An Overview of Privacy in Machine Learning

Emiliano De Cristofaro, UCL & Alan Turing Institute
e.decristofaro@ucl.ac.uk

March 2020

1 Prologue

Over the past few years, providers such as Google, Microsoft, and Amazon have started to provide customers with access to software interfaces allowing them to easily embed machine learning tasks into their applications. Overall, organizations can now use Machine Learning as a Service (MLaaS) engines to outsource complex tasks, e.g., training classifiers, performing predictions, clustering, etc. They can also let others query models trained on their data. Naturally, this approach can also be used (and is often advocated) in other contexts, including government collaborations, citizen science projects, and business-to-business partnerships.

However, if malicious users were able to recover data used to train these models, the resulting information leakage would create serious issues. Likewise, if the inner parameters of the model are considered proprietary information, then access to the model should not allow an adversary to learn such parameters. In this document, we set to review privacy challenges in this space, providing a systematic review of the relevant research literature, also exploring possible countermeasures.

More specifically, we provide ample background information on relevant concepts around machine learning and privacy. Then, we discuss possible adversarial models and settings, cover a wide range of attacks that relate to private and/or sensitive information leakage, and review recent results attempting to defend against such attacks.

Finally, we conclude with a list of open problems that require more work, including the need for better evaluations, more targeted defenses, and the study of the relation to policy and data protection efforts.

2 Background

In this section, we provide some background information and definitions related to machine learning (ML) as well privacy technologies for ML.

2.1 Machine Learning (ML)

Learning process. ML provides automated methods of analysis for large sets of data, or to perform tasks that are too hard to program by hand. The general approach to create a machine learning model is as follows [46]:

- *Training:* Most ML models can be seen as parametric functions $h_\theta(x)$ taking an input x and a parameter vector θ. The input x is often represented as a vector of values called *features*. The space of functions $\{\forall\theta, x \rightarrow h_\theta(x)\}$ is the set of candidate hypotheses to model the distribution from which the dataset was sampled. A learning algorithm analyzes the training data to find the value(s) of parameter(s) θ. In other words, during training, an ML algorithm aims to learn the "characteristics" of some data with respect to a given "task" – e.g., given enough pictures of pets, the algorithm should learn to distinguish what differentiates pictures of, say, cats from dogs.
- *Validation:* The model performance is then validated on a test dataset, which must be disjoint from the training dataset in order to measure the model's generalization. In other words, validation performs a sanity check to make sure the algorithm has indeed learned what it is supposed to.
- *Inference (or testing):* Once training completes, the model is deployed to make predictions on inputs unseen during training: the value of parameters θ are fixed, and the model computes $h_\theta(x)$ for new inputs x. The model prediction may take different forms but, e.g. for classification, the most common is a vector assigning a probability for each class of the problem, which characterizes how likely the input belongs to that class. In other words, we can now use the algorithm in the wild, and use it for the task at hand – e.g., given a picture, the algorithm tells us whether that is a picture of a cat or a dog.

Stochastic gradient descent (SGD). With each function h and θ, one typically associates a loss $L(h, \theta)$, a value that quantifies the cost of any discrepancies between the model's prediction $h_\theta(x)$ and the true value $f(x)$, over all examples x (approximated by the training examples) [3]. Training the model h is the process of searching for a value of θ with the smallest loss, or with a tolerably small loss. Often, both the model h and the loss L are differentiable functions of θ. Therefore, training often relies on a process called stochastic gradient descent (SGD), where one repeatedly picks an example x (or batches thereof), calculates $h_\theta(x)$, and the corresponding loss, and adjusts θ to reduce the loss by going in the opposite direction of the gradient [3]. In a nutshell, SGD is a very common process used during training to find the settings in the ML algorithm that maximize its accuracy.

Tasks. The process discussed above is typical in what is referred to as supervised learning. However, ML tasks are

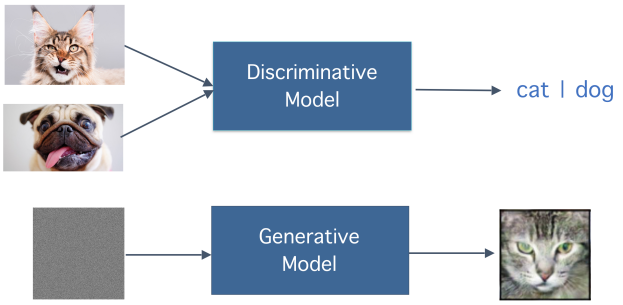

Figure 1: A simple illustration of how one can use discriminative vs generative models. The former learns to distinguish between two classes, i.e., pictures of cats or dogs. The latter estimates the underlying distribution of a dataset (pictures of cats) and randomly generate realistic, yet synthetic, samples according to their estimated distribution.

actually commonly divided into two main types, depending on the structure of the data at hand:

- *Supervised learning:* an association between inputs and outputs based on training examples in the form of inputs labeled with corresponding outputs. If the output data is categorical, the task is called classification, and real-valued output domains define regression problems [46]. Classic examples of supervised learning tasks include object recognition in images, machine translation, and spam filtering. In this setting, the model parameters are adjusted to reduce the gap between model predictions $h_\theta(x)$ and the expected output indicated by the dataset.
- *Unsupervised learning:* When the method is given unlabeled inputs, its task is unsupervised. Unsupervised learning considers problems such as clustering points according to a similarity metric, dimensionality reduction to project data in lower dimensional subspaces, etc [46].

Other types of techniques include *reinforcement learning*, which we omit to ease presentation.

ML approaches. ML models can be also categorized depending on the probability distributions they learn. Assuming one has some input data x and wants to classify it into labels y, then, roughly speaking, one can use either:

- *Discriminative models* to learn the conditional probability distribution $p(y|x)$, or
- *Generative models* to learn the joint probability distribution $p(x, y)$. Among these, Generative Adversarial Networks (GANs) [22] were originally proposed for unsupervised learning, but now also used for supervised and reinforcement learning, and have become very popular as they learn to generate new data with the same statistical properties as the training set. The two kinds of model are exemplified in Figure 1.

Another distinction is based on whether the learning task is centralized or (somewhat) distributed:

- *Centralized learning:* in conventional ML methodologies, all training data is pooled and stored at a single entity, and models are trained on this joint pool.

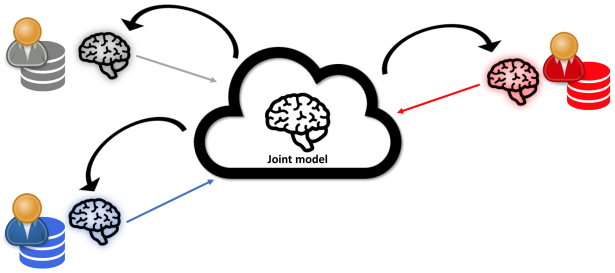

Figure 2: An overview of the federated learning approach.

- *Collaborative/federated learning:* multiple participants, each with their own training dataset, construct a joint model by training a local model on their own data, but periodically exchange model parameters, updates to these parameters, or partially constructed models with the other participants. This intuition is illustrated in Figure 2. There are several techniques in this category, including federated learning deployed by Google [34] on million of devices, e.g., for training of predictive keyboards on character sequences that users type on their phones.

ML Algorithms. As discussed above, ML involves creating models. These can be built using different approaches, sometimes referred to as algorithms. Supervised learning techniques include Linear Regression, Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machines, Random Forest, Naive Bayes, etc. For instance, Logistic Regression is a linear model where the probabilities describing the possible outcomes of a single trial are modeled via a logistic (logit) function; the parameters of the model are estimated with maximum likelihood estimation, using an iterative algorithm. Unsupervised learning algorithms include clustering ones–e.g., hierarchical clustering, k-means, DBSCAN, etc.

Neural networks, which have attracted a lot of attention in recent years, are also types of ML instantiations, where learning can be either supervised or unsupervised. A neural net consists of a large number of simple processing nodes that are densely interconnected, usually organized into layers, to which weights are assigned. During training, weights are initially set to random values; as training data passes through the layers, weights are continually adjusted until training data with the same labels consistently yield similar outputs [25]. In particular, in deep neural networks ("deep learning"), each level learns to transform its input data into a slightly more abstract and composite representation; in fact, the word "deep" typically refers to the number of layers through which the data is transformed.

Overall, choosing the right machine learning algorithm depends on several factors, including, but not limited to: data size, quality and diversity, as well as what answers businesses want to derive from that data. Additional considerations include accuracy, training time, parameters, data points and much more. Therefore, the right choice is typically both a combination of business need, specification, experimentation, and time available.

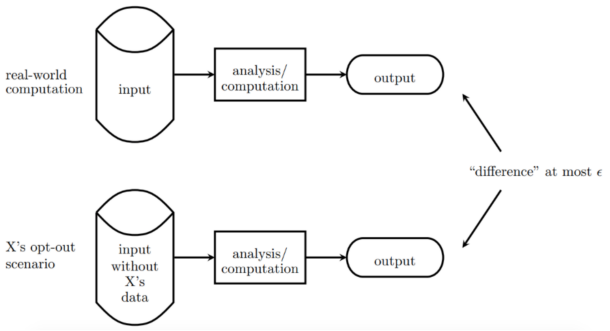

Figure 3: High-level illustration of differential privacy: an adversary can distinguish between computation done on a dataset with or without data X only with at most a small probability ϵ.

2.2 Machine Learning as a Service (MLaaS)

Many cloud providers, including Microsoft, Amazon, and IBM, have launched Machine Learning as a Service (MLaaS) offerings, aimed to help clients benefit from machine learning without the cost, time, and risk of building in-house infrastructure from scratch. MLaaS offers ready-made, generic machine learning tools, such as predictive analytics, APIs, data visualization, and natural language processing, that can be adapted by small and medium sized companies according to their needs.

Users who purchase MLaaS services can access these tools via prediction APIs on a pay-per-query basis. Typical image classification service costs around \$1–\$10 per 1,000 queries, depending on the customization and sophistication of the machine learning model.

MLaaS services vary a lot across different providers. In some cases, providers enable clients to download and deploy machine learning models locally, while others only allow clients to access machine learning models via a prediction query interface, which provides both the predicted label and the confidence score. The latter is much more popular. Finally, some platforms also allow clients to upload their own models and charge others for using their models.

2.3 Privacy-Enhancing Technologies for Machine Learning

2.3.1 Cryptography

Cryptography, and more precisely encryption, can be used to protect data confidentiality. There are two main primitives that are relevant in the context of ML and in general data analysis/processing: 1) *secure multi-party computation* (SMC), and 2) *fully homomorphic encryption* (FHE).

SMC allows two or more parties to jointly compute a function over their inputs, while keeping those inputs hidden from each other. Typically, SMC protocols build on tools like garbled circuits, secret sharing, oblivious transfer (for a detailed overview of such tools, we refer the reader to https://securecomputation.org).

Whereas, FHE is an encryption scheme that allows processing of the underlying cleartext data while it still remains in encrypted form, and without giving away the secret key. In other words, FHE allows (almost) arbitrary computation over encrypted data.

2.3.2 Differential Privacy (DP)

DP addresses the paradox of learning nothing about an individual while learning useful information about a population [16]. Generally speaking, it provides rigorous, statistical guarantees against what an adversary can infer from learning the result of some randomized algorithm.

Typically, differentially private techniques protect the privacy of individual data subjects by adding random noise when producing statistics. In other words, DP guarantees that an individual will be exposed to the same privacy risk whether or not her data is included in a differentially private analysis, as illustrated in Figure 3.

Definition. Formally, for two non-negative numbers ϵ, δ, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if and only if, for any neighboring datasets D and D' (i.e., differing at most one record), and for the possible output $S \subseteq Range(\mathcal{A})$, the following formula holds:

$$\Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(D') \in S] + \delta$$

The ϵ, δ parameters. Differential privacy analysis allows for some information leakage specific to individual data subjects, controlled by the privacy parameter ϵ. This measures the effect on each individual's information on the output of the analysis. With smaller values of ϵ, the dataset is considered to have stronger privacy, but less accuracy, thus reducing its utility. An intuitive description of ϵ privacy parameter, along with examples, is available in [40].

Sensitivity. The notion of the sensitivity of a function is very useful in the design of differentially private algorithms, and define the notion of sensitivity of a function with respect to a neighboring relationship. Given a query F on a dataset D, the sensitivity is used to adjust the amount of noise required for $F(D)$. More formally, if F is a function that maps a dataset (in matrix form) into a fixed-size vector of real numbers, we can define the L_i-sensitivity of F as: $S_i(F) = \max_{D,D'} ||F(D) - F(D')||_i$, where $||\cdot||_i$ denotes the L_i norm, $i \in \{1, 2\}$ and D and D' are any two neighboring datasets.

3 Privacy in ML

The security of any system is measured with respect the adversarial goals and capabilities that it is designed to defend against; to this end, we now discuss different threat models. Then, we attempt to provide a definition of privacy in ML, focusing on the different types of attacks, which are reviewed in detail in Section 4.

3.1 Adversarial Models

Overall, we focus on the privacy of the model. *NB: adversarial examples are out of the scope of this document.* In the rest of this section, we discuss adversarial goals related to extracting information about the model or training data.

When the model itself represents intellectual property—e.g., in financial market systems—the model and its parameters should be kept private. In other contexts, it is imperative that the privacy of the training data be preserved, e.g., medical applications. Regardless of the goal, the attacks and defenses for them relate to exposing or preventing the exposure of the model and training data.

Access. We first discuss what kind of *access* the attacker might have:

- *White-Box:* she has some information about the model or its original training data, e.g., ML algorithm h, model parameters θ, network structure, or summary, partial, or full training data.
- *Black-Box:* she has no knowledge about the model. Rather, she might explore a model by providing a series of carefully crafted inputs and observing outputs.

Inference vs training. Another variable is *where* the attack might take place:

- *Training Phase:* the adversary attempts to learn the model, e.g., accessing a summary, partial or all of the training data. She might create a substitute model (aka auxiliary model) to use to mount attacks on the victim system.
- *Inference Phase:* the adversary collects evidence about the model characteristics by observing inferences made by it.

Passive vs Active. Finally, one can also distinguish between passive and active attacks, roughly mirroring the traditional distinction in security literature between honest-but-curious and fully malicious adversaries. Consider for instance federated learning, where the attacker is one of the participants in the collaborative setting:

- *Passive attack:* the adversary passively observes the updates and performs inference, e.g., without changing anything in the training procedure;
- *Active attack:* the adversary actively changes the why she operates, e.g., in the case of federated learning, by extending their local copy of the collaboratively trained model with an augmented property classifier connected to the last layer.

3.2 Types of attacks

Before delving into the state of the art in actual attacks, we define what privacy means in the context of machine learning or, alternatively, what it means for a machine learning model to breach privacy. Specific attacks proposed in literature are then reviewed in Section 4.

3.2.1 Inference about members of the population

- *Statistical disclosure:* the adversary learns something about the input to the model from the model; in theory, one would like to control statistical disclosure (this is also known as the "Dalenius desideratum" [11]), in that a model should reveal no more about the input to which it is applied than would have been known about this input without applying the model. However, as also pointed out in [52], this cannot be achieved by any useful model [15].
- *Model inversion:* an adversary can use the model's output to infer the values of sensitive attributes used as input to the model. Note that it may not be possible to prevent this if the model is based on statistical facts about the population : for example, suppose that training the model has uncovered a high correlation between a person's externally observable phenotype features and their genetic predisposition to a certain disease; this correlation is now a publicly known fact that allows anyone to infer information about the person's genome after observing that person [52].
- *Inferring class representatives:* overall, model inversion can be generalized to potential breaches where the adversary, given some access to the model, infers features that characterize each class, making it possible to construct representatives of these classes.

3.2.2 Inference about members of the training dataset

Here the focus is on privacy of the individuals whose data was used to train the model. This motivation is closely related to the original goals of differential privacy [14]. Of course, members of the training dataset are members of the population, too. Therefore, one should focus on what the model reveals about them beyond what it reveals about an arbitrary member of the population:

- *Membership inference:* given a model and an exact data point, the adversary infers whether this point was used to train the model or not.
- *Property inference:* training data may not be identically distributed across different users whose records are in the training set; unlike model inversion, the adversary tries to infer properties that are true of a *subset* of the training inputs but not of the class as a whole. For instance, when Bob's photos are used to train a gender classifier, she infers that Alice appears in some of the photos.

3.2.3 Inferring Model Parameters

As discussed earlier, MLaaS allows model owners to charge others for queries to their commercially valuable models. This pay-per-query deployment option exemplifies an increasingly common tension: on the one hand, the query interface of an ML model may be widely accessible, yet the model itself and the data on which it was trained may be proprietary and confidential. Moreover, for security applications such as spam or fraud detection, an ML model's secrecy is critical to its utility; an adversary that can learn the model can also often evade detection [5].

In this space, we can distinguish between:

- *Model Extraction:* a black-box adversary that can query an ML model to obtain predictions on input feature vectors, and may or may not know the model type (e.g., logistic regression) or the distribution over the

data used to train the model. The adversary's goal is to extract an equivalent or near-equivalent ML model.

- *Functionality Stealing:* Rather than stealing the model, here the ultimate goal is to create "knock-offs" of the (black-box) model solely based on input-output pairs observed from MLaaS queries [43].

4 Attacks

4.1 Membership Inference Attack (MIA)

As mentioned above, membership inference relates to the problem of deciding, given a data point, whether or not it was included in the training dataset.

4.1.1 Definition and Relevance

Formally, in a membership inference attack (MIA), the adversary, given a target datapoint d^* and *some* access to a model $h_\theta(x)$, tries to infer whether $d^* \in x$. This can constitute a serious privacy breach in a number of settings, which we discuss next.

Sensitivity of task/model. First of all, MIA can directly violate privacy if inclusion in a training set is itself sensitive based on the nature of the task at hand. For example, if health-related records (or images like MRIs) are used to train a classifier, discovering that a specific record was used for training inherently leaks information about the individual's health. Similarly, if images from a database of criminals are used to train a model predicting the probability that one will re-offend, successful membership inference exposes an individual's criminal history.

Signal of leakage. When a record is fully known to the adversary, learning that it was used to train a particular model is an indication of information leakage through the model. Overall, MIA is often considered to be a signal—a measuring stick of sort—that access to a model leads to potentially serious privacy breaches. In fact, MIAs are often a gateway to further attacks: e.g., if the adversary infers that data of a victim is part of the information she has access to she can mount other attacks, like profiling, property inference, etc.

Establishing wrongdoing. On the other hand, MIA can also be used by regulators to support the suspicion that a model was trained on personal data without an adequate legal basis, or for a purpose not compatible with the data collection. For instance, DeepMind was recently found to have used personal medical records provided by the UK's National Health Service for purposes beyond direct patient care; the basis on which the data was collected [57].

MIA beyond machine learning. As a side note, we remark that MIAs have been studied not only in the context of machine learning, but also in other fields. Overall, given a data point d^* and a function $f(x)$, one can define membership inference as the problem of determining whether d^* is part of the input to the function f, i.e., $d^* \in x$. Often, f is some aggregation function, and in fact researchers have demonstrated the existence of successful MIAs against aggregate statistics in the context of genomic studies [28], location data [49], and noisy statistics in general [17].

4.1.2 State of the Art

Attacking Machine Learning as a Service. MIA against black-box machine learning models was first studied by Shokri et al. [52], in the context of supervised learning. They focus on classification models trained by commercial Machine Learning as a Service (MLaaS) providers, such as Google and Amazon, whereby a user has API access to a trained model.

More specifically, customers in possession of a dataset and a data classification task can upload the dataset to the MLaaS service and pay it to construct a model. The service then makes the model available to the customer—typically as a black-box API. This setting is illustrated in Figure 4. For example, a mobile-app maker can use such a service to analyze users' activities and query the resulting model inside the app to promote in-app purchases to users when they are most likely to respond. Moreover, some machine-learning services also let data owners expose their models to external users for querying or even sell them.

Inference via overfitting. Shokri et al. [52]'s approach exploits differences in the model's response to inputs that were or were not seen during training. For each class of the targeted black-box model, they train a *shadow model*, with the same machine learning technique; the intuition is that the model ends up "overfitting" on data used for training [52]. Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points, and thus performs better on the training inputs than on the inputs drawn from the same population but not used during the training. Therefore, the attacker can exploit the confidence values on inputs belonging to the same classes and learn to infer membership.

Then, Salem et al. [50], although also building on overfitting signals, relax several adversarial assumptions—e.g., using multiple shadow models, knowledge of the target model structure, and having a dataset from the same distribution as the target model's training data—showing that MIAs are very broadly applicable at low cost and thereby pose a more severe risk than previously thought. Overall, the attacks are relatively successful across the board, more or less so depending on the adversary's prior knowledge, the datasets, the task at hand, etc.

ML and overfitting. Another line of work studies the relationship between overfitting and information leakage. In particular, Yeom et al. [59] assume that the adversary knows the distribution from which the training set was drawn and its size, and that the adversary colludes with the training algorithm. Their attacks are close in performance to Shokri et al.'s [52], and show that, besides overfitting, the influence of target attributes on model's outputs also correlates with successful attacks. Truex et al. [56] extend [52] to a more general setting and show how membership inference attacks are largely transferable.

Generative models. While the research discussed above focus on discriminative models, other work target generative models. As discussed in Section 2.1, they are used to generate new samples from the same underlying distribution of a given training dataset, e.g., to artificially generate plausible images and videos. Here the attacker targets a

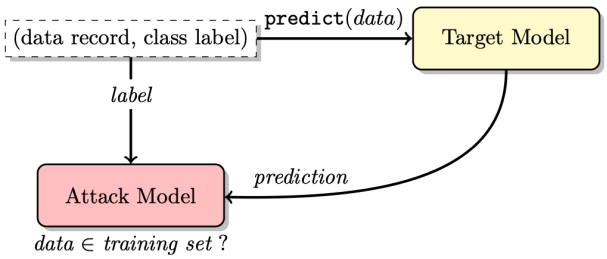

Figure 4: Membership inference attack against MLaaS in the black-box setting [52].

MLaaS engine that provides synthetic samples on demand – e.g., the user's query is *"provide an image sample of a cat"* – based on a trained generative model. Once again, inferring whether specific data points are part of the training set for that generative model may constitute a serious privacy breach. Note that membership inference on generative models is much more challenging than on discriminative models: in the former, the attacker cannot exploit confidence values on inputs belonging to the same classes, and therefore it is more difficult to detect overfitting and mount the attack.

Hayes et al. [26] consider both black-box and white-box attacks: in the former, the adversary can only make queries to the model under attack, i.e., the target model, and has no access to the internal parameters. In the latter, she also has access to the parameters. To mount the attacks, they train a Generative Adversarial Network (GAN) [22] on samples generated from the target model; i.e,, using generative models as a method to learn information about the target generative model, and thus create a local copy of the target model from which they can launch the attack. The intuition is that, if a generative model overfits, then a GAN—which combines a discriminative model and a generative model—should be able to detect this overfitting, since the discriminator is trained to learn statistical differences in distributions. Moreover, for white-box attacks, the attacker-trained discriminator itself can be used to measure information leakage of the target model.

Federated Learning. In this setting, the attack can be mounted by an adversary, a participant in the federated learning, attempting to infer whether a specific record is part of the training set of either a specific or any participant. The first MIA against federated learning is presented by Melis et al. [36], whose main intuition is to exploit unintended leakage from either the *embedding layer* (all deep learning models operating on non-numeric data where the input space is discrete and sparse first use an embedding layer to transform inputs into a lower-dimensional vector representation) or the *gradients* (in deep learning models, gradients are computed by back-propagating the loss through the entire network from the last to the first layer). An illustration of Melis et al. [36]'s attack is in Figure 5. Then, Nasr et al. [38] design MIAs during the training phase in a white-box setting, including passive and active attackers based on the different adversary prior knowledge.

4.2 Model Inversion

As mentioned earlier, model inversion techniques aim to infer class features and/or construct class representatives, given that the adversary has *some* access (either black-box or white-box) to a model.

4.2.1 Definition and Early Work

The concept of model inversion is introduced by Fredrikson et al. [18, 19]. First, they show how an attacker can rely on outputs from a classifier to infer sensitive features used as inputs to the model itself: given the model and some demographic information about a patient whose records are used for training, an attacker might predict sensitive attributes of the patient. Then, they use so-called "hill-climbing" on the output probabilities of a computer-vision classifier to reveal individual faces from the training data.

These techniques are sometimes described as violating privacy of the training data, even though the inferred features characterize an entire class and not specifically the training data, except in the cases of pathological overfitting where the training sample constitutes the entire membership of the class.

4.2.2 Further Attacks

Collaborative learning. Hitaj et al. [27] show that a participant in collaborative learning can use GANs to construct class representatives. However, this technique has been evaluated only on models where all members of the same class are visually similar (handwritten digits and faces). Thus, there is no evidence that it produces actual training images or can distinguish a training image and another image from the same class.

Aono et al. [4] show that, in the collaborative deep learning protocol of [51], an adversarial server can partially recover participants' data points from the shared gradient updates, although in a greatly simplified setting where the batch consists of a single data point.

Unintended Memorization. Song et al. [53] engineer a machine learning model that memorizes the training data, which can then be extracted with black-box access to the model, without affecting the accuracy of the model on its primary task.

Then, Carlini et al. [9] show that deep learning-based generative sequence models trained on text data can unintentionally memorize specific training inputs, which can then be extracted with black-box access. Even though the models are trained on text, extraction is demonstrated only for sequences of digits (artificially introduced into the text), which are not affected by the relative word frequencies in the language model.

4.3 Property Inference

As mentioned above, work presented in [5, 18, 27] aimed to infer properties that characterize an entire class: for example, given a face recognition model where one of the classes is Bob, infer what Bob looks like (e.g., Bob wears glasses). However, while Ateniese et al. [5] are actually

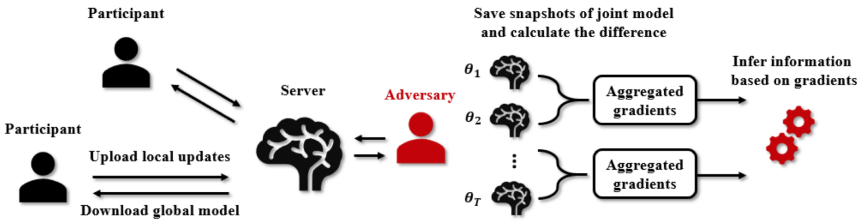

Figure 5: Inference attacks against federated learning (passive adversary) by Melis et al. [36].

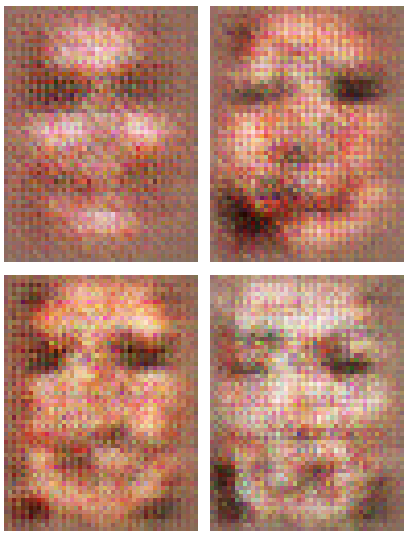

Figure 6: Samples from a GAN attack on a gender classification model where the class is "female."

the first, to the best of our knowledge, to reason about extracting "something meaningful relating to properties of the training set," it is not clear that hiding this kind of information in a good classifier is possible or desirable.

4.3.1 Attacks

Melis et al. [36]'s work. By contrast, here we focus on the adversarial goal of inferring properties that are *true of a subset of the training inputs but not of the class as a whole*. For instance, when Bob's photos are used to train a gender classifier, can the attacker infer that Alice appears in some of the photos? In particular, Melis et al. [36] focus on the properties that are *independent* of the class's characteristic features. In contrast to the face recognition example, where "Bob wears glasses" is a characteristic feature of an entire class, in their gender classifier study they infer whether people in Bob's photos wear glasses—even though wearing glasses has no correlation with gender. There is no "legitimate" reason for a model to leak this information; it is purely an artifact of the learning process.

The work in [36] studies this kind of property inference in the context of collaborative/federated learning. More specifically, their intuition is that a participant's contribution to each iteration of collaborative learning is based on a batch of their training data, and the adversary can infer *single-batch properties*, i.e., detect that the data in a given batch has the property but other batches do not. She can also infer *when a property appears in the training data*, which has very serious privacy implications. For instance, the adversary can infer when a certain person starts appearing in a participant's photos or when the participant starts visiting a certain type of doctors. Finally, they infer properties that characterize a participant's entire dataset (but not the entire class), e.g., authorship of the texts used to train a sentiment-analysis model.

Follow-up work. Ganju et al. [20] also develop property inference attacks, mainly against fully connected, relatively shallow neural networks. They focus on the post-training, white-box release of models trained on sensitive data, and the properties inferred by the adversary may or may not be correlated with the main task.

4.4 Model and Functionality Stealing

Finally, we look into adversarial efforts toward inferring model parameters.

4.4.1 Model Extraction

The concept of model stealing, or extraction, is first presented by Tramer et al. [55]. In this kind of attack, an adversary with black-box access, but no prior knowledge of an ML model's parameters or training data, aims steal the model parameters. The intuition behind their attack is to exploit the information-rich outputs returned by the ML prediction APIs, e.g., high-precision confidence values in addition to class labels.

Consider the case of ML algorithms like logistic regression: the confidence value is a simple log-linear function $1/(1+e^{-(w\cdot x+\beta)})$ of the d-dimensional input vector x. By querying $d+1$ random d-dimensional inputs, an attacker can with high probability solve for the unknown $d+1$ parameters w and β defining the model. (Such equation-solving attacks extend to multi-class logistic regressions and neural networks).

Overall, Tramer et al. [55]'s work is focused on inferring model parameters. In follow-up work, other researchers have then gone a step further and perform hyperparameter stealing [58], architecture extraction [41], etc. In the former, the focus is on hyperparameters, rather than parameters, which are configurations external to the model and whose values cannot be estimated from data. In the latter, a black-box adversary succeeds to infer (hidden) model architectures (e.g., the type of non-linear activation) of neural networks in MLaaS as well as their optimization processes (e.g., stochastic gradient descent or ADAM).

4.4.2 Functionality Extraction

As mentioned in Section 3.2, the goal of functionality extraction is, rather than to steal the model, to create "knock-offs." In [43], Orekondy et al. do so solely based on input-output pairs observed from MLaaS queries. More specifically, the adversary interacts with a black-box "victim" Convolutional Neural Network (CNN) by providing it input images and obtaining respective predictions. The resulting image-prediction pairs are used to train a knock-off model, e.g., to compete with the victim model at the victim's task.

Additional work in this category includes [45], whereby the adversary trains a local model to substitute for a victim deep neural network (DNN), using inputs synthetically generated by an adversary and labeled by the target DNN.

4.5 Take Aways

Throughout this section, we have presented and discussed the state of the art in attacks against privacy in ML.

Membership inference attacks are real. As evident from the above discussion, there has been a very significant amount of research work on membership inference attacks against ML. Arguably, this is motivated by 1) the seriousness of the privacy risks stemming from such attacks, 2) the fact that MIA is often just a signal of leakage and can serve as a canary for broad privacy issues, and 3) the interesting challenges in making attacks more effective, less reliant on strong assumptions, etc.

Overall, several attacks have been proposed in the context of a wide variety of datasets (images, text, etc.), models (discriminative, generative, federated), as well as threat models (API access, white-box, black-box, active, passive, etc.). Such attacks are realistic but obviously their effectiveness depends on the actual settings, e.g., adversary's knowledge of records, model parameters, etc., and are likely to affect certain users more than others.

While we discuss possible defenses and the overall outlook, respectively, in Section 5 and **??**, we are confident in arguing that MIAs are a real problem that at the very least should make practitioners and researchers question whether deploying ML models in the wild is a good idea, privacy-wise, whenever training data is sensitive. However, further work is needed to provide clear guidelines and usable tools for practitioners willing to provide access to trained model to fully understand the privacy risks, on their specific data/specific learning task, for the users whose data is used for training.

Limitations of model inversion. Although research roughly falling in the "model inversion" category is important, we believe there are some limitations in what they mean for privacy. Class members produced by model inversion and GANs are similar to the training inputs only if all members of the class are similar, as is the case for MNIST (the dataset of handwritten digit used in [27]) and facial recognition. This does not violate privacy of the training data; it simply shows that machine learning works as it should. A trained classifier reveals the input features characteristic of each class, thus enabling the adversary to sample from the class population. For instance, Figure 6 shows GAN-constructed images for the gender classification task on the Labeled Faces in the Wild (LFW) dataset, taken from [36]. These images show a generic female face, but there is no way to tell from them whether an image of a specific female was used in training or not.

Therefore, the informal property violated by such attacks is, roughly speaking: "a classifier should prevent users from generating an input that belongs to a particular class or even learning what such an input looks like." However, it is not clear why this property is desirable or whether it is even achievable. In fact, this motivated us to study what we defined as property inference attacks.

Property inference needs further work. Overall, property inference attacks are not to be ignored, even though, their effectiveness depends on the context. As mentioned earlier, inferring sensitive attributes is really a privacy breach when the attacker can confidently assess that those attributes related to records in the training set, and even more so if they do not leak simply because the class the model is learning to classify is strictly correlated.

So really the only "attack" in this sense we are aware of is that of Melis et al. [36], which has only been studied in the context of collaborative learning. Even in that case, the authors essentially show that the accuracy of the attack quickly degrades with increasing number of participants. In fact, if this is large enough, then differentially private defenses based on the moments accountant method [1] (discussed in Section 5) can be used to thwart such attacks.

It remains however an open research questions to investigate whether property inference attacks: 1) are possible, as per our definition, in non collaborative learning settings and at scale, and 2) can be thwarted in collaborative settings involving a small number of participants. (We discuss the latter in more detail Section 5.)

5 Defenses

Overall, we can categorize defenses against attacks discussed in this document and in general aiming to protect confidentiality and privacy in ML based on the main tools they rely on. These include advanced *privacy-enhancing technologies* like cryptography and differential privacy (reviewed in Section 2.3) as well approaches used as part of the learning process (mainly, training) to reduce information available to the adversary.

5.1 Cryptography

Cryptography in ML can support confidential computing scenarios where, for instance, a server has a model trained on its private data and wishes to provide inferences (e.g., classification) on clients' private data. In this context, there are a number of research proposals and prototypes in literature, which allow the client to obtain the inference result without revealing their input to the server, while at the same preserving the confidentiality of the server's model. For instance, privacy-enhancing tools based on secure multi-party computation (SMC) and fully homomorphic encryption (FHE) have been proposed to securely train supervised machine learning models, such as matrix factor-

ization [39], linear classifiers [7, 23], decision trees [8, 32], linear regressors [13], and neural networks [6, 12, 33, 37].

SMC has also been used to build privacy-preserving neural networks in a distributed fashion. For instance, SecureML [37] starts with the data owners (clients) distributing their private training inputs among two non-colluding servers during the setup phase; the two servers then use MPC to train a global model on the clients' encrypted joint data. Then, Bonawitz et al. [6] use secure multi-party aggregation techniques, tailored for federated learning, to let participants encrypt their updates so that the central parameter server only recovers the sum of the updates.

Confidentiality vs Privacy. Overall, cryptography in ML is really aimed at protecting *confidentiality*, rather than *privacy*, which constitutes the main focus of our report. The two terms are often confused, both in the context of ML and in general, but they actually refer to different properties. Confidentiality is an explicit design property whereby one party wants to keep information (e.g., training data, testing data, model parameters, etc.) hidden from both the public and other parties (e.g., clients with respect to servers or vice-versa). Whereas, privacy is about protecting against *unintended* information leakage, whereby an adversary aims to infer sensitive information through some (intended) interaction with the victim. In other words, cryptographically-enforced confidential computing does not provide any guarantees about what the output of the computation reveals. Therefore, we will focus on privacy rather than confidentiality defenses.

5.2 Differential Privacy (DP)

The state-of-the-art method for providing access to information free from inferences is to satisfy differential privacy (DP) [14]. This applies to ML as well, and more precisely to providing access to models that have been trained on (sensitive) datasets [2, 44, 47, 48, 51]. More precisely, there are two main privacy-preserving model-training approaches in literature: 1) using noisy stochastic gradient descent (noisy SGD) [3], and 2) Private Aggregation of Teacher Ensembles (PATE) [44, 47].

Noisy SGD. As mentioned in Section 2.1, ML models are often used trained using stochastic gradient descent (SGD). The intuition is to add noise to the SGD process; however, the main challenge is to do so while ensuring that the noise is carefully calibrated. The sensitivity of the final value of θ (the parameter vector) to the elements of the training data is generally hard to analyze. On the other hand, since the training data affects θ only via the gradient computations, one may achieve privacy by bounding gradients (by clipping) and by adding noise to those computations. In a nutshell, this is the idea behind the seminal work by Abadi et al. [1], in particular in the accounting of the privacy loss, and hence it is general referred to as the *Moments Accountant* algorithm.

PATE. To protect the privacy of training data during learning, PATE transfers knowledge from an ensemble of "teacher" models trained on partitions of the data to a "student" model; see Figure 7. Intuitively, privacy is provided by training teachers on disjoint data, and strong guarantees stem from noisy aggregation of teachers' answers.

Collaborative Learning. In the collaborative learning setting, Shokri and Shmatikov [51] support distributed training of deep learning networks in a privacy-preserving way. Specifically, their system relies on the input of independent entities which aim to collaboratively build a machine learning model without sharing their training data. To this end, they selectively share subsets of noisy model parameters during training. Moreover, federated learning proposals, which have already mentioned, tackle the problem of training deep learning models with differential privacy guarantees for the tasks of training language models [35] and digits classification [21].

5.3 Trusted Execution Environments

A different line of work focuses on privacy (as well as integrity) guarantees for ML computations in untrusted environments (i.e., tasks outsourced by a client to a remote server, including MLaaS) by leveraging so-called Trusted Execution Environments (TEEs), such as Intel SGX or ARM TrustZone. TEEs use hardware and software protections to isolate sensitive code from other applications, while attesting to its correct execution. The main idea is that TEEs outperform purely cryptographic approaches by multiple orders of magnitude.

In this area, there are three main approaches. The first includes work supporting oblivious data access patterns [42] and in general training for a range of ML algorithms run inside SGX [29, 30]. The second, by Tramer and Boneh [54], focuses on *high performance* execution of Deep Neural Networks (DNNs) in TEEs, by efficiently partitioning DNN computations between trusted and untrusted devices. The third, by Hanzlik et al. [24], is essentially a guarded offline deployment of MLaaS: models are executed locally on the client's side (therefore, the data never leaves the device).

5.4 ML-Specific Approaches

Finally, a number of ML techniques are used to reduce information available to the adversary to mount their attacks. For instance, *dropout* is a popular technique often used to mitigate overfitting in neural networks; as such, this might reduce the effectiveness of MIAs based on overfitting.

Additional techniques in this space include weight normalization (re-parameterization of the weights vectors that decouples the length of those weights from their direction), dimensionality reduction (e.g., only using inputs that occur many times in the training data), selective gradient sharing (in collaborative learning, participants could share only a fraction of their gradients during each update), etc.

6 Conclusion

This document provided a literature review on privacy and machine learning. We provided ample background information on relevant concepts, including machine learning, differential privacy, etc., as well as adversarial models. Then, we presented a wide range of attacks that relate to

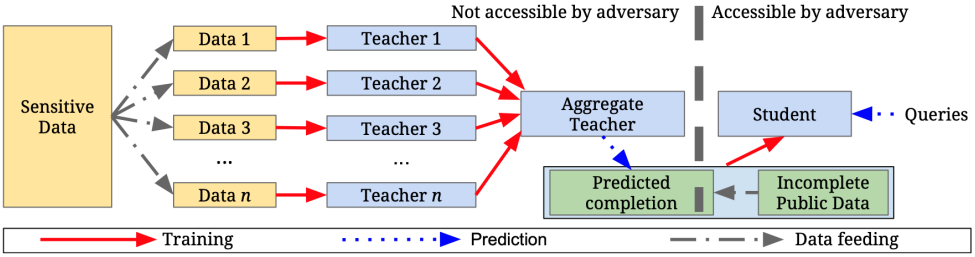

Figure 7: Overview of PATE's approach [47].

private and/or sensitive information leakage. Finally, we discussed recent results attempting to defend against such attacks. Next, we list three areas where further work is desperately needed.

6.1 Severity of Attacks

Overall, our review showed that there exists a large body of literature presenting a wide range of different attacks. Specifically, we covered membership inference attacks (MIAs), model inversion, property inference, as well as model and functionality stealing attacks.

We concluded that membership inference attacks are very much possible, but it is hard to grasp the real-world effect on actually deployed models due to the lack of case studies vis-a-vis impact on actual users, relation to adversary's prior knowledge, etc. We also offered a critic of model inversion attacks, while covering less studied attacks in the field of property inference and model stealing. Much work is left to be done – especially considering ways to provide guidelines and evaluation framework for practitioners.

6.2 Policy Implications and Further Study Needed

The implication of the attacks covered in this manuscript vis-a-vis policy and data protection is also largely unexplored. The only exception in this context is the work by Cohen and Nissim [10], which rephrases privacy attacks in the General Data Protection Regulation (GDPR) framework and more specifically within the *"singling out"* concept. While the GDPR heavily focuses on the concept of identification, what it means for a person to be "identified, directly or indirectly" is not clear. As pointed in [10], Recital 26 sheds a little more light: "To determine whether a natural person is identifiable account should be taken of all the means reasonably likely to be used, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly."

Therefore, singling out is one way to identify a person in data, and only data that does not allow singling out may be excepted from the regulation. Clearly, more work linking up privacy attacks (and defenses) with regulation and data protection efforts needs to ramp up.

6.3 Need for Better Evaluations

Overall, several defense techniques against privacy attacks have been proposed over the past few years. However, it is very hard to assess how generalizable they are and what is the trade-off they incur with respect to privacy and utility. This prompts the need for a more thorough evaluation of how defenses fare in practice, vis-a-vis realistic use cases and datasets, rather than the standard public ones that, more often than not, say little or nothing about real-world performance.

In this context, some recent work has taken some good steps in the right direction; for instance, Jayaraman and Evans [31] study the impact of variable choices of the ϵ parameter, different variants of differential privacy, and several learning tasks on both utility and privacy (including in the context of MIAs) for privacy-preserving machine learning. Alas, however, their main finding is that there is no way to obtain privacy for free?relaxed definitions of differential privacy that reduce the amount of noise needed to improve utility also increase the privacy leakage. In other words, current mechanisms for differentially private machine learning rarely offer acceptable utility-privacy trade-offs for complex learning tasks: settings that provide limited accuracy loss provide little effective privacy, and settings that provide strong privacy result in useless models.

Once again, this points to the need to better understand where trade-offs are possible, in what context, and at what expenses, rather than hoping to deploy generic, one-size-fits-all defenses across the board.

Ackwnoledgments. The author wish to thanks Vitaly Shmatikov, Luca Melis, Jamie Hayes, and Yang Zhang for valuable discussions and feedback.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *ACM CCS*, 2016.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

[3] M. Abadi, U. Erlingsson, I. Goodfellow, H. B. McMahan, I. Mironov, N. Papernot, K. Talwar, and L. Zhang. On the protection of private information in machine learning systems: Two recent approches. In *IEEE CSF*, 2017.

[4] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al. Privacy-preserving deep learning: Revisited and Enhanced. In *ATIS*, 2017.

[5] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN*, 2015.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B.

McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS*, 2017.

[7] J. W. Bos, K. Lauter, and M. Naehrig. Private Predictive Analysis on Encrypted Medical Data. *Journal of Biomedical Informatics*, 50:234–243, 2014.

[8] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. Technical report, Cryptology ePrint Archive Report 2014/331, 2014.

[9] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song. The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets. *arXiv preprint 1802.08232*, 2018.

[10] A. Cohen and K. Nissim. Towards Formalizing the GDPR's Notion of Singling Out. *arXiv preprint arXiv:1904.06009*, 2019.

[11] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15(429-444), 1977.

[12] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.

[13] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *IEEE ICDM*, 2004.

[14] C. Dwork. Differential privacy: A survey of results. In *TAMC*, 2008.

[15] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1), 2010.

[16] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9, 2013.

[17] C. Dwork, A. Smith, T. Steinke, J. Ullman, and S. Vadhan. Robust traceability from trace amounts. In *FOCS*, 2015.

[18] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, 2015.

[19] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security*, 2014.

[20] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *CCS*, 2018.

[21] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint 1712.07557*, 2017.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[23] T. Graepel, K. Lauter, and M. Naehrig. Ml confidential: Machine learning on encrypted data. In *ICISC*, 2013.

[24] L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz. Mlcapsule: Guarded offline deployment of machine learning as a service. *arXiv:1808.00590*, 2018.

[25] L. Hardesty. Explained: Neural networks. http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414, 2017.

[26] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1).

[27] B. Hitaj, G. Ateniese, and F. Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *ACM CCS*, 2017.

[28] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet*, 2008.

[29] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv:1803.05961*, 2018.

[30] N. Hynes, R. Cheng, and D. Song. Efficient deep learning on multi-source private data. *arXiv:1807.06689*, 2018.

[31] B. Jayaraman and D. Evans. Evaluating Differentially Private Machine Learning in Practice. In *USENIX Security*, 2019.

[32] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, 2000.

[33] J. Liu, M. Juuti, Y. Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *ACM CCS*, 2017.

[34] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[35] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. In *ICLR*, 2018.

[36] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

[37] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, 2017.

[38] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, 2019.

[39] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *Security & Privacy*, 2013.

[40] K. Nissim, T. Steinke, A. Wood, M. Altman, A. Bembenek, M. Bun, M. Gaboardi, D. R. OBrien, and S. Vadhan. Differential privacy: A primer for a non-technical audience. In *Privacy Law Scholars Conference*, 2017.

[41] S. J. Oh, M. Augustin, M. Fritz, and B. Schiele. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[42] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security*, 2016.

[43] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *IEEE CVPR*, 2019.

[44] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[45] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical Black-Box Attacks against Machine Learning. In *AsiaCCS*, 2017.

[46] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and privacy in machine learning. In *IEEE EuroS&P*, 2018.

[47] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. In *ICLR*, 2018.

[48] N. Phan, X. Wu, and D. Dou. Preserving differential privacy in convolutional deep belief networks. *ML*, 2017.

[49] A. Pyrgelis, C. Troncoso, and E. De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *NDSS*, 2018.

[50] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*, 2019.

[51] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *ACM CCS*, 2015.

[52] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, 2017.

[53] C. Song, T. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *ACM CCS*, 2017.

[54] F. Tramer and D. Boneh. SLALOM: Fast, verifiable and private execution of neural networks in trusted hardware. In *ICLR*, 2019.

[55] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security*, 2016.

[56] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. Towards demystifying membership inference attacks. *arXiv:1807.09173*, 2018.

[57] J. Vincent. Google DeepMind will use machine learning to spot eye diseases early. https://www.theverge.com/2016/7/5/12095830/google-deepmind-nhs-eye-disease-detection, 2016.

[58] B. Wang and N. Z. Gong. Stealing hyperparameters in machine learning. In *IEEE Symposium on Security and Privacy*, 2018.

[59] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE CSF*, 2018.