been adaptations of deep learning models to the encrypted domain [**? ? ?**].
(2) **Secure Multi-Party Computation (MPC):** the idea behind MPC is that two or more parties' who do not trust each other can transform their inputs into "nonsense" which gets sent into a function whose output is only sensical when

[4] https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html
[5] https://www.independent.co.uk/life-style/gadgets-and-tech/news/amazon-alexa-echo-listening-spy-security-a8865056.html

the correct number of inputs are used. Among other applications, MPC has been used for genomic diagnosis using the genomic data owned by different hospitals [**?**], and linear regression, logistic regression, and neural networks for classifying MNIST images [**?**]. [**?**] is a prime example of the kind of progress that can be made by having access to sensitive data if privacy is guaranteed. There are a number of tasks which cannot be accomplished with machine learning given to the lack of data required to train classification and generative models. Not because the data isn't out there, but because the sensitive nature of the information means that it cannot be shared or sometimes even collected, spanning medical data or even speaker-specific metadata which might help improve automatic speech recognition systems (e.g., age group, location, first language).

(3) **Federated Learning:** federated learning is basically on-device machine learning. It is only truly made private when combined with differentially private training (see DPSGD in the previous section) and MPC for secure model aggregation [**?**], so the data that was used to train a model cannot be reverse engineered from the weight updates output out of a single phone. In practice, Google has deployed federated learning on Gboard (see their blog post about it[6]) and Apple introduced federated learning support in CoreML3[7].

5 MODEL PRIVACY

AI models can be companies' bread and butter, many of which provide predictive capabilities to developers through APIs or, more recently, through downloadable software. Model privacy is the last of the four pillars that must be considered and is also core to both user and company interests. Companies will have little motivation to provide interesting products and spend money on improving AI capabilities if their competitors can easily copy their models (an act which is not straightforward to investigate).

Evidence

Machine learning models form the core product and IP of many companies, so having a model stolen is a severe threat and can have significant negative business implications. A model can be stolen outright or can be reverse-engineered based on its outputs [**?**].

Solutions

(1) There has been some work on applying differential privacy to model outputs to prevent model inversion attacks. Differential privacy usually means compromising model accuracy; however, [**?**] presents a method that does not sacrifice accuracy in exchange for privacy.

(2) Homomorphic encryption can be used not only to preserve input and output privacy, but also model privacy, if one chooses to encrypt a model in the cloud. This comes at significant computational cost, however, and does not prevent model inversion attacks.

[6] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
[7] https://developer.apple.com/documentation/coreml

6 SATISFYING ALL FOUR PILLARS

As can be seen from the previous sections, there is no blanket technology that will cover all privacy problems. Rather, to have perfectly privacy-preserving AI (something that both the research community and industry have yet to achieve), one must combine technologies:

- Homomorphic Encryption + Differential Privacy
- Secure Multi-Party Computation + Differential Privacy
- Federated Learning + Differential Privacy + Secure Multi-Party Computation
- Homomorphic Encryption + PATE
- Secure Multi-Party Computation + PATE
- Federated Learning + PATE + Homomorphic Encryption

Other combinations also exist, including some with alternative technologies that do not have robust mathematical guarantees yet; namely, (1) secure enclaves (e.g., Intel SGX) which allow for computations to be performed without even the system kernel having access, (2) data de-identification, and (3) data synthesis. For now, perfectly privacy-preserving AI is still a research problem, but there are a few tools that can address some of the most urgent privacy needs.

7 PRIVACY-PRESERVING MACHINE LEARNING TOOLS

- Differential privacy in Tensorflow[8]
- MPC and Federated Learning in PyTorch[9]
- MPC in Tensorflow[10]
- On-device Machine Learning with CoreML3[11]

8 ACKNOWLEDGMENTS

Many thanks to Pieter Luitjens and Dr. Siavash Kazemian for their feedback on earlier drafts of this write-up.

[8] https://github.com/tensorflow/privacy
[9] https://github.com/OpenMined/PySyft
[10] https://github.com/mpc-msri/EzPC
[11] https://developer.apple.com/documentation/coreml