WHITEPAPER

PRIVACY-PRESERVING MACHINE LEARNING

A PRACTICAL GUIDE

OCTOBER 2020

ALEXANDRA INSTITUTE

AUTHORS

Zaruhi Aslanyan, the Alexandra Institute

Panagiotis Vasilikos, the Alexandra Institute

Published by

THE ALEXANDRA INSTITUTE

October 2020

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

A key enabler for Artificial Intelligence (AI) techniques is the collection of large amounts of data, and Machine Learning (ML) – at the core of AI – exploits such data to produce predictive models. However, gathering data and using them to predict behaviours presents great challenges to the privacy of individuals and organisations, such as data breaches, privacy loss, and the corresponding financial and reputational damages. Already today, “much of the most privacy-sensitive data analysis – such as search algorithms, recommendation engines, and adtech networks – are driven by machine learning” [1].

Privacy-preserving machine learning aims at bridging the gap between preserving privacy and reaping the benefits of ML. It is a key enabler for privatising collected data and complying with data protection regulations.

In this paper, we introduce the main concepts of privacy-preserving machine learning. We present techniques that combine machine learning and privacy techniques and review some of the available tools.

The paper is intended as a practical guide to privacy-preserving machine learning for a reader without prerequisite knowledge of machine learning or data privacy. However, it may also be of interest to the specialist as an overview of the subject. In particular, it may prove useful to profiles responsible for data privacy, such as security managers or machine learning specialists.

2 INTRODUCTION

The European Commission [2] states that “artificial intelligence will change our lives by improving healthcare, increasing the efficiency of farming, contributing to climate change mitigation and adaptation, improving the efficiency of production systems through predictive maintenance, increasing the security of Europeans, and in many other ways that we can only begin to imagine”.

According to Gartner, artificial intelligence-based technologies have gained more and more focus in recent years from both industry and society (Figure 1). One such key technology is Machine Learning (ML), centred on the processing of large amount of data to produce predictive models.

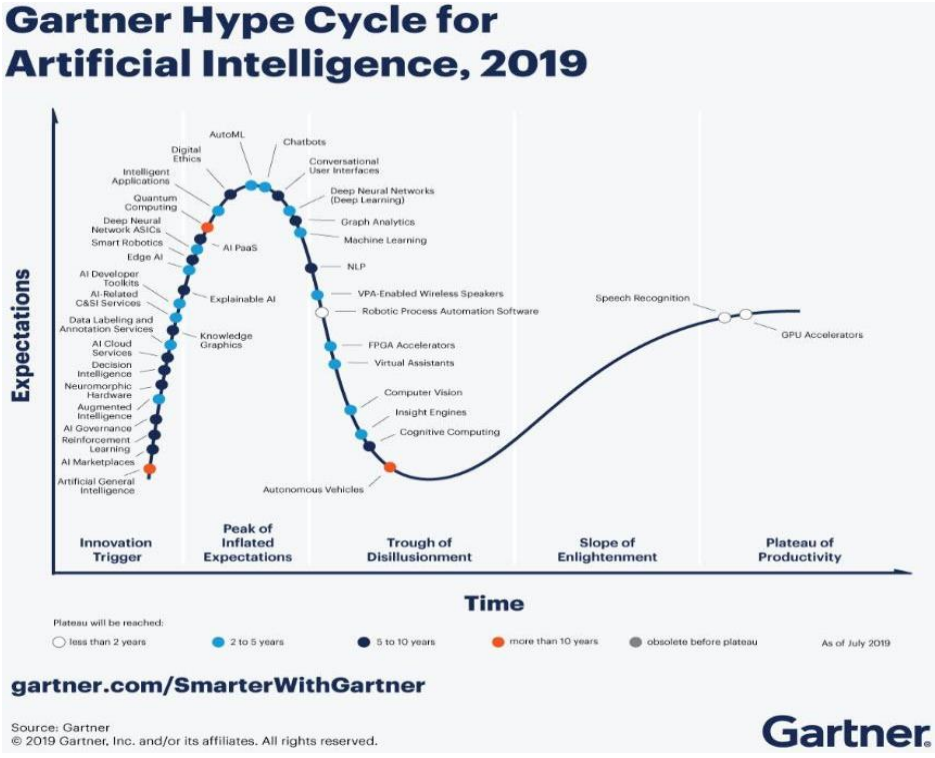

Figure 1: Gartner Hype Cycle for Artificial Intelligence.
Illustration from https://www.forbes.com/sites/louiscolumbus/2019/09/25/ whats-new-in-gartners-hype-cycle-for-ai-2019/

ML algorithms are at the heart of a great many applications that impact our daily life, such as medical diagnosis or spam and malware filtering (Figure 2). For example, an ML algorithm can help identify skin cancer [3]. The key benefit of ML algorithms is the ability to process large amount of data for identifying and extracting patterns and trends, thereby allowing to build systems that adapt to a changing environment with little or no human intervention. However, in order to build an accurate and reliable model, and to obtain precise predictions, ML algorithms require significant amount of data for training.

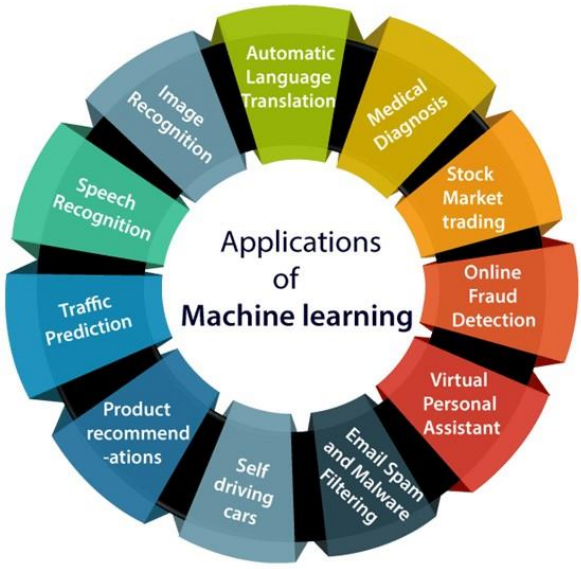

Figure 2: Applications of Machine Learning. Illustration from https://www.javatpoint.com/applications-of-machine-learning

With the advantages of ML applications, enormous data privacy issues come as well – consider, for instance, applications for healthcare or intrusion detection. Cyberattacks and data breaches have become increasingly common and costly to handle. Large collections of data stored for training purposes attract cybercriminals, who want to steal data that can be used to identify individuals or other valuable information that can be sold. Moreover, ML models themselves constitute a threat, as it is possible to extract sensitive information from them. For example, Shokri et al. show how to identify whether a record was used in the training dataset for a given ML model. They tested their technique on Amazon and Google Cloud machine learning systems and obtained 74% and 94% accuracy, respectively [4].

In this context, a critical issue is the protection of personally-identifiable information (PII), that is, data which can be used to identify a given person. Besides protecting PII from potential leakage, companies should comply with various data protection regulations, such as the General Data Protection Regulation (GDPR) in Europe. In case of violation of GDPR, the imposed penalties can be significant.

Besides threatening the end users to whom the information refers, cyberattacks bring legal, financial and reputational risks to the organisations collecting data. Simply removing PII from a dataset, such as names and addresses, would not suffice, for there

are other quasi-identifiers that can be used to pinpoint an individual in a dataset. A known example is the re-identification of William Weld, governor of Massachusetts, from presumably anonymised health data records containing only birth date, sex and ZIP code, in a study carried out by Latanya Sweeney [5].

Privacy-preserving ML aims at overcoming these issues by enhancing ML with techniques that safeguard data privacy. These include perturbation techniques, such as differential privacy, cryptographic approaches, such as homomorphic encryption and multi-party computation, and ML-specific approaches, such as federated learning.

The choice of a privacy-preserving ML technique depends on various factors, such as the use-case, the assets to be protected, etc. There is no silver-bullet solution; understanding the various factors helps identify the right approach. Moreover, we have to balance scenario-specific considerations with the need for building portable and reusable approaches.

While in recent years the research on privacy-preserving ML has flourished and significant progress has been made, there is still a gap between theoretical advancements and their applicability to real-world cases.

This paper provides an overview of the key concepts in privacy-preserving machine learning to provoke the reader and raise awareness. The aim of this paper is to guide organisations and individuals who want to protect data in ML applications, providing an overview of the available techniques and suggesting which to select in order to protect given assets.

The rest of the paper is organised as follows: in Section 3, we introduce the required background on ML, while we also discuss privacy goals and threats in the context of ML. In Section 4, we give an overview of privacy-preserving techniques, while in Section 5, we discuss how these techniques can be utilised in ML applications in order to achieve privacy. In Section 5.5, we compare different privacy-preserving techniques for ML, while we also discuss various works resulting from the combinations of a number of techniques. Finally, in Section 6, we refer to public libraries and tools which can be used to implement privacy-preserving ML solutions. Our concluding remarks are in Section 7.

3 MACHINE LEARNING

In this section, we begin by describing the core steps and roles involved throughout an ML process. We then discuss different privacy goals for ML processes, while we also describe threat models and attacks which can compromise them.

3.1 MACHINE LEARNING PROCESS

Traditional problem-solving using computers involves the creation of a program $\mathcal{M}$, which given some input information x, produces an output y, where y can be understood as the answer to our problem. Whenever the program is deterministic, it can be seen as a function $\mathcal{M} : X \mapsto Y$ from an input set X to an output set Y (Figure 3 top). However, as the complexity of problems increases, constructing precisely such a program $\mathcal{M}$ can be a daunting task.

To tackle the latter, the field of ML provides us with mathematical processes, which can automatically construct an approximation of the program of interest. In particular, given a dataset Z that contains information related to the program input space X, an ML process can build a parametric *model* $\mathcal{M}_\theta : X \mapsto Y$, where θ is the model's parameter (Figure 3 bottom).

As an example, consider the medical diagnosis problem given in [6], where by taking a dataset Z, which contains X-ray images, the goal of an ML process is to construct a model $\mathcal{M}_\theta : X \mapsto Y$ that given an arbitrary X-ray image would determine whether a patient has cancer or not. In this example, the input set X contains all the possible X-ray images encoded as pixel vectors, while the output set $Y = \{0,1\}$ consists only of the two elements 0 and 1, with 0 representing that a patient does not have cancer, whereas 1 indicates that the patient has cancer.

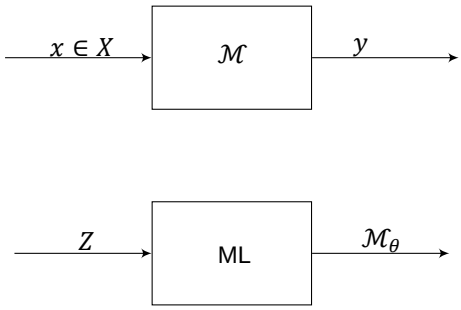

Figure 3: Machine Learning versus traditional programming.

At the core of an ML process there is an algorithm $\mathcal{A}$ which *trains* the model on Z or on a set of *features* extracted by Z, and eventually produces the final model $\mathcal{M}_\theta$. Common ML algorithms are Random Forest, Neural Networks, Decision Trees, Logistic Regression, Support Vector Machine, K-Nearest Neighbors, KMeans, Gradient boosting, etc.[1]. Finally, the choice of the algorithm depends on the problem specifics, as well as other factors such as the size of the dataset Z, the algorithm execution time and accuracy, etc.

Training the model on a set of features instead of training it directly on Z may have additional benefits, such as boosting the performance of the training process. For instance, in the cancer diagnosis example, the set of features can be produced by removing certain pixels of the X-ray images, where it is known that they don't contain information which is correlated with the presence of cancer. The overall training time can now be reduced since we need to process fewer pixels.

In addition to the set of features used for training the model, some ML algorithms require annotating the elements in the feature set with their corresponding outputs. These annotations are called *labels*. This is known as supervised learning, whereas ML processes that do not use label annotations is known as unsupervised learning. For instance, going back to the cancer diagnosis example, some of the X-ray images may have been collected from patients who have already been diagnosed with cancer, and thus their corresponding images can be labelled with 1.

Finally, the readiness and maturity of the produced model is *validated* on a test set of features which is also extracted by Z. The test set should not contain any element used for training the model.

In general, most ML processes involve six fundamental steps, as depicted in Figure 4 and described below.

[1] *The details of those algorithms are out of the scope of this paper.*

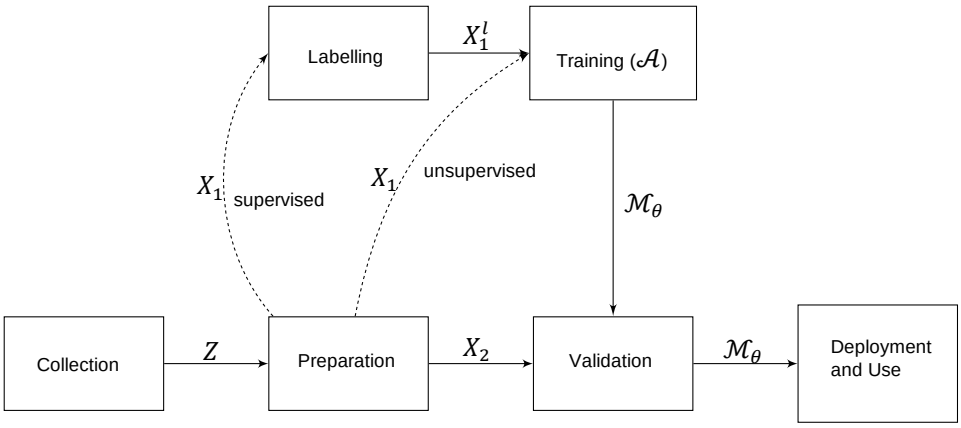

Figure 4: The fundamental steps involved in an ML process.

- **Data Collection**. The dataset Z is collected. The data collection step is very crucial since the model accuracy relies vastly on the amount and the quality of the data collected. Most ML processes require a large amount of data in order to produce an accurate model. From now on, we will refer to Z as the *raw dataset*, since it has not undergone any processing yet.

- **Data Preparation**. The raw dataset Z is processed in order to be "cleaned" and for extracting *features* which are relevant for producing the final model $\mathcal{M}_\theta$. In particular, this processing produces two new datasets, the *training* dataset X_1 and the *validation* set X_2 where $X_1 \subseteq X$, $X_2 \subseteq X$ and X_1 and X_2 are mutually disjoint (i.e. $X_1 \cap X_2 = \emptyset$). Sometimes it is the case that the raw dataset Z is no further prepared, and in this case, we have that Z is directly partitioned in the training and the validation set i.e., $Z = X_1 \cup X_2$.

- **Data Labelling**. In this step, the training dataset X_1 is labelled i.e., each element $x \in X_1$ is associated with an element $y \in Y$, with y being the model's output on x, and we write $X_1^l \subseteq X_1 \times Y$ for the resulting labelled set. This step applies only if the supervised learning approach is used.

- **Training**. The algorithm $\mathcal{A}$ is fed with X_1 (or X_1^l in the case of supervised learning) and starts building the model $\mathcal{M}_\theta$. By the end of this stage, the algorithm $\mathcal{A}$ has tuned the parameter θ and the model is ready for validation. Very often, in the background of the training process, a loss function is used, which for each choice of the parameter θ provides a metric about the model's deviation from the ideal program $\mathcal{M}$. Thus, training a model can be seen as an optimization problem where one looks for the parameter θ which minimises the loss function. A widely-used technique for improving the model's accuracy using loss functions is Stochastic Gradient Descent (SGD).

- **Validation**. The dataset X_2 is used to validate the accuracy of the model $\mathcal{M}_\theta$. This step is also known as the model's generalisation, where the model $\mathcal{M}_\theta$ is exposed to new unseen inputs.

- **Deployment and Use**. Finally, the model $\mathcal{M}_\theta$ is ready for deployment. Once the model is deployed, one can query an input $x \in X$ and receive the response $y = \mathcal{M}_\theta(x) \in Y$.

Actors with different roles participate in an ML process, and each role may observe or manipulate information involved throughout the different ML steps. In particular, we have four main roles: (1) the *data contributors*, (2) the *model creators*, (3) the *model owners* and (4) the *model consumers*.

Data Contributors. Data contributors play a vital role in an ML process since they are providing the raw dataset Z. Data contributors are mainly humans who make use of a company's or organisation's service or product, which requires the collection, processing and storage of an individual's information.

For instance, an online shop may store information related to its customers' shopping habits, or a smartwatch may collect information about one's fitness activities. The collected datasets may be available through public repositories, or they can be accessible only internally in the company or organisation which provides the service. Data contributors are involved in the data collection step.

Model Creators. Model creators consist mainly of the data scientists and data engineers that are responsible for building the model $\mathcal{M}_\theta$. They are the ones who perform the tasks involved in the data collection step and up to the model validation step. The role of the model creator requires the highest access rights to the datasets collected from the data contributors.

Model Owners. The model owner is usually the company or the organisation who either offers the final model $\mathcal{M}_\theta$ as a service or uses it for its own benefit internally. For instance, in the cancer diagnosis example, a private hospital may provide the model that predicts cancer as a service to its patients at additional fees, while an online shop could benefit from a model which suggests products to their customers based on their past purchases.

Model Consumers. Finally, the model consumers are the ones who are able to interact and use the model once it has been deployed. The consumers are able to provide part of or the entire input $x \in X$ to the model $\mathcal{M}_\theta$ and observe the output $y = \mathcal{M}_\theta(x)$.

3.2 PRIVACY GOALS IN MACHINE LEARNING

An ML process involves many assets which may need to be protected. In this paper, we focus on the following assets: (1) The identity of the data contributors, (2) the raw dataset Z collected by them, (3) the feature datasets X_1 and X_2 extracted by Z, (4) the model $\mathcal{M}_\theta$ itself, and (5) its input (or part of it) whenever the model is queried.

Identity Privacy. Protecting the data contributor's identity is necessary in many data collection processes, especially when the data collected may reveal political beliefs, sexual preferences, health conditions etc. Achieving identity privacy can be seen as ensuring anonymity of the data contributors both from the model creators and the model

owners, who are highly involved in the data collection stage of an ML process, but also from adversaries which in many cases are acting as model consumers.

Raw Dataset Privacy. Often, the raw dataset Z contains sensitive information, such as the *X*-ray images from the cancer diagnosis example, and if compromised the patient's privacy can be violated. Such privacy violations could also introduce further new attack vectors. For instance, consider a scenario where an ML model can calculate energy efficient plans, and previously it has been trained on a dataset which contains the energy consumption and location of houses. Compromising this dataset can expose the time intervals in which the residents of a specific house are absent. The latter can then be used as input in defining strategies for robberies. Often, compromising the privacy of the raw dataset could also result in compromising identity privacy, since data in Z can be used to identify a specific data contributor. A common malpractice which eases the task of compromising Z is storing the dataset as plaintext instead of using state of the art encryption.

Feature Datasets Privacy. The privacy of the feature datasets is as important as the privacy of the raw dataset. This is because both the training set X_1 and the validation set X_2 are extracted from Z, and in case of them being compromised, information about Z could also be recovered. Unfortunately, the privacy of the feature datasets is threatened in many ML models, which by default store the set X_1 (or the labelled set X_1^l, in the case of supervised learning models) inside the model.

Model Privacy. Privacy of the model $\mathcal{M}_\theta$ refers to the secrecy of the produced model $\mathcal{M}_\theta$ and its parameter θ. Keeping the model secret is a common goal desired by the model owners, who want to stay ahead of their competitors. However, exposing part of the model's functionality to the model consumers is inevitable, since observing the responses to queries may reveal information about the internal structure of the model.

Input Privacy. Often the input (or part of it) to the model may be sensitive and needs to be protected. Therefore, only the output of the model is revealed to the model consumers. For instance, consider the scenario of an ML model that takes as input medical records to predict if a certain disease can be inherited. In such cases, it is likely that the medical record contributors do not desire to disclose their medical records to each other. In many cases, achieving input privacy requires not only keeping the input x secret, but also limiting the information about x disclosed through the response $y = \mathcal{M}_\theta$.

3.3 THREAT MODELS AND ATTACKS

We now briefly discuss the adversary models and the type of attacks that target the assets of ML processes. We mainly distinguish between two different adversary models (1) *white box adversaries* who know the structure of the model $\mathcal{M}_\theta$, its parameter θ, while they may also know part of the raw dataset Z and finally, they can interact with the model (i.e., the adversary is a model consumer in this case); and (2) *black box adversaries* who have no knowledge about the model and its parameters but they can query the model and observe its response.

Membership Inference Attacks [4,7,8]. In this type of attack, the adversary wants to determine if a given data point z was part of the raw dataset Z, or if a particular data point

x was part of the training set X_1. Membership inference attacks usually exploit that ML models often behave differently on the data used for their training versus unseen data. Membership inference attacks can compromise the raw dataset privacy or the feature datasets privacy. For instance, this attack could be used by an adversary to learn whether a specific individual's record was used to train an ML model which determines the presence of a certain disease. This attack can be performed by a black box adversary.

De-anonymisation Attacks [9, 10, 11]. This attack aims to identify an individual who has contributed his data into a dataset. Many model owners publish the raw dataset Z or the feature datasets X_1 and X_2 , which may allow an adversary to compromise the identity privacy of the data contributors, even though the dataset has been first anonymised (see Section 4.1). This attack requires white box access to the dataset. However, it can also be performed using black-box access by chaining other attacks, which are discussed below.

Reconstruction Attacks [12, 13]. The goal of this attack is to construct the raw dataset Z by reverse engineering the feature training dataset X_1 or the validation dataset X_2. This attack can compromise the raw dataset privacy. In general, it requires white box access to a model that hard-codes the feature datasets inside it.

Model Extraction Attacks [14, 15]. The goal of this attack is to construct an approximation $\widehat{\mathcal{M}_\theta}$ of the model $\mathcal{M}_\theta$. The more accurate the approximation is, the more successful the attack is. In this type of attack, the adversary has black box capabilities and trains his model $\widehat{\mathcal{M}_\theta}$ by collecting pairs of queries and their responses received by the model $\mathcal{M}_\theta$. This attack violates the model privacy.

Model Inversion Attacks [16, 17, 18]. Model inversion attacks aim at inferring properties about the training dataset X_1, even when it is not explicitly stored in the model $\mathcal{M}_\theta$. This type of attack can even be performed by a limited black box attacker who can interact with the model by querying it and collecting its responses. It is also common that an adversary may first perform a model extraction attack and then utilise the model $\widehat{\mathcal{M}_\theta}$ to accomplish a model inversion attack.

4 PRIVACY-PRESERVING TECHNIQUES

In this section, we will briefly describe the privacy-preserving techniques that can be used to strengthen ML algorithms to protect individuals' data.

4.1 ANONYMISATION

Data anonymisation aims at protecting the privacy of an individual in a given dataset. Before releasing a dataset, the first step is to remove all direct identifiers, such as name, address and phone numbers. However, in most of the cases this is not enough. Only removing the direct identifiers will still allow an attacker to re-identify an individual in the dataset by linking the data with other additional information that he/she has.

In order to avoid such cases, additional anonymisation steps should be enforced. Several methods, such as k-anonymity and l-diversity, have been studied that make it harder for an adversary to learn anything about an individual from an anonymised dataset.

For a great many cases l-diversity together with k-anonymity provides a robust privacy model. An intuitive definition of these techniques is as follows: k-anonymity is an anonymisation method that ensures the information about an individual in the published dataset cannot be distinguished from at least k-1 other individuals in the same dataset. On the other hand, a dataset is called l-diverse if every equivalence class has at least $l \geq 2$ different values for the sensitive attributes.

It is worthwhile observing that perfectly-anonymised data are not useful. We should always balance between privacy and utility of the database, as presented in Figure 5. Though challenging, anonymisation needs to be applied aiming at providing data privacy while preserving utility.

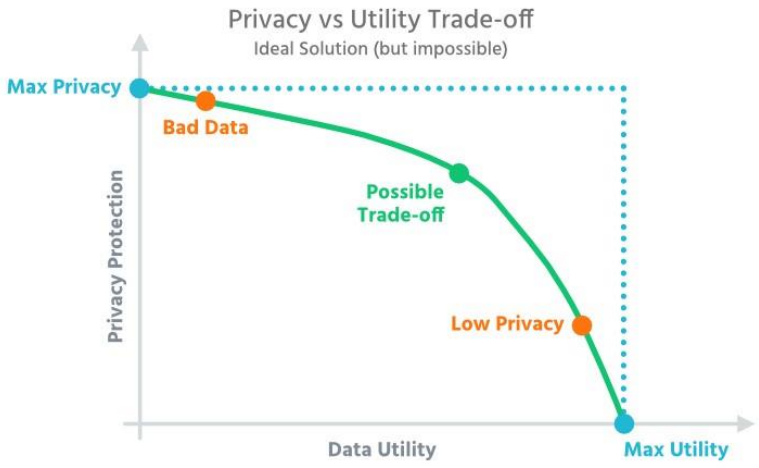

Figure 5:Trade-off between privacy and utility.
Illustration from https://aircloak.com/background/analytics-and-privacy/ power-of-anonymization/

For an overview of various anonymisation techniques and tools, we refer the reader to [19].

4.2 DIFFERENTIAL PRIVACY

Differential privacy is a privacy method that helps reveal useful information about a dataset without revealing any private information about an individual in the dataset. The method guarantees that even if an attacker knows all records in a dataset, s/he will not be able to link a specific record to an individual on the basis of the output of a differentially-private method. In other words, the outcome of the analysis is not (significantly) dependent on an individual's record being in the dataset. Hence, the privacy risk is essentially the same with or without the individual's participation in the dataset.

Differential privacy is achieved by adding random noise to the result, which can be done through various differentially-private mechanisms, such as the Laplace mechanism, the exponential mechanism and the randomised response mechanism.

Formally, differential privacy is defined as follows [20]. A randomised function K is ϵ-differentially-private if for all pairs of datasets D_1 and D_2 that differ on at most one element, and for every set S of outcomes,

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S]$$

The privacy budget ϵ measures the privacy loss of an individual from a single query. For multiple t queries that use independent randomisation mechanisms with privacy budget ϵ_i, the total privacy budget will be $\sum_{i=1}^{t} \epsilon_i$. For this reason, the number of differentially-

private analyses on a specific dataset is limited. The value of ϵ in practice is usually small, such as 0.01,0.1 or ln2. In general, the smaller ϵ, the stronger the privacy guarantee.

Below we briefly mention some differentially-private mechanisms. For more details, we refer to [20] and [21].

The Laplace mechanism is one of the commonly used differential-privacy mechanisms for numeric queries. The mechanism adds Laplace-distributed noise of magnitude depending on the privacy budget ϵ and the sensitivity of the query, that is, the maximum difference in the output that the query may take on a pair of databases that differ for only one record.

Randomised response is a technique that provides plausible deniability for individuals responding to sensitive surveys. It is widely used in statistical analysis for obtaining statistical information about a population without obtaining any information about the individuals in the population. The approach randomises the data in the following way: for answering a binary question concerning private information, a user flips a coin. If the outcome is “head”, the user answers “yes”. Otherwise, if the outcome is “tail”, the user answers truthfully (yes or no). This approach allows to compute an approximate answer for the true response rate without getting a true answer from all the users.

In Section 5.1 we will discuss how differential privacy is used in an ML context.

4.3 HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (HE) is a cryptographic method that allows computing on encrypted data, so that the decrypted output is identical to the output of the computation on the original unencrypted input. The method is typically used as follows:

1. the owner of the data encrypts them by means of a homomorphic function and shares the result with a third party in charge of performing a given computation;
2. the third party performs the computation on the encrypted data and returns the result, which is encrypted because the input data are encrypted;
3. the owner of the data decrypts the result, thereby obtaining the result of the computation on the original plain-text data.

During this process the third party does not have access to the unencrypted input or output.

Whilst powerful, it is worthwhile observing that HE has a number of limitations.

First of all, whilst there are various encryption schemes that allow performing different computations on encrypted data, they are generally limited to working with addition or multiplication over integers. For example, an additively encryption scheme allows to take two encrypted messages, m_1 and m_2, and produce the encrypted message $m_1 + m_2$.

Moreover, for security purposes, some noise is typically added to the input data during the encryption process, and the noise accumulates with the operations performed on encrypted data. For instance, for the encrypted message $m_1 + m_2$. the noise is the sum of the noise of m_1 and m_2. The noise growth is particularly severe under multiplication.

In order to get the expected result when decrypting, the noise must be kept below a certain threshold. This threshold affects the number of computations that can be performed on encrypted data.

In order to overcome the noise issue, Fully Homomorphic Encryption (FHE) has been proposed, which is an HE scheme that allows arbitrary operations on encrypted messages [22]. However, FHE has a significant computational cost, thus making it impractical for many applications.

More practical schemes that deal with noise have been developed, such as Somewhat Homomorphic Encryption (SHE) [23] and Leveled Homomorphic Encryption (LHE) [24]. In both schemes, the operations are limited to addition and multiplication on encrypted messages. However, a SHE scheme only supports one operation, while an LHE scheme requires that the number of computations is known in advance. ML algorithms using HE often resort to SHE or LHE.

In Section 5.2 we will discuss how homomorphic encryption can be used to enhance ML techniques.

4.4 MULTI-PARTY COMPUTATION

Multi-Party Computation (MPC) is a technology that allows multiple parties to compute a function on their private inputs without revealing them. The parties are independent and do not trust each other. The main idea is to allow to perform a computation on the private data while keeping the data secret. MPC guarantees that all participants learn nothing more than what they can learn from the output and their own input.

Below we briefly mention some techniques for secure MPC. For more details, we refer to [25].

Garbled circuits is a widely-used cryptographic protocol for two-party secure computation on Boolean functions (circuits). The steps of the protocol are as follows:

- the first party, Alice, encrypts (or garbles) the function (circuit) and sends it to the second party, Bob, together with her encrypted input;
- with the help of Alice, Bob encrypts his own input using oblivious transfer, i.e., Alice and Bob transfer some information such that the sender remains oblivious what information has been transferred;
- Bob evaluates the function using both encrypted inputs and gets the encrypted output;
- finally, the two parties communicate to learn the output.

Secret sharing is a method common to many MPC approaches. A (t, n)-secret sharing method divides the secret s into n shares and allocates a share to a participant. The secret *s* can be reconstructed by combining together *t* shares, while no information about *s* will be revealed when any t*-1* of the shares are combined. In other words, the secret is divided in such a way that any group of at least t participants can reconstruct the secret, while no group of less than t participants can.

Both MPC and homomorphic encryption are powerful privacy techniques, however they usually have high communication and computation costs. In Section 5.3 we will discuss MPC in connection with ML.

5 PRIVACY-PRESERVING MACHINE LEARNING TECHNIQUES

Machine learning techniques are widely used in a great many real-life applications, and data collection is the key element to enable such techniques. However, the increasing collection and usage of data for ML purposes pose a threat to our privacy. For example, an adversary can extract private information memorised in an ML model.

In order to reap the benefits of ML while protecting private information, various privacy-preserving ML techniques have been developed. In particular, the literature focuses significantly on privacy techniques such as differential privacy, homomorphic encryption, secure multi-party computation, federated learning and their combinations.

In the following, we will discuss the main privacy techniques used in combination with ML. For a general overview of privacy-preserving techniques and the challenges associated with them, we refer the reader to [26, 27].

It is worthwhile observing that anonymisation techniques can, in principle, be applied to any dataset in the ML process. In this way, we can protect the privacy of individuals and avoid or mitigate de-anonymisation attacks. By means of these techniques, it is also possible to generate a synthetic dataset, i.e., a new dataset that is not derived from the original one but preserves only some of its statistical traits. Moreover, anonymisation techniques allow the model creator to use existing, common frameworks instead of specific privacy-preserving ones. However, the effect of applying anonymisation techniques in a standalone fashion can impact the accuracy of the target ML model. Hence, in the following we discuss approaches explicitly investigated in connection with ML.

5.1 TECHNIQUES BASED ON DIFFERENTIAL PRIVACY

Differential privacy is an anonymisation technique to which we can resort in order to enhance ML to mitigate privacy concerns. Differential privacy can be used to generalise the ML process, aiming at hiding the effects of specific input data and providing differential privacy with respect to individuals, thereby giving a provable guarantee of privacy. This goal can be achieved in different ways, applying differential privacy principles to the training data or the resulting ML model.

In the following, we will group together and discuss differential privacy ML algorithms on the basis of the assets that they aim at protecting.

Training data. While differential privacy can in principle be applied to any dataset throughout an ML process, the literature focuses on protecting the training data. A

number of approaches have been proposed to train a model on *private* data, where the general idea is to add noise to the training data.

Nonetheless, the introduction of noise in the training set can destroy the utility of the model, hence due care needs to be exercised.

Differential privacy principles have been successfully applied to training data for ML algorithms based on the Stochastic Gradient Descent method (SGD), an iterative algorithm for incremental gradient updates aimed at minimising a loss function. The differentially-private SGD algorithm allows to train an ML model with provable privacy guarantees. The high-level steps are as follows. First, the algorithm computes the gradient for a random subset of the training data; then, it adjusts the gradient values; finally, it adds noise to the gradient with the objective of minimising the loss function. Some of the existing algorithms also compute the overall privacy loss throughout the training, which allows to evaluate the privacy of the resulting ML model. For more details on differentially-private SGD algorithms, we refer the reader to [28, 29, 30].

Another approach presented in the literature is the Private Aggregation of Teacher Ensembles framework (PATE), which trains an ML model with private data and provides strong privacy guarantees. The high-level steps of the framework are as follows:

1. the input data are partitioned into disjoint datasets;
2. the same ML algorithm is applied to each disjoint dataset, thereby producing as many ML models as datasets. A trained ML model is called a *teacher*;
3. when a new problem to solve is received, the outcomes of all teachers are aggregated (e.g., by majority) and noise is added to the aggregated outcome;
4. a new ML model (called *student*) is trained on the aggregated outcomes with added noise.

The output of the framework is the trained student model. Figure 6 provides the overview of the PATE framework. For more details on PATE, we refer the reader to [31, 32, 33].

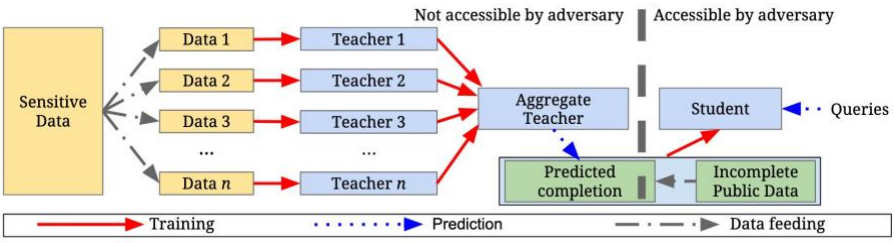

Figure 5: Overview of the PATE framework.

A drawback of applying differential privacy techniques to the training data is that the added noise can lead to a significant reduction in utility due to the fact that the process is typically iterative and requires to add noise at each iteration.

Training data in the ML model. The ML model implicitly contains information about the training data which also needs to be protected. In particular, here we want to prevent an

attacker from inferring whether or not a given individual's data was included in the training set, i.e., a membership inference attack. Differential privacy techniques, such as the Laplace mechanism, can be applied to introduce noise in the model to render attacks more difficult.

Model inversion attacks aim at inferring some attributes about a target individual from a given ML model which is available to the attacker. Differential privacy-preserving regression models focus on preventing such attacks. The high-level idea is to derive regression models that preserve privacy of an individual, which can be achieved by adding noise to the objective function. In other words, the privacy of the model is preserved by applying differential privacy on model outputs. For more details, we refer the reader to [34, 35].

Another approach to protecting the privacy of an individual in a model relies on generative adversarial network (GAN) for generating differentially-private datasets. The technique uses a generative model for training. First, we train the generative model on input data. Then, we apply noise (e.g., Gaussian) to make the generative model differentially private. Finally, we generate a synthetic privacy-preserving dataset using the generative model. Hence, both the model and the output are differentially private. For more details, we refer the reader to [36].

A drawback of applying differential privacy techniques to the ML model is that accuracy is compromised. However, some of the research mentioned above tried to take care of that.

5.2 TECHNIQUES BASED ON HOMOMORPHIC ENCRYPTION

Homomorphic encryption is one of the most used cryptographic techniques to achieve privacy in ML. It is used to train an ML model on encrypted data, thus obtaining a model that produces encrypted outcomes that only the owner of the input can decrypt. Similarly to differential privacy, homomorphic encryption provides a provable guarantee of privacy.

Homomorphic encryption is widely used in scenarios where a third party trains an ML model and provides services using that model, such as prediction or a result to user queries. This is known as Machine Learning as a Service (MLaaS). In this scenario, two types of data need to be protected:

1. the data used to train and validate an ML model;
2. the data contained in a query that the trained model is asked to solve.

Homomorphic encryption techniques help address this concern by training a model on encrypted data so that the output of that model is also encrypted, thereby preventing the service provider to access plain-text information.

In recent years, various studies have investigated how to protect privacy in this scenario, in particular adapting neural network to work on encrypted domains. CryptoDL is one such solution that runs a neural network algorithm on encrypted data [37]. A drawback of CryptoDL is that the training of ML model on encrypted data requires heavy computation, hence making the training process slow.

CryptoNet is another solution that combines homomorphic encryption with neural network. In particular, CryptoNet assumes that an ML model is already available and converts it to a model that takes as input encrypted data and provides as output encrypted predictions [38]. Note that CryptoNet does not support training of the model on the encrypted data.

5.3 TECHNIQUES BASED ON MULTI-PARTY COMPUTATION

As we discussed above, MPC provides techniques that allow multiple parties to compute a function without revealing their secret inputs. In an ML context these techniques can be utilised in either training an ML model on multiple secret datasets or computing the result of an ML model on multiple secret inputs. The former aims at achieving raw dataset, features datasets privacy and identity privacy, while the latter targets the input privacy goal.

For the first approach, the idea is that each participant can share its dataset using the secret sharing technique and thus the desired ML model $\mathcal{M}_\theta$ could be computed locally. In contrast to the standard ML processes, here each party contributing its dataset is a data contributor, a model creator since it builds the model locally, a model owner and model consumer since the goal of each participant is to use the model for its own purpose. Since MPC operates on values of the finite set Z_p (i.e., the integers modulo p), the values in each participant's dataset need first to be mapped to elements in Z_p. For instance, if $Z_p = Z_{199}$ the float 1.32 could be mapped to 132. Next, in order for the participants to train and build the ML model $\mathcal{M}_\theta$, one first needs to translate the ML algorithm $\mathcal{A}$ to an arithmetic or Boolean circuit (depending on the ML task), and after that the training and validation phase can take place by computing on the shared encrypted data. Once the final model has been produced, each participant can query it locally with its own data. For an example of an implementation of this approach, we refer to [39].

For the second approach, it is assumed that a model $\mathcal{M}_\theta$ is known to the participants. The model may be owned by them or provided by some other party. As a first step, $\mathcal{M}_\theta$ needs to be translated into a Boolean or arithmetic circuit (depending on the ML task). Observe that this is different from what was discussed in the first approach, where the ML algorithm A was translated into a circuit. Once this is done, the participants can use the secret sharing technique but now for sharing their inputs for the model $\mathcal{M}_\theta$ and compute the joint result. Finally, the requirement that the inputs need to first be mapped to elements from the Z_p applies here as well.

MPC allows ML algorithms to operate on fully encrypted datasets or fully encrypted inputs. However, this comes with certain limitations. The requirement of working on fixed-point arithmetic from Z_p is not always desirable in many ML tasks, while the size of the input and the datasets impacts the circuit representation of an ML algorithm or model, which can be significantly large, whose storage requires large amounts of memory. Finally, MPC protocols require continues data transfer between the involved participants and continuous online availability, which introduces additional performance overhead and scalability limitations.

5.4 FEDERATED LEARNING

The accuracy of most ML models vastly depends on the size of the data contributor's data. Since model owners are usually big corporations, the data collection part usually takes place in cloud server(s) owned by those corporations, allowing them to have full control over the data contributor's datasets. The latter not only discourages individuals sharing their personal information and thus putting barriers on producing highly accurate models, but it also creates centralised data banks attracting more adversaries and increasing the risk of compromising the collected data.

Federated learning allows ML processes to be decentralised limiting the information exposed from the contributor's datasets and thus controlling the risk of compromising the dataset's and identity privacy. The general idea of federated learning is that a central ML model $\mathcal{M}_\theta$ owned by some central authority (e.g., a company) can be further trained on new private datasets from data contributors, by having each contributor performing the training locally with its own dataset and then updating the central model $\mathcal{M}_\theta$ (i.e., update the model's parameter θ).

In particular, federated learning works as follows (1) the central model $\mathcal{M}_\theta$ is distributed among a number of n participants (i.e., the data contributors); (2) each participant updates locally the model $\mathcal{M}_\theta$ by training it on its own local dataset Z_l, producing a new local parameter θ_l; (3) each participant sends its update θ_l to the central authority; (4) the central authority aggregates the local parameters of each participant, producing a new parameter θ' which is used to update the central model. This process can be further continued until the central model has been trained enough.

Federated learning can be seen as a flexible approach for achieving privacy of raw datasets, feature datasets and identity privacy, since it lifts the requirement of collecting vast amounts of information from the data contributors and then train on them in a centralised manner. However, communication in federated learning networks could create performance overhead, which is not present in centralised ML approaches, and put in conjunction with the availability of devices in the network, scalability also becomes a concern. Finally, in many cases of federated learning, each local parameter θ_l may be highly correlated with its corresponding dataset Z_l, allowing one to infer information about Z_l by processing θ_l.

5.5 COMBINING PRIVACY-PRESERVING TECHNIQUES

There is no silver-bullet technique when it comes to achieving privacy in ML. The privacy degree offered by the techniques we presented varies a lot depending on many factors such as the ML algorithm used, the adversary's capabilities and resources, and counting. Achieving higher degrees of privacy may require the combination of multiple privacy-preserving ML techniques. In the following, we first give a high-level overview of some key properties of these techniques, and we then discuss their combination.

Each privacy-preserving ML technique targets different privacy goals, hardening an ML infrastructure and limiting the vulnerability surface exposed to an adversary. Table 1

summarises the main characteristics of the privacy-preserving ML techniques discussed above, such as the privacy goals, their strengths and weaknesses.

Technique	Privacy Goal					Strengths	Weaknesses
	Identity	Raw Dataset	Feature Datasets	Model	Input		
Differential Privacy	√	√	√	√		Provable guarantee of privacy	Compromising accuracy of an ML model
Homomorphic Encryption	√	√	√		√	Computing on encrypted data, provable guarantee of privacy	High computation costs, works on numerical data
Multi-Party Computation	√	√	√		√	Computing on encrypted data	High communication costs, high availability, high computation costs, works on fixed-point arithmetic
Federated Learning	√	√	√	√		Decentralised training	High communication costs, high availability

Table 1: Overview of privacy-preserving techniques.

Differential privacy addresses the privacy of different datasets involved in an ML process, offering a provable privacy guarantee. However, it can be challenging to strike the right balance between utility and privacy of the anonymised data, which are inversely related.

Homomorphic encryption allows to work on encrypted data, thereby preserving the utility of the original datasets, but its domain is quite limited, and scalability can be an issue.

MPC allows multiple parties to compute a function by sharing their encrypted data. This can be utilised (1) for training a common ML model without requiring the parties to reveal private information, and thus achieving certain degrees of identity, raw dataset and feature datasets privacy; and (2) for computing the result of an ML model by having the parties sharing their encrypted inputs, achieving input privacy. However, similarly to HE, the flexibility of computing on encrypted data comes with certain communication costs, while the requirement of transforming the model or an ML algorithm may create additional performance overhead. Also, the requirement of constant availability of the computing parties may introduce scalability issues, while the requirement of operating on fixed-point arithmetic limits the application space of ML.

Federated learning lifts the restriction of centralised model training by allowing models to be trained locally, without requiring the sharing of the data contributor's datasets, thus aiming for identity, raw dataset and feature datasets privacy. However, the network of

devices for training the model requires high communication costs and availability of devices while training.

Recent literature has proposed combinations of multiple privacy-preserving ML techniques, aiming for higher privacy guarantees.

In [40], the authors propose a novel protocol called “secure aggregation”, which enhances the privacy offered by federated learning by combining it with MPC. In their setting, a central ML model hosted on a server is trained within a network of mobile devices, where each device updates the central model by calculating its local parameter θ_l. The secure aggregation protocol utilises MPC techniques so that the server learns only the aggregation of the local parameters instead of each parameter individually. This approach deals with the problematic case of the local parameters being highly correlated with the local datasets used to train the local models on the devices. Finally, secure aggregation is designed to be efficient and robust against availability issues arising when some of the mobile devices leave the network.

Along the same line, [41] has combined federated learning and differential privacy in the context of model training within a network of mobile devices. This work is concerned with malicious mobile clients trying to perform model inversion attacks once they have downloaded the central model. To address this situation, whenever the central server aggregates the local parameters from the mobile clients, it adds a certain amount of noise before updating the model and making it available for training again.

Finally, differential privacy and federated learning have been combined in [42]. In this work, once the central model is distributed to the clients, each client trains the model locally, and before the client sends its updates to the server, it first applies a local differential private mechanism on it.

6 PRIVACY-PRESERVING MACHINE LEARNING TOOLS

While there is still a gap between theoretical advancements and their applicability to real-world cases, there are some open-source projects and tools specific to privacy-preserving ML. In the following section, we discuss some of these tools.

6.1 TENSORFLOW PRIVACY

TensorFlow Privacy (TFP) is a Python library for training and building ML models with differential privacy, developed by Google. The library builds on Google's TensorFlow (https://www.tensorflow.org/), which is another open-source platform used for traditional ML training without privacy considerations.

One of the key privacy-preserving ML techniques offered by the library is training an ML model by applying differential private SDG. On top of that, using TFP, one can calculate the privacy guarantees offered by the chosen differential private mechanism, which can then be used to (1) compare ML models in terms of privacy and (2) account about the drop of utility when choosing one model over another.

To get started with the TFP library, one could visit the GitHub repository https://github.com/tensorflow/privacy.

6.2 PYSYFT

PySyft is an open-source library (written in Python) for secure and private ML. It is a part of the OpenMined project that focuses on developing libraries and tools for privacy-preserving AI. The library supports different privacy-preserving techniques such as differential privacy, HE, MPC and federated learning. Moreover, PySyft extends the major deep learning frameworks, such as PyTorch, TensorFlow and Keras. Figure 6 presents the high-level architecture of PySyft and its integration with other frameworks.

The library is under constant development and builds on published research in the field of privacy-preserving ML.

For more details about the library, related publications and how to use, we refer to the official website of OpenMined and the GitHub page of PySyft:
https://www.openmined.org/ and https://github.com/OpenMined/PySyft.

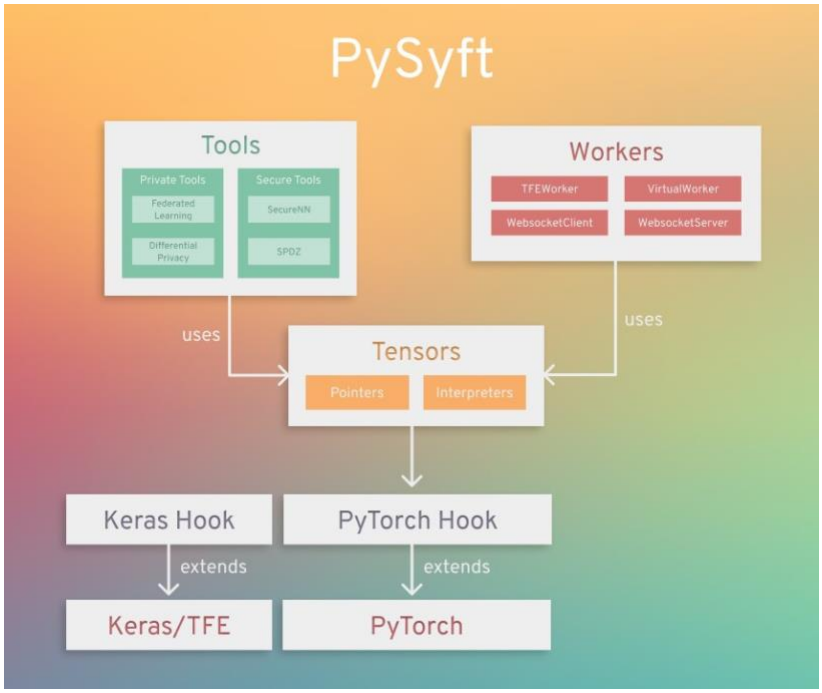

Figure 6: High-level architecture. Illustration from https://github.com/ OpenMined/PySyft.

6.3 ML PRIVACY METER

Assessing the robustness of an ML model against certain attacks is as important as integrating privacy mechanisms in the ML process used to build the model.

ML Privacy Meter is a Python library, built on top of Google's TensorFlow, which allows one to quantify the privacy risks within an ML model. In particular, the tool can be used to generate membership inference attacks under both whitebox and blackbox adversary models. The tool can then calculate the privacy risk scores under the chosen adversary model. The risk scores can be understood as an accuracy measure of such attacks on the model of interest. Finally, the tool is also able to visualise the results and produce privacy reports.

For more information about the tool, we refer to the GitHub repository https://github.com/privacytrustlab/ml_privacy_meter, while one could also read the tool's scientific paper [43].

6.4 CRYPTFLOW

CrypTFlow [44] is a system which provides a solution for securely querying ML models by utilising techniques from the fields of programming languages and MPC.

In particular, inside the core of CrypTFlow there is a compiler which transforms code from TensorFlow to an MPC protocol. Converting code from the well-known ML framework

TensorFlow automatically eases the task of integrating MPC into ML tasks and makes CrypTFlow an appealing tool for ML practitioners.

CrypTFlow has been developed by Microsoft Research and it is open-source. For further information, we refer to the public repository at:
https://github.com/mpc-msri/EzPC//github.com/mpc-msri/EzPC.

6.5 CRYPTEN

CrypTen is a research-based privacy-preserving ML framework. The tool is built on the open-source ML framework PyTorch. Currently, the framework supports MPC, with the possible extension of supporting HE in the future. Figure 7 presents the high-level architecture of CrypTen.

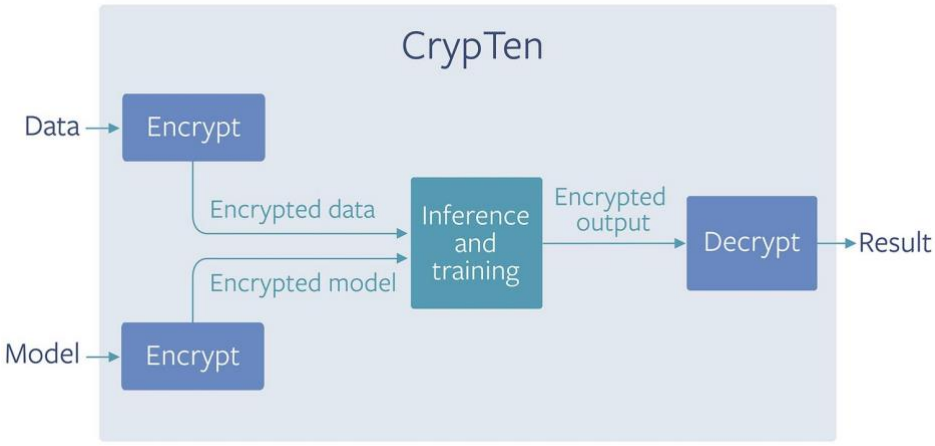

Figure 7: High-level architecture. Illustration from https://crypten.ai/

The framework is under development and builds on published research in the field of privacy-preserving ML. For more details about the framework, we refer to the official website and the GitHub page of CrypTen:
https://crypten.ai/ and https://github.com/facebookresearch/CrypTen

7 CONCLUSION

Machine learning is a technology that allows to process large amount of data to produce predictive models. The application of ML in everyday software has increased significantly in recent years, raising more and more concerns about its usage on sensitive information, such as health-care or financial data. In fact, both the data collected to train an ML model and the model itself can be subject to attack and disclose private information, exposing companies to critical incidents, including reputational damage and fines from regulators.

Privacy-preserving ML allows to process data limiting unwanted disclosure of specific information. It strengthens ML with privacy-preserving techniques that provide provable guarantees of privacy, such as differential privacy, homomorphic encryption and multi-party computation. These techniques can be exploited to protect information embedded in any asset of an ML framework, such as the training or validation data and the ML model itself. However, before applying any privacy-preserving ML technique, the specific scenario must be analysed, as there is no silver-bullet solution.

In this paper, we have presented an introductory guide to privacy-preserving ML. We discussed a wide range of available techniques and some tools, and for each technique we highlighted the range of protection, so as to assist in choosing the suitable technique for a given use case.

It is worthwhile observing that we have not discussed risk analysis frameworks. However, it is a good practice to assess the privacy loss of a privacy-preserving ML model before disclosing it.

REFERENCES

[1] C. F. Kerry, “Protecting privacy in an ai-driven world,” 2020. Last accessed 24 August 2020, https://www.brookings.edu/research/ protecting-privacy-in-an-ai-driven-world/.

[2] European Commission, “White paper on artificial intelligence–a european approach to excellence and trust,” 2020.

[3] T. Kubota, “Deep learning algorithm does as well as dermatologists in identifying skin cancer,” 2017. Last accessed 14 August 2020, https://news.stanford.edu/2017/01/25/ artificial-intelligence-used-identify-skin-cancer/.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 3–18, IEEE Computer Society, 2017.

[5] L. Sweeney, “k-anonymity: A model for protecting privacy,” *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[6] C. M. Bishop, Pattern recognition and machine learning. springer, 2006.

[7] A. Pyrgelis, C. Troncoso, and E. D. Cristofaro, “Knock knock, who’s there? membership inference on aggregate location data,” in 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018, The Internet Society, 2018.

[8] 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 *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, The Internet Society, 2019.

[9] J. Qian, X. Li, C. Zhang, and L. Chen, “De-anonymizing social networks and inferring private attributes using knowledge graphs,” in *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, pp. 1–9, IEEE, 2016.

[10] S. Gambs, M. Killijian, and M. N. del Prado Cortez, “De-anonymization attack on geolocated data,” *J. Comput. Syst. Sci.*, vol. 80, no. 8, pp. 1597– 1614, 2014.

[11] S. Gambs, M. Killijian, and M. N. del Prado Cortez, “De-anonymization attack on geolocated data,” in 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013 / 11th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA-13 / 12th IEEE International Conference on Ubiquitous Computing and Communications, IUCC-2013, Melbourne, Australia, July 16-18, 2013, pp. 789–797, IEEE Computer Society, 2013.

[12] J. Feng and A. K. Jain, “Fingerprint reconstruction: From minutiae to phase,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 209– 223, 2011.

[13] M. Al-Rubaie and J. M. Chang, “Reconstruction attacks against mobilebased continuous authentication systems in the cloud,” *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2648–2663, 2016.

[14] T. Takemura, N. Yanai, and T. Fujiwara, “Model extraction attacks against recurrent neural networks,” *CoRR*, vol. abs/2002.00123, 2020.

[15] R. N. Reith, T. Schneider, and O. Tkachenko, “Efficiently stealing your machine learning models,” in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, WPES@CCS 2019, London, UK, November 11, 2019* (L. Cavallaro, J. Kinder, and J. Domingo-Ferrer, eds.), pp. 198– 210, ACM, 2019.

[16] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015* (I. Ray, N. Li, and C. Kruegel, eds.), pp. 1322–1333, ACM, 2015.

[17] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, “The secret revealer: Generative model-inversion attacks against deep neural networks,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 250–258, IEEE, 2020.

[18] M. Fredrikson, E. Lantz, S. Jha, S. M. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing,” in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014* (K. Fu and J. Jung, eds.), pp. 17– 32, USENIX Association, 2014.

[19] Z. Aslanyan, “White paper on anonymisation–a practical guide,” Alexandra Institute, 2020.

[20] C. Dwork, “Differential privacy,” in Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 1014, 2006, Proceedings, Part II (M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds.), vol. 4052 of Lecture Notes in Computer Science, pp. 1–12, Springer, 2006.

[21] C. Dwork, “A firm foundation for private data analysis,” *Commun. ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[22] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*, vol. 20. Stanford University, 2009.

[23] N. P. Smart and F. Vercauteren, “Fully homomorphic encryption with relatively small key and ciphertext sizes,” in *International Workshop on Public Key Cryptography*, pp. 420–443, Springer, 2010.

[24] Z. Brakerski and V. Vaikuntanathan, “Efficient fully homomorphic encryption from (standard) lwe,” *SIAM Journal on Computing*, vol. 43, no. 2, pp. 831–871, 2014.

[25] D. Evans, V. Kolesnikov, and M. Rosulek, “A pragmatic introduction to secure multi-party computation,” *Foundations and Trends ® in Privacy and Security*, vol. 2, no. 2-3, 2017.

[26] M. Al-Rubaie and J. M. Chang, “Privacy-preserving machine learning: Threats and solutions,” *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019.

[27] E. D. Cristofaro, “An overview of privacy in machine learning,” *CoRR*, vol. abs/2005.08679, 2020.

[28] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

[29] S. Song, K. Chaudhuri, and A. D. Sarwate, “Stochastic gradient descent with differentially private updates,” in *2013 IEEE Global Conference on Signal and Information Processing*, pp. 245–248, IEEE, 2013.

[30] R. Bassily, A. Smith, and A. Thakurta, “Private empirical risk minimization: Efficient algorithms and tight error bounds,” in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473, IEEE, 2014.

[31] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” *arXiv preprint arXiv:1610.05755*, 2016.

[32] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson, “Scalable private learning with pate,” *arXiv preprint arXiv:1802.08908*, 2018.

[33] N. Papernot and I. Goodfellow, “Privacy and machine learning: two unexpected allies?,” 2018. Last accessed 02 July 2020, http://www.cleverhans.io/privacy/2018/04/29/ privacy-and-machine-learning.html.

[34] Y. Wang, C. Si, and X. Wu, “Regression model fitting under differential privacy and model inversion attack,” in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[35] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, “Functional mechanism: regression analysis under differential privacy,” *arXiv preprint arXiv:1208.0219*, 2012.

[36] A. Triastcyn and B. Faltings, “Generating differentially private datasets using gans,” *CoRR*, vol. abs/1803.03148, 2018.

[37] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, “Privacypreserving machine learning as a service,” *PoPETs*, vol. 2018, no. 3, pp. 123–142, 2018.

[38] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (M. Balcan and K. Q. Weinberger, eds.), vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 201–210, JMLR.org, 2016.

[39] J. Miller, “Multi-party computation on machine learning”, https://blog.trailofbits.com/2019/10/04/multi-party-computation-on-machine-learning/

[40] 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017* (B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds.), pp. 1175–1191, ACM, 2017.

[41] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private language models without losing accuracy,” *CoRR*, vol. abs/1710.06963, 2017.

[42] S. Truex, L. Liu, K. H. Chow, M. E. Gursoy, and W. Wei, “Ldp-fed: federated learning with local differential privacy,” in Proceedings of the 3rd International Workshop on Edge

Systems, Analytics and Networking, EdgeSys@EuroSys 2020, Heraklion, Greece, April 27, 2020 (A. Y. Ding and R. Mortier, eds.), pp. 61–66, ACM, 2020.

[43] S. K. Murakonda and R. Shokri, “ML privacy meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning,” *CoRR*, vol. abs/2007.09339, 2020.

[44] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, “Cryptflow: Secure tensorflow inference,” in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pp. 336–353, IEEE, 2020.

THE ALEXANDRA INSTITUTE
IT CITY OF KATRINEBJERG
Aabogade 34 ▪ DK-8200 Aarhus N
+45 70 27 70 12

UNIVATE
Njalsgade 76, 3rd floor ▪ DK-2300 Copenhagen S
+45 70 27 70 91

The Alexandra Institute helps private and public organisations develop innovative solutions, products and services based on state-of-the art IT research. Our mission is to enable Danish companies realise the potential of new technology.