A review of privacy-preserving techniques for deep learning

Amine Boulemtafes, Abdelouahid Derhab, Yacine Challal

A Review of Privacy-Preserving Techniques for Deep Learning

Amine Boulemtafes[1,*], Abdelouahid Derhab[2], and Yacine Challal[3]

[1] Division Sécurité Informatique, Centre de Recherche sur l'Information Scientifique et Technique, Algiers, Algeria, and also Département Informatique, Faculté des Sciences exactes, Université de Bejaia, 06000 Bejaia, Algeria (email : aboulemtafes@cerist.dz)
[2] Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudia Arabia (email : abderhab@ksu.edu.sa)
[3] Laboratoire de Méthodes de Conception des Systèmes, Ecole Nationale Supérieure d'Informatique, and also Division Sécurité Informatique, Centre de Recherche sur l'Information Scientifique et Technique, Algiers, Algeria (email : y_challal@esi.dz)

* Corresponding author

Abstract—Deep learning is one of the advanced approaches of machine learning, and has attracted a growing attention in the recent years. It is used nowadays in different domains and applications such as pattern recognition, medical prediction, and speech recognition. Differently from traditional learning algorithms, deep learning can overcome the dependency on hand-designed features. Deep learning experience is particularly improved by leveraging powerful infrastructures such as clouds and adopting collaborative learning for model training. However, this comes at the expense of privacy, especially when sensitive data are processed during the training and the prediction phases, as well as when training model is shared. In this paper, we provide a review of the existing privacy-preserving deep learning techniques, and propose a novel multi-level taxonomy, which categorizes the current state-of-the-art privacy-preserving deep learning techniques on the basis of privacy-preserving tasks at the top level, and key technological concepts at the base level. This survey further summarizes evaluation results of the reviewed solutions with respect to defined performance metrics. In addition, it derives a set of learned lessons from each privacy-preserving task. Finally, it highlights open research challenges and provides some recommendations as future research directions.

Keywords—Deep Learning, Deep Neural Network, Privacy, Privacy preserving, Sensitive data, Taxonomy

1. Introduction

Deep learning, one of the most advanced approaches of machine learning, has attracted a lot of attention in research as it provides the ability to overcome the dependency on hand-designed features that is faced by traditional learning algorithms. Deep learning or usually Deep Neural Networks (DNNs) typically comprises two phases: a training step to optimize the accuracy of the model and an inference phase where model is used for analysis as classification or prediction (see the next section for more details). Nowadays, deep learning is being used in different domains including big data analytics and different applications such as pattern recognition, speech recognition, computer vision, natural language processing, intrusion detection, and medical predictions [1].

Deep learning experience is particularly improved by leveraging powerful infrastructures such as clouds and adopting collaborative learning for model training. As user devices are limited in terms of resources, the solution is to offload resource-demanding operations to an external infrastructure with high-power computation and massive storage such as a cloud. On the other hand, collaborative learning is applied on large and diversified datasets that are originated from different sources, e.g., medical organizations or patients, which results in achieving better learning accuracy. However, privacy concerns, which are related to sensitive data for both model training and its use for inference, are raised. Such concerns include identification of individuals, unauthorized commercial sharing of confidential information, illegitimate use of private data, and the disclosure of sensitive data or inferred private information like disease risks from health records. Additionally, other privacy concerns related to sharing a deep learning model need to be considered. In fact, it has been shown that if training private data are not well protected, they are subject to leakage through model parameters or predictions [2-8].

To tackle the above mentioned concerns, various approaches have been proposed. This survey aims to present a state-of-the-art of recent deep learning techniques and approaches addressing potential privacy concerns, and particularly related to input data which is the focus of this work, along with potential interesting directions and learned lessons.

In the literature, there are two related surveys [9, 10] that deal with privacy-preserving in deep learning. However, these surveys are short, and no discussion regarding research challenges and practical future

directions is provided. Zhang et al. [9] reviewed privacy-preserving in deep learning by focusing on collaborative learning, and considering two phases: training and using. As for training, authors considered both direct collaborative learning where local data is directly uploaded to the central server, and indirect learning where local models updates are uploaded to the central server for aggregation. Chang et Li [10] focused on privacy issues during the training and the prediction phases of the neural network learning, along with their corresponding threats and countermeasures including attacks on trained models. Noting that both centralized and distributed training, equivalent to direct and indirect training presented in [9], were considered.

Differently from [9] and [10], the main contributions of this paper are the following points:

- Reviews more than 45 recent solutions papers, and more than 40 different privacy-preserving deep learning techniques.
- Proposes a multi-level taxonomy that classifies the privacy-preserving deep learning techniques with respect to four levels:
  - The first level distinguishes between three privacy-preserving (PP) tasks, namely: (1) PP Model learning, (2) PP Analysis, and (3) PP model releasing, as will be described in Sections 3, 4, and 5.
  - The second level distinguishes between collaborative and individual model learning.
  - The third level differentiates between server-based and server-assisted solutions of the PP model learning and PP analysis.
  - The fourth level classifies the privacy-preserving techniques with respect to the used key technological concepts.
- Summarizes evaluation results of the reviewed solutions with respect to performance metrics.
- Discusses and outlines a number of learned lessons of each privacy-preserving task.
- Highlights open research challenges and provides some recommendations for future research.

The rest of this paper is organized as follows: Section 2 gives background on deep learning, privacy concerns, main technologies, and performance metrics. It also describes the proposed multi-level classification for privacy-preserving techniques. Sections 3, 4, and 5, each of which reviews the existing solutions of one privacy-preserving task. The solutions are discussed within their respective tasks along with a summary of evaluation results. Each task is concluded with a number of learned lessons. Section 6 highlights open challenges and provides some recommendations for future research. Finally, Section 7 concludes the paper.

2. Privacy-preservation in deep learning

2.1 Background

The deep learning architectures, as shown in Figure 1, are built as multi-layer neural networks that are composed of different layers: input, output, and one or multiple hidden layers that connect the input and the output layer. In a typical neural network, layers are connected through neurons, which mathematically transform the data. They compute the total input, i.e., weighted average of its inputs, add a bias signal, and apply a nonlinear activation function like sigmoid, rectifier or hyperbolic tangent to produce the neuron's output [3, 11].

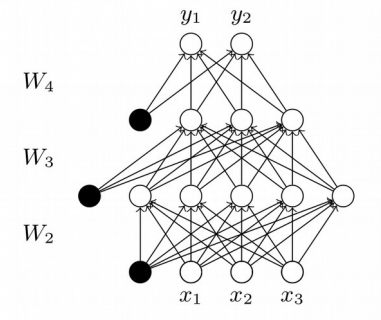

Fig. 1. A typical neural network with two hidden layers. x_i are the inputs, y_i are the outputs, *black circles* are biases and W_i are the weight vectors for computing the weighted averages of inputs [3]

Deep learning consists of the following two phases [2, 6, 12]:

- Training phase, in which each layer of data is assigned initially some random weights and the deep learning classifier caries out a forward pass through the data to predict the class labels and scores. The class scores are compared against the actual labels and an error is computed, i.e., loss function. This error is then back propagated through the network and weights are updated accordingly. The weights are updated after each sample or after a batch of samples.
- Inference phase: a trained model is used to infer/predict testing and real-world data by performing a similar forward pass as the training phase. Since the goal is not to learn the model, the inference phase does not include a back-propagation step to compute the error and update the weights.

In the literature, we can find different deep learning architectures such as:

- *MLP (Multi-Layer Perceptron)*: is a feed-forward neural network with many hidden layers (multi-layer). Hidden layers in MLP are fully connected, i.e., each node in each layer is connected to every node of the following layer with a certain weight [13, 14].
- *CNN (Convolutional Neural Network)*: The neuro-biological model of the visual cortex largely inspired the CNN concept. In this network, the input is convolved using several small filters then sub-sampled repeatedly until high level features are extracted. After the last sub-sampling layer, several fully-connected layers are used to achieve final classification [15].
- *RBM-based (Restricted Boltzmann Machines) techniques particularly DBN (Deep Belief Network)*: are a variant of Boltzmann machines (BMs) where BMs can be considered as Neural networks with bidirectionally connected stochastic processing units. DBN is a composition of RBMs connected via hidden layers. DBN initialization is performed using efficient layer-by-layer greedy learning strategy then fine-tuned based on target outputs [1, 15].
- *DAE (Deep Auto Encoder):* An Autoencoder is a Neural network that has the same number of input and output nodes, and designed to recreate the input vector instead of assigning a class label to it. Under high dimensional input data, an Autoencoder with a single hidden layer may not be enough for representing all data. To deal with this issue, a DAE architecture is created by stacking many Autoencoders on top of each other and is often pretrained with DBN. DAE, considered as a non-linear feature extraction method, is usually used for dimensionality reduction and efficient encoding learning. [13, 15]
- *RNN (Recurrent Neural Network):* It is designed to work with sequential information. RNN is an important neural network for cases where the output depends on previous computations, like speech, text or DNA sequences analysis. In RNN, which has memory to store previous information, the output of new data is provided by exploiting the current and the recent past inputs. As for applications with longtime lags of unknown sizes between important events, LSTM (Long Short-Term Memory), a variant of RNN, is particularly suitable. This variant was proposed to address the problem triggered by long input sequences called vanishing gradient problem, and which prevents the parameters of the models to be updated at a certain time during training [14, 15].

More information regarding deep learning, its basics, algorithms, architectures, and applications can be found in [1], [3], [13], [14], [15], and [16].

2.2 Privacy concerns in deep learning

The use of deep learning raises some privacy concerns especially (1) when a powerful infrastructure such as a cloud is involved, and (2) when collaborative model is used. The privacy concerns are particularly related to sensitive input data either during training or inference and to the sharing of the trained model with others.

- The use of a powerful remote server or a cloud can be helpful at improving the back-propagation efficiency [2, 17]. However, privacy concerns can occur at both sides, i.e., the users and the server. In [6], it is mentioned that neither the server should obtain information regarding users data, nor information about training model should be revealed through inferences.
- It has been reported in [18] and [19] that it is possible to approximately reconstruct a part of the training data by only observing the predictions, such as recovering images from a facial recognition system through model inversion attack. The access to model parameters by an adversary with full

knowledge of the training mechanism may also lead to privacy risks [18], which shifts the problem from data privacy-preserving to model privacy-preserving.

- In the case of collaborative learning [3], the leakage of sensitive data among participants should also be considered due to information sharing and potential interaction among users [9].

Moreover, the leakage of sensitive data during transmission between the users and the external infrastructure particularly over Internet should also be considered [8].

2.3 Design and evaluation of privacy-preserving techniques

In this section, we briefly introduce the different aspects that are considered in the design and evaluation of the privacy-preserving techniques, including main technologies, target applications, deep learning models, adversary models, and performance metrics.

Different technologies are being used in order to preserve privacy. Differential privacy mechanisms [71, 72, 4] and homomorphic encryption [73, 37] which is generally coupled with polynomial approximation [37] are from the top of the list. Model splitting [8], partial parameters sharing [21], mimic learning [35] and other technologies are also interesting for enabling privacy as will be described later.

As for deep learning architectures, a number of models are proposed in the literature to evaluate privacy-preserving deep learning techniques. Multi-layer perceptron MLP [3, 13, 14] and Convolutional neural network CNN [3, 15] are the most widely used in experimentation, followed by the different variants of deep Auto Encoder (like Stacked Auto Encoder [2] or Tensor Auto Encoder [30] models). Combinations of models are also possible like CNN with Deep Belief Network to produce Convolutional Deep Belief Network (CDBN) [34], and CNN with Long Short-Term Memory (LSTM) to produce LSTM-CNN [20].

Privacy-preserving deep learning techniques are also used in various applications, and evaluated with different datasets. Image recognition like gender classification are popular. They are used with different datasets like MNIST [3] (Handwritten digits), CIFAR-10 [2] (Classified color images), or SVHN [3] (Street View House Numbers). Popular applications also include e-health like activity recognition over WISDM [32] or HAR [22] datasets. Other applications and related datasets include for instance, income prediction using ADULT [4] dataset, climate prediction applications using Climate [5] dataset, or traffic flow prediction with PeMS [30] dataset.

Performance metrics

To assess the merit of the privacy preserving deep learning techniques, we consider in this survey the following performance metrics:

(a) Effectiveness: It is defined in general as the degree to which the desired objectives/results are achieved. In our context, it represents the capability of the deep learning model to resolve the classification or prediction tasks. Accuracy is one of the major metrics that are used to measure effectiveness. The objective is to produce high accuracy in case privacy is involved or not, or make a trade-of between effectiveness and privacy [6,20,21].

(b) Efficiency: It generally measures the required time or overhead to perform a specific task. It may include for example running time to get an inference [6], training time [2] or network traffic [22].

(c) Privacy: It is evaluated theoretically or formally through properties, or empirically through measures in order to analyze or prove privacy preservation.

Adversary models

Honest-but-curious (HbC) adversary model is considered as a standard security model [12], and is the mostly assumed when evaluating privacy-preserving techniques. HbC is also called semi-honest and known as passive adversary model. It considers that any assumed semi-honest entity (cloud, data owners, ...) follows honestly the security protocol without performing malicious actions towards protocol or participants, but it could try to learn or infer sensitive information from private data, potentially colluding with some participants [3, 20, 23]. However, some reviewed works also adopted some scenarios of active adversary model [29, 56] where an adversary could deviate from the protocol in an arbitrary way. It is worth noting that in case of some scenarios, adversaries may have additional capabilities and knowledge than others.

2.4 Taxonomy of privacy-preserving deep learning techniques

Privacy risks might arise during the different phases of the deep learning process, i.e., (1) training a model, (2) inference through a model, and (3) releasing a model.

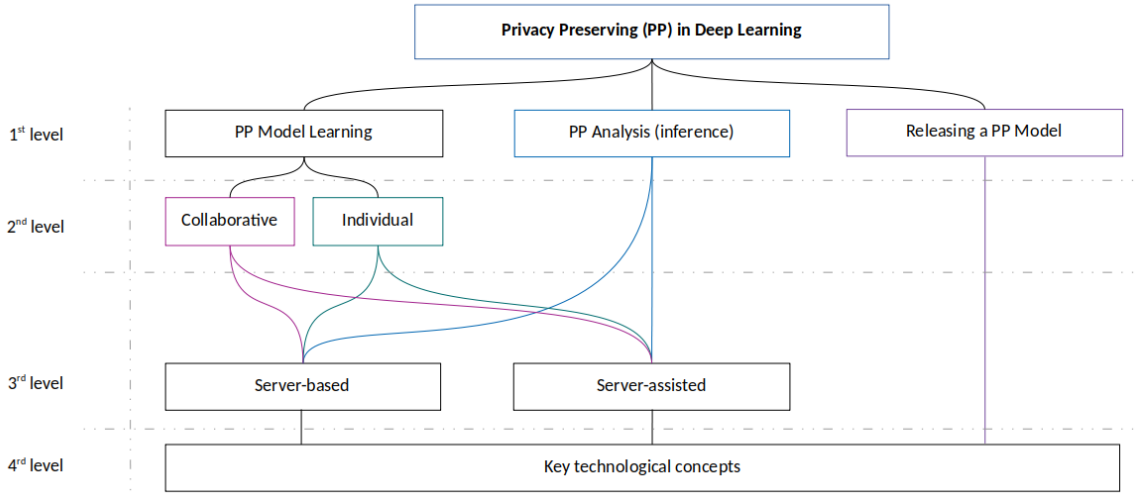

Fig. 2. Privacy-preserving techniques in Deep learning: Taxonomy overview

Based on the above mentioned phases, we propose a multi-level taxonomy, as shown in Figure 2, which classifies the state of the art of privacy-preserving deep learning techniques with respect to the following levels (see Figures 8, 10 and 13):

- The 1st level categorizes the reviewed works as : (1) *privacy-preserving learning* (see Section 3), (2) *privacy-preserving inference* (see Section 4), and (3) *releasing a privacy-preserving model* (see Section 5).
- The 2nd level, is built under the privacy-preserving model learning class, and distinguishes between (1) *collaborative learning* where multiple participants are involved and jointly take part in the deep learning process, and (2) *individual learning* where a single participant is involved.
- The 3rd level, is considered in both privacy-preserving model learning and privacy-preserving inference, and differentiates between (1) *server-based* where the inference and the model learning processes are performed exclusively on an external-to-participants infrastructure such as a cloud or remote server, and (2) *server-assisted* where the inference and the model learning are performed cooperatively between participants and the external infrastructure.
- The 4th and last level represents the key concepts used for designing different techniques such as encryption or model splitting as it will be detailed later.

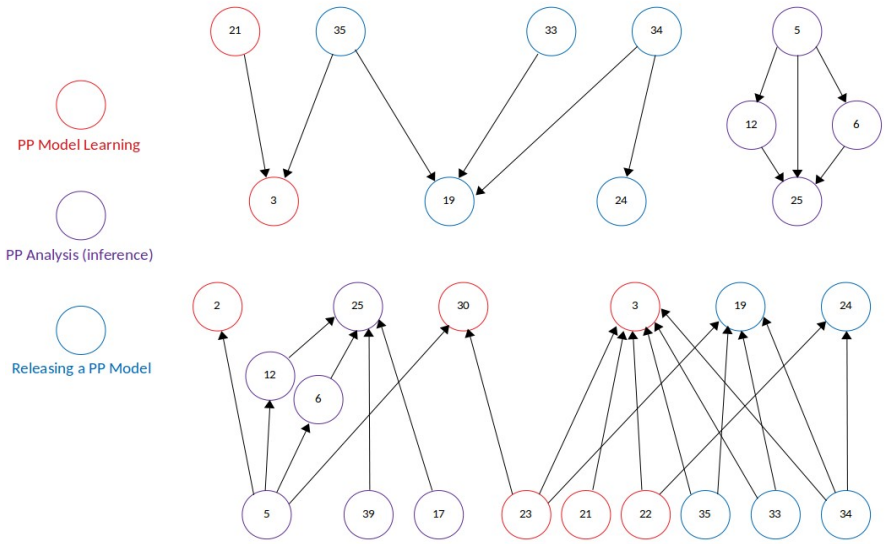

Fig. 3. Relationship between privacy-preserving deep learning techniques

In Figure 3, we show the relationship between reviewed techniques as a directional graph. In the upper part, a link (A,B) denotes that technique A was compared to technique B, while in the bottom part, a link (A,B) denotes that technique A reviewed, discussed, and sometimes criticized the limitations of technique B. It has been observed that [3], [19], [24], and [25] are the most reviewed works by others, i.e., considered as the most popular techniques within the three main classes of the proposed taxonomy. Specifically, [3], [25], and [19, 24] are the most popular techniques among PP Model Learning, for PP Analysis (inference), and Releasing a PP Model, respectively.

3. Privacy-preserving (PP) model learning

Deep learning experience is particularly improved by leveraging powerful infrastructures such as clouds and adopting collaborative learning. However, this comes at the expense of privacy, especially when sensitive data are processed during the training. In this section, we present the multi-level classification that is related to the privacy-preserving model learning. Then, we give a summary description of the relevant techniques, and finally we discuss them and derive some learned lessons.

3.1 Classification

In the literature, a number of techniques have been proposed, as shown in Figure 4. They consider collaborative training, i.e., the training is performed collaboratively between different participants or, individual training where the training is performed by a single participant, such as a client that wants to take advantage of a cloud to train its own model. In the latter case, some techniques ([8], [52], and [31]) might allow a cloud model to be trained by successive participants. Noting that some collaborative learning techniques ([20] and [23]), can also be applied for individual training.

As depicted below, reviewed techniques are based on different key concepts such as encryption, model splitting, or partial parameters sharing, with which a number of technologies are being employed, as will be described later.

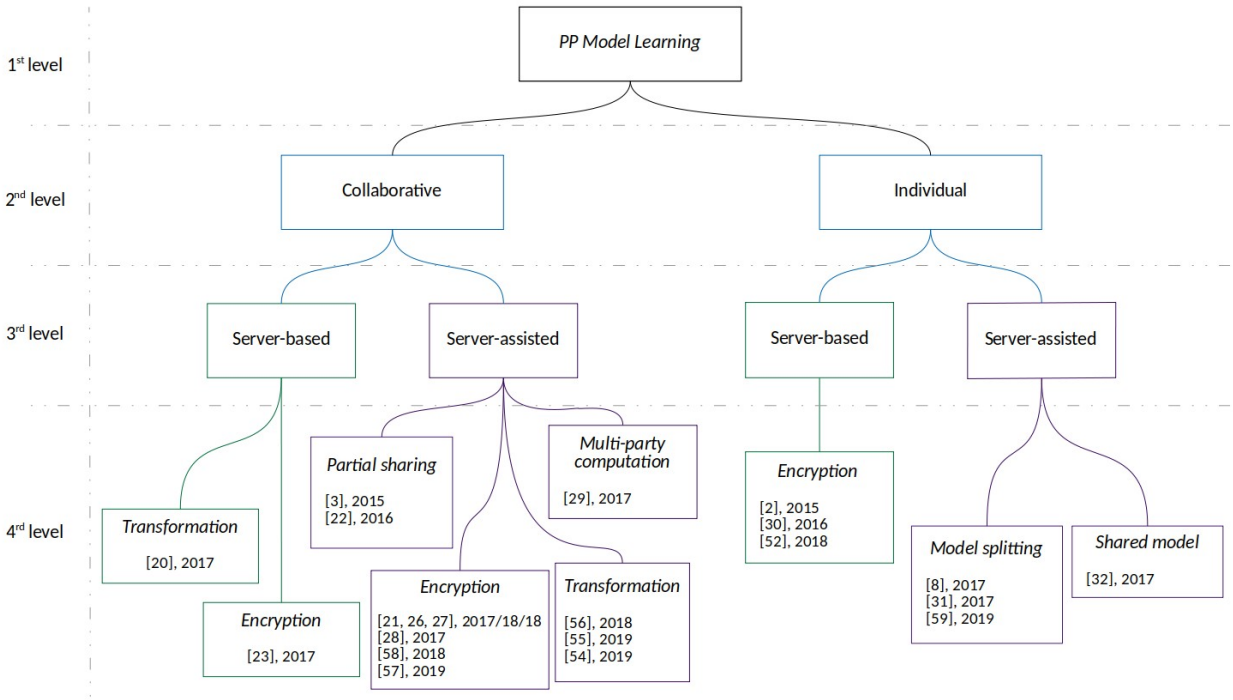

Fig. 4. Taxonomy of PP Model learning techniques

3.2 Solutions review

Based on the classification presented in 3.1, we describe in this section the reviewed solutions (see Tables 2 to 5) along with a summary of evaluation results with respect to the performance metrics (see the below Table 1).

Table 1. Main measures and properties to evaluate PP model learning techniques

Performance metrics	Measures and properties
Effectiveness	Accuracy, Reconstruction rate, F-score, Area Under Curve (AUC), Mean Relative Error (MRE)
Efficiency	Training time, Communication overhead, Computational overhead Support of unreliable networks, participants dropouts and resource-constrained devices, ...
Privacy	(Direct leakage) privacy of training data, (Indirect leakage) privacy of parameters, Privacy budget consumption, Resistance to ICA or MAP estimation attacks, ...

a) Collaborative PP model training: It can be either (1) "server-based" where the learning process is performed exclusively on an external-to-participants infrastructure such as a cloud, or (2) "server-assisted" where the learning process is performed cooperatively between the participants and the infrastructure. In both cases, mechanisms to enable privacy-preserving such as encryption are used by one or both sides, or even by a third party.

a.1) Server-based collaborative PP model learning

Table 2. Reviewed solutions for PP server-based collaborative model learning

Reference *Target task*	**Key concept** Main techniques & technologies	**Adversary & Experimentation**
Lyu et al. (2017) [20] *Learning on cloud service a DL model on the union of data contributed by a large number of participants*	**Transformation** - Repeated Gompertz (RG) *for data perturbation* - Row-orthogonal random projection (RP) matrix *for projecting high-dimensional data to lower dimension*	- Cloud service semi-honest - Synthetic and real datasets (Adult, HAR, MH, ...) - LSTM-CNN model (authors *proposal*) Comparison : *Privacy* > RP, tanh+RP, DL+RT schemes | *Model accuracy* > LSTM, CNN, DBN
Li et al. (2017) [23] *Allow multiple participants with different datasets to learn collaboratively a Neural Network model through a cloud*	**Encryption** - Multi-key Fully Homomorphic Encryption (MK-FHE) - Polynomial approximation *of activation function – Taylor series* - Secure Multi-party Computation (SMC) - Double decryption mechanism (BCP scheme)	- All participants honest-but-curious - Cloud and authorized center not colluding - Face recognition application (theoretical analysis)

Lyu et al. [20] proposed a two-stage privacy-preserving scheme called RG-RP, based on randomization and perturbation. First, repeated Gompertz (RG) nonlinear perturbation is applied on training data to mitigate maximum a posteriori (MAP) estimation attacks. Second, high-dimensional data is projected to a lower dimension, using a row-orthogonal random projection (RP) matrix, in order to resist Independent Component Analysis (ICA) attacks, maintain accuracy, and reduce transmission energy. Using transformed data of the participants, the cloud builds a deep learning model. Evaluation results showed that combining the proposed LSTM-CNN model with RG+RP privacy scheme provides a good trade-off between accuracy and privacy. As for effectiveness, LSTM-CNN outperforms comparison models with and without privacy, although LSTM-CNN privacy version achieved slightly less accuracy in comparison to the non-privacy version. As for privacy, resistance to ICA attack and recovery resistance to MAP estimation attack were theoretically proved. Besides, LSTM-CNN was shown as potentially useful for automated health monitoring.

Differently from [20], Li et al. [23] proposed a solution with two schemes, based on homomorphic encryption and coupled with polynomial approximation. In the basic scheme, the cloud performs the learning process on encrypted datasets received from the participants, which jointly perform SMC (Secure Multi-party Computation) protocol to decrypt and extract the results. In the advanced scheme, the interaction among participants is avoided by using a non-colluding authorized center, double decryption mechanism (BCP), and Fully Homomorphic Encryption (FHE). The theoretical analysis of privacy and security against polynomially indistinguishable chosen-plaintext attack (IND-CPA) showed that that learning results remain private for the cloud and authorized center, and that the solution is resilient against corrupt data owners. However, the challenge of reducing computing and communication costs was set as a future work.

a.2) Server-assisted collaborative PP model learning

Table 3. Reviewed solutions for PP Server-assisted Collaborative Model Learning

Reference *Target task*	**Key concept** Main techniques & technologies	**Adversary & Experimentation**
Shorki et al. (2015) [3] *Allow multiple participants to jointly*	**Partial sharing** - Partial sharing [21] - *Selection of parameters to*	- Passive adversary model - MNIST, SVHN datasets - MLP, CNN models

learn an accurate model over their private datasets	*train and share* - Differential privacy - *Laplace mechanism* - Sparse vector technique (SVT)	Comparison : Centralized SGD on entire dataset, privacy-violating model, Non-collaborative models learned by participants individually
Liu et al. (2016) [22] *Enable multiple participants to collaboratively learn a model in a distributed mobile environment*	**Partial sharing** - Partial sharing - *Selection of parameters to train and share*	- HAR, Human Activity Recognition datasets - MLP and CNN models Comparison : Centralized deep learning trained by mini-batch SGD on entire dataset
Phong et al (2017/2018) [21, 26, 27] *Allow many learning participants to collaboratively learn a Deep NN over a combined dataset of all*	**Encryption** - Additively Homomorphic Encryption schemes – *LWE-based and Paillier* - TLS/SSL secure channels	- Cloud server honest-but-curious & Participants honest - MNIST, SVHN datasets - MLP and CNN models Comparison : Ordinary ASGD / [3] for CNN
Zhang et al. (2017) [28] *Train accurate deep neural network models jointly by participants*	**Encryption** - Lightweight HE – *El Gamal* - Threshold secret sharing - *Shamir's SS [ρ(rho)-visibility]* - Local differential privacy (DP) - *DP randomization, sampled from symmetric distributions e.g., Laplace and Gaussian*	- Semi-honest attack model - MNIST and SVHN datasets (respectively simple and hard DL tasks) - MLP and CNN (resp. weak and strong DL models)
Hao et al. (2019) [57] *Train a cloud deep model through privacy-preserving federated deep learning*	**Encryption** - Additively homomorphic encryption - Differential privacy - *Laplace mechanism*	- Honest-but-curious server, cloud server colluding with multiple users - MNIST dataset - CNN model Comparison : Non-private model > accuracy, [27] > communication overhead and computational cost
Phong et al. (2018) [58] *Learn a deep model over the datasets of multiple data owners*	**Encryption** - Symmetric encryption - TLS/SSL secure channels - Optional enhancements : *Differential privacy, dropout, ...*	- Server assumed to be honest-but-curious - 5 UCI (Pima, breasts, ...), MNIST, CIFAR-10/100 datasets - Different DNN models, MLP & CNN, and CNN & ResNet
Bonawitz et al. (2017) [29] *Aggregate in a federated learning setting, user-learned model updates of a DNN model*	**Multi-party computation** - One time pad - Threshold secret sharing - *Shamir's secret sharing protocol* - Diffie-Hellman key exchange - Double-Masking structure - UF-CMA *secure signature scheme*	- Honest-but-curious *and* malicious settings - Focused on mobile devices setting i.e. communication is extremely expensive and dropouts are common - Deep neural network (DNN) model
Zhao et al. (2018) [56] *Collaboratively build a deep model, (i) with the overall data, (ii) without a central data storage, and (iii) reducing the impact of participants with low quality data*	**Transformation** - Functional exponential mechanism - Polynomial approximation *for objective function* - Cryptography and hashing *against eavesdrop attacks*	- Passive and active adversaries - Integrated Public Use Microdata Series (US), MNIST datasets - MLP (prediction), compact CNN (classification) models Comparison : [3], centralized and standalone schemes
Hartmann et al. (2019) [55] *Train a model on data spread over different clients*	**Transformation** - Cancelable noise (differential privacy) - Anonymization network (such as Tor)	- At least two honest clients - Malicious server possibly colluding with a maximum of 2 clients
Fu et al. (2019) [54] *Train a deep model on a distributed data*	**Transformation** - Mixup *data augmentation*	- White and black-box settings - CIFAR 10/100 datasets - DNN model Comparison : Vanilla-learning

Shokri et al. [3] and Liu et al. [22] adopted a "partial parameter sharing"-based approach for collaborative model learning.

In [3], each participant trains a chosen fraction of parameters, then shares resulting gradients with others directly or via a central server, so that they update their local models. This process is repeated until an approximate minimum is obtained. Instead of random selection of shared parameters, a smarter strategy is to choose parameters far from local optima, i.e., with a larger gradient. Besides, a stochastic vector technique (SVT) with Laplace mechanism is proposed to prevent potential leakage regarding gradients selection and actual values. The assessment of the solution presented good results beside comparison models, even with small fraction sharing, when using differential privacy. As for efficiency, unreliable networks can be supported by the solution. Regarding privacy, a trade-off between accuracy and privacy is noted, but accuracy could also be increased with a larger number of participants. Besides, [23] noted that global/local optimal might be difficult to achieve in partial sharing technique, while [21] theoretically demonstrated that even a small fraction of gradients stored on the cloud could be exploited to leak useful information. [33] and [34] reported that this solution might consume large portion of unnecessary privacy budget to ensure accuracy, while [35] stated that large number of parameters prevents meaningful privacy guarantee as bounded per-parameter.

Similarly to [3], Liu et al. [22] proposed sharing only very small fraction of parameters to preserve privacy. Starting from the most updated parameters retrieved from global server, each participant performs local training, then uploads resulting parameters with the largest gradients. Two protocols for parameters exchange are proposed, namely round-robin, and blocking asynchronous scenario where each participant works independently without any order. In the latter, the global server defines its status through a flag variable. The authors evaluated the effectiveness through reconstruction rate as they argue that accuracy cannot reflect how performance is affected by decentralized architecture and parameter selection. The results showed that reconstruction rate can stay around 90% for sharing fraction up to 0.01, while it slightly decreases as the sharing fraction decreases. As for efficiency, it was shown that network traffic depends on mini-batch size, as well as on the network architecture like the number of model parameters. Network traffic can be however reduced though parameter selection. As for privacy, the analysis of potential leakage risks using small fraction of gradients presented by [21] and described above might be also valid for this solution.

Encryption-based approach as in [21] and [28] is another interesting concept for collaborative model learning.

Phong et al. [21, 26, 27] proposed a solution where, starting from the initial weights downloaded from the cloud, participants send after each iteration of local training, the computed encrypted local gradients to the cloud. Each participant uses a different TLS/SSL secure channel in order to protect the integrity of homomorphic ciphertexts. The cloud then updates the encrypted global weights vector, which is downloaded by the participants. The solution achieves theoretically the same accuracy as conventional asynchronous SGD, while through evaluations, 97% and 99% of accuracy were achieved for MLP and CNN respectively. As for efficiency, an overhead in communication and computation was noted, although the authors considered it as small. However, the trade-off accuracy/privacy could be shifted to efficiency/privacy, which allows to keep the accuracy intact, while it can be furthermore solved by involving more processing units or dedicated programming codes. As for privacy, encrypting gradients theoretically ensure that no information is leaked to the cloud.

Zhang et al. [28] proposed a solution with two variants ($\propto$MDLc and $\propto$MDLd) under loose and tight coordination respectively were proposed. In $\propto$MDLc, participants locally train their model, and apply differential privacy on computed gradients of each parameter. A participant registers a request for a set of parameters with the Manager M. If the minimum of registered participants for a parameter is reached, M invokes participants, which encrypt their gradients, and upload them to M. The global model is updated and participants are notified to download it to update their local models. In $\propto$MDLd, there is no registration process, and the global gradient is immediately updated once the threshold of participants is reached. Less coordination is therefore required for higher execution efficiency and fault tolerance, but with more communication complexity. Evaluation results showed desirable model utility close to standalone privacy-violating model, as the negative impact of LDP on utility can be mitigated using ρ-visibility. However, as larger ρ implies higher training cost per epoch, a decision for balancing utility and training efficiency should be made in low-end systems and large-scale training. While encryption implies fairly limited system overhead in comparison to training the local models, proposing two variants further allows to control communication and computational costs. As for privacy, presented analysis showed that combining LDP and ρ-visibility mechanisms ensures both global and local gradients privacy.

A recent solution based on encryption was proposed by Hao et al. [57]. The process is simple, users perform local training over their private datasets, perturb and encrypt the resulting local gradients, then upload them to the cloud which computes the global encrypted gradients. On that basis, users update their model parameters with the global gradients after decryption. The above process is repeated until loss function reaches minimum. The authors of [57] presented a theoretical security analysis and considered the solution as secure against server cloud and compromised users. Besides, performance evaluations described by authors showed better results in terms of communication overhead and computing cost compared to [27], and high accuracy compared to a non-private model.

Phong et al. [58] recently proposed a solution where the trainers share the model weights instead of the gradients, arguing that it is more robust against information leakage. In Server-aided Network Topology (SNT) version, a trainer retrieves from the central server the latest weights, and performs local training. It then uploads to the server computed weights vector encrypted with a key shared by trainers. Once the process is repeated by all the trainers, the best weights vector stored at trainers is shared among others. In Fully-connected Network Topology (FNT) version, the same process is followed except that the weights are directly

transmitted over trainers randomly or following an agreed order. The authors theoretically proved a certain degree of privacy and performance guarantees through a series of theorems, but they pointed out that although exactly inverting input data from the weights was hard, some information might however be leaked. In this context, techniques like differential privacy, dropouts, and anonymous transmission were proposed. Besides, through experiments conducted over different datasets and using different architectures, the authors considered their solution as both effective in terms of accuracy and efficient in terms of communication and computation.

A different approach is proposed by Bonawitz et al. [29] on the basis of Secure Multi-party Computing (SMC). After that the participants evaluate their local models, a third party (not necessarily trusted) securely aggregates updates, and the aggregated model is shared with the server. Two variants are proposed in order to guarantee privacy against both honest-but-curious and malicious adversaries. Secure aggregation uses masking with one time pad through addition and subtraction of random masking vectors method in order to ensure that inputs are not revealed. Pseudo-random generator is used in order to mitigate communication overhead of random vectors exchange, while threshold secret sharing is used to better handle dropouts of participants. In case the server could reconstruct masks, double masking structure is used to protect users data. For malicious adversaries, a public-key infrastructure (PKI) is assumed, and a consistency check round is added before unmasking, in order to prevent the server from reconstructing more secrets that it is allowed to. Evaluation of the approach showed a low overhead in terms of client runtime and communication, as well as tolerance to large number of failing devices, which might make the solution ideal for mobile applications. However, unlike bandwidth and client runtime, the server running time significantly increases with the fraction of dropouts. As for privacy, security against honest-but-curious and malicious adversaries was theoretically proved. However, no protection is provided against malicious clients trying to prevent the server from learning any sum at all.

Another interesting concept for local models parameters aggregation approach with privacy preservation is the introduction of transformations to input data [54], models parameters [55], or loss function |56]. In the following three techniques, the participants locally train private models over their local datasets, then share the resulting model parameters with a central server for aggregation. Once obtained, the new parameters vector is distributed over the participants.

Zhao et al. [56] proposed a solution where objective functions of local models are perturbed in the training process using functional exponential mechanism, in order to prevent leakage from local shared parameters. To perform perturbation, the polynomial approximation of the objective function is derived with two different loss functions, then noise is injected to the coefficients. Besides, the solution considered the potential existence of "irregular participants" having low-quality data. In order to reduce the impact of such participants, an accuracy score for participants is calculated by running the model with each participant's weights over an auxiliary data set, then decide whether one model parameters are accepted for aggregation or not. A theoretical security analysis of the solution concluded that ε-differential privacy is guaranteed, while cryptography and hashing were proposed in case of eavesdropping attacks of malicious adversaries. Besides, experimental resultsshowed that the solution could ensure rigorous privacy, high accuracy (measured using MRE, Mean Relative Error), while being robust to irregular participants. However, a trade-off between accuracy and privacy is noted.

Hartmann et al. [55] recently proposed a technique based on differential privacy noise-based transformation. The idea is to add cancelable noise to local gradients before uploading them to the server. Local gradients become useful again after that users send negative value vectors of added noises to the server. To guarantee that negative values are not linkable to a specific gradient, they are sent through an anonymization network such as Tor. The server adds them then to the sum of the gradients, gets the global gradients, and update the model parameters. For malicious adversaries, some solutions are proposed such as how to detect server protocol violation through iterating requests and examining responses, or using hashes which reduces communication costs. A theoretical analysis was presented to demonstrate the provided privacy guarantees. A computation and communication analysis showed that, in comparison to standard distributed gradient descent, communication increases by only a logarithmic factor in the number of users and parameters. However, a training round needs to be restarted whether a client drops out. The solution was considered therefore more suited for cases where clients/server connection is stable.

Differently from the other solutions, Fu et al. [54] proposed a solution based on mixup data augmentation. In this approach, each client processes its local data using mixup, then it trains its local model on the new data. The resulting local models parameters are uploaded to the server for aggregation. The aggregated model parameters are then distributed to each client, which iterates the training process until the model converges. Experimentation results showed that the approach is suitable for image classification, and that the model has

better generalization ability with mixup, when compared to distributed vanilla-learning (VL). Over text classification, the solution achieves a high AUC, which is slightly better than VL. As for privacy, through an analytical experiment on the facial recognition task under white-box settings, the solution achieved good visual and quantitative results using hamming distance. However, a privacy/accuracy trade-off was noted.

b) Individual PP model training, similarly to collaborative training, it can be classified as: "server-based" or "server-assisted", but there is only a single participant, which trains its own model or the server model.

b.1) Server-based individual PP model learning

Table 4. Reviewed solutions for PP Server-based Individual Model learning

Reference *Target task*	**Key concept** Main techniques & technologies	**Adversary & Experimentation**
Bu et al. (2015) [2] *Train a local model on cloud*	**Encryption** - Full Homomorphic encryption - *BGV* - Polynomial approximation *for activation function - Maclaulin formula*	- STL-10, CIFAR-10 Image classification datasets - Stacked Auto-Encoder model Comparison : Conventional stacked auto-encoder (high-order back-propagation)
Zhang et al. (2016) [30] *Training a model for big data feature learning by offloading expensive operations to the cloud*	**Encryption** - Full Homomorphic encryption – *BGV* - Polynomial approximation *for activation function - Taylor theorem*	- STL-10 and NUS-WIDE, PeMS and DleMP datasets - Several stacked Tensor auto-encoders Comparison: (Accuracy) Conventional Deep model / (Training) Conventional high-order back-propagation algorithm
Zhang et al. (2018) [52] *Contribute in the training of a server-based deep neural network model*	**Encryption** - *Paillier* partially homomorphic cryptosystem - Outsourcing non-linear encrypted computation to client	- Semi-honest model - Iris, Diabetes, kr-vs-kp and MNIST datasets - DNN, CNN models Comparison : Cryptonets [25] and BGN-Net

Based on encryption, the privacy-preserving back-propagation algorithm proposed by Bu et al. [2], relies on BGV fully homomorphic encryption and Maclaulin polynomial approximation of sigmoid activation function. Input data and initialized parameters are encrypted by the client and uploaded to the cloud, which runs one iteration. The client downloads and decrypts the results, and updates its local model once. It then encrypts and sends the updated parameters again to the cloud, which performs another iteration. This process continues until the maximum error threshold or the maximum number of iterations is reached. Although BGV encryption allows to protect private data during learning process, it however requires approximation of activation function. This might lead to accuracy reduction as evaluation results showed a loss of 2%. As for efficiency, although the solution could achieve a 2 times higher efficiency, i.e., 45% training time of the conventional model, it incurred however computation and communication overhead because of the encryption-related steps over each iteration.

Similarly to [2], the technique proposed by Zhang et al. [30] follows the same steps but using Taylor theorem in order to polynomially approximate the sigmoid activation function. Evaluation results showed accuracy loss for both classification and prediction tasks. However, the authors mentioned adding more Taylor series in order to reduce the loss in classification, but this increases BGV encryption level and, thus leads to low performance. As for efficiency, the solution could achieve over classification 2.5 times, and 2 times higher efficiency in terms of learning time for one iteration, and overall time respectively. Over prediction, the overall training time is almost the same as non-privacy-preserving model, while efficiency could be improved by almost 2 times when data size is larger. It is also reported that the scheme performance could be improved by introducing more cloud servers, which makes it particularly suitable for big data feature learning. However, [5] criticized the overall solution for its very high communication complexity.

A more recent solution based on encryption was described by Zhang et al. [52]. A client, willing to contribute to the training of the model, sends its data encrypted using Paillier scheme to the server, which performs all possible neural network computations except of non-linear activation functions. In order to resume the execution, the encrypted weighted sums before each activation function are sent back to the client, who will be responsible of performing the computation. The result is then re-encrypted and sent again to the server. The authors of [52], on the basis of conducted experiments, considered the training as stable, and without accuracy loss. Moreover, the solution achieved a speed-up of 14 to 35 times in computation speed compared to CryptoNet. However, computation and communication overheads on the client side need to be evaluated, especially for deeper models as the communication between client and server is required in order to locally compute each activation function, which can be costly.

b.2) Server-assisted individual PP model learning

Table 5. Reviewed solutions for PP Server-assisted Individual Model Learning

Reference *Target task*	**Key concept** Main techniques & technologies	**Adversary & Experimentation**
Dong et al. (2017) [8] *Model provider to train a NN for user*	**Model splitting** - 1st layer at local-side - Dropping connections and activation outputs (*Hadamard product*) - Dropout and Dropconnect	- MNIST, CIFAR10 (more challenging) datasets - DNN and CNN models
Li et al. (2017) [31] *DNN model training based on cloud*	**Model splitting** - Local feature extractor - Channel pruning and selection	- Worst-case evaluation : attackers have representations, labels and original data but not the transformation induced by the FEN - CIFAR-10/100 - VGG16 pre-trained on Imagenet dataset (CNN)
Yu e al. (2019) [59] *Train deep learning models on cloud*	**Model splitting** - Step-wise activation functions	- Even in case weights are compromised by the adversary - MNIST, CIFAR-10 datasets - CNN (MobileNetV2)
Servia-Rodriguez et al. (2017) [32] *Train a personalized model built from a shared model learned from a relatively small set of others parties*	**Shared model** - Tuning shared model with personal data - *Start from the weights and bias of the shared model* - Differential privacy *for training shared model - Optional*	- WISDM / Activity recognition (accelerometer traces) - MLP with 2 layers / 1 hidden layer Comparison : *Shared model* : trained using (N – 1) subjects data & tested using remaining subject data / *Local* : trained and tested using 1 subject data

Dong et al. [8] proposed a technique where the model layers are split among the client's local device and the server. The first layer of the model is migrated to the local device where its activation function is applied locally. During the feed-forward stage, a method for dropping activation output encrypts input data and make the output of invertible activation functions non-invertible. The method randomly sets some activation outputs to zero using hadamard product between activation function outputs and a binary vector. During back-propagation stage, dropout and dropconnect methods are used in order to prevent over-fitting. Non-invertible functions were discussed and a mathematical proof of possible encryption was presented. The presented results showed that the input data can be encrypted by only dropping a few activation outputs without leading to noticeable degradation of performance, noting that the same result can also be achieved by adding small noise values to few activation outputs.

Similarly, Li et al. [31] used the concept of model splitting, but with a more advanced local part compared to [8]. The local NN represents the feature extractor network (FEN), and the cloud NN targets the learning task based on locally extracted intermediate representations. The solution focused on the FEN topology, which is determined by the three factors, number of layers, depth of output channels, and subset of channels selected as the output. The FEN is selected from a pre-trained NN pool. The number of input and output layers is determined using pre-characterization flow, which considers local and cloud constraints, i.e., performance and storage profile of local platform, and cloud-based privacy characterization. The selection of FEN output channels is subsequently conducted using supervised pruning of output channels with the worst utility. Moreover, in order to protect anonymity of the FEN, pre-trained NN pool can be enlarged and channel selection applied on both intermediate and output channels, which make it harder to guess how FEN is derived and which channels form it. Good accuracy could be achieved through evaluation using useful features embedded in the released representations. Privacy was evaluated in terms of distance between the reconstructed and original images. The authors described how selecting the local NN topology allows to protect privacy through controlling the specific features to release. Moreover, even if the pre-trained NN is known, it still remains very hard to determine the number of FEN layers an channels. Besides, it was noted that the accuracy/privacy trade-off can be controlled by the number of FEN layers, and the output channel depth and selection. However, Sharma et al. [36] reported that intermediate representations are still identifiable visually.

A recent solution based on model splitting is proposed by Yu et al. [59]. At the local part comprising the first layer, the private data is processed into an irreversible transformed data (called metadata). On their basis, the cloud part, comprising the rest of the neural network layers, continues model's training. In order to make the metadata irreversible, the authors described how to modify the activation function at the local side to be step-wise by dividing the input domain into intervals. Experimental evaluation of [59] showed a trade-off between accuracy and privacy, which is controlled by interval value. In fact, as the interval value decreases, reversed inputs become more unrecognizable, which increases privacy but decreases accuracy. However, the

solution could achieve good accuracy while creating considerable difficulties to recover original data. Consequently, proper interval parameter value could be investigated to be set according to the target settings requirements.

From a reverse perspective, Servia-Rodriguez et al. [32] proposed to start from a model learned and shared by the cloud, to produce a local personal model through fine tuning. The weights of the shared cloud model are initialized to random small values, with bias set to zero, and the training is done using sets of voluntarily users data. Differential privacy can also be optionally used to ensure confidentiality of such data. The shared model once distributed to a user, can be use it directly, or locally tuned (i.e. re-trained) with private user data in order to obtain a personalized model. Evaluation showed that fine-tuned models could perform with higher accuracy than comparison models, but if enough local samples are available. In fact, it was noted that during local retraining, performance slightly drops while the model is adapting to the new scenario, but with more local samples, the accuracy exceeds the one obtained with the shared model. Besides, the solution was shown as robust against poisoning attacks in both dumb and smart scenarios, whereas privacy is guaranteed for non compromised devices since personal data and personal model never leave the devices.

3.3 Discussion and learned lessons

We present a summary table of comparison (Table 6) of the privacy-preserving-learning techniques that are reviewed in this section, then we discuss them, and identify a number of learned lessons and insights with respect to the key concepts.

Table 6. Comparison of reviewed PP Model learning solutions

IND : Individual training
COL : Collaborative training
SUC : Training by successive participants
+ : Alternative use of the technique

SE: Exclusively on external server(s)
SC: On both client and server(s)

TR : Transformation*
EN : Encryption
PS : Partial sharing
MC : Multi-party computation
MS : Model splitting
SM : Shared model
IO : Irreversible layer output**
op : Optional technique

TD: Training data
LG: Local trained model gradients
LW: Local trained model weights
LO: Final output of the model local part

CBC : Communication required between clients
ECS : Extra communications*** required between clients and server(s)
LTI : After each iteration**** of local model training
STI : After each iteration of server-based model training
AFI : At each activation function computation of each training iteration
LSI : After model layer separation (last local layer) of each training iteration

** Including differential privacy, randomization, mixup, ...etc. It can be applied on input data, model parameters, ...etc.*
*** Including dropping connections, step-wise activation functions, ...etc.*
**** Apart from initialization, input, and end of training*
***** Can be by sample or mini-batch. If the entire dataset is used at once, there will be no iteration*

** Indicates the main exposed information to be prevented from leaking information related to training data*

Reference	Clients collaboration	Training location		Main privacy-enabling techniques							Information protected*	Additional server(s)	CBC	ECS
		SE	SC	TR	EN	PS	MC	MS	SM	IO				
Lyu et al. (2017) [20]	COL, +IND	•		•							TD			
Li et al. (2017) [23]	COL, +IND	•			•		•				TD	*Adv. scheme* •	*Basic scheme* •	
Shorki et al. (2015) [3]	COL		•	•		•					LG		*If no server - LTI* •	*LTI* •
Liu et al. (2016) [22]	COL		•			•					LG			*LTI* •
Phong et al. (2017/18) [21, 26, 27]	COL		•		•						LG			*LTI* •
Zhang et al. (2017) [28]	COL		•	•	•		•				LG			*LTI* •
Hao et al. (2019) [57]	COL		•	•	•						LG			*LTI* •
Phong et al. (2018) [58]	COL		•	*op*	•					*op*	LW		*If no server - LTI* •	*LTI* •
Bonawitz et al. (2017) [29]	COL		•		•		•				LG	•		*LTI* •
Zhao et al. (2018) [56]	COL		•	•	•						LW			*LTI* •

Hartmann et al. (2019) [55]	COL		•	•							LG			LTI •
Fu et al. (2019) [54]	COL		•	•							LW			LTI •
Bu et al. (2015) [2]	IND	•			•						TD			STI •
Zhang et al. (2016) [30]	IND	•			•						TD			STI •
Zhang et al. (2018) [52]	IND, +SUC	•			•						TD			AFI •
Dong et al. (2017) [8]	IND, +SUC		•					•		•	LO			LSI •
Li et al. (2017) [31]	IND, +SUC		•					•			LO			LSI •
Yu e al. (2019) [59]	IND		•					•		•	LO			LSI •
Servia-Rodriguez et al. (2017) [32]	IND		•	*op*					•		*No information exposed*			

Six typical key concepts are used for privacy-preserving learning of a deep model, namely: shared model, multi-party computation, transformation, partial sharing, model splitting, and encryption.

Some researchers pointed out that using encryption concept in the training stage prevents data scientists from manually tuning the model, inspecting data and trained models for potential corrections of mislabeled items, and adding features [25]. Nevertheless, if manual tweaking is not considered and encryption is adopted, two main techniques can be used namely, Homomorphic encryption and Yao's garbled circuit. However, some observations need to be considered:

- Yao's garbled circuit technique might not be the best choice or at least might need more investigation. In fact, it was reported in [9] that Yao's garbled circuit is not suitable for model training because of the large cost of nonlinear activation functions, and its deployment in collaborative environment which is not easy.
- The choice of activation function with appropriate polynomial approximation when opting for homomorphic encryption might still need more investigation. In fact, it is observed that different results were obtained using different functions and approximations methods. In [23], sigmoid function was approximated using Taylor series, while in [2], Maclaulin formula is used for approximation, which leads to accuracy loss. In this context, [30] proposed adding more Taylor series terms in order to reduce accuracy loss implied by polynomial approximation. However, this also leads to increasing the encryption scheme levels and degrading the privacy-preservation back-propagation algorithm performance.
- Still in case of homomorphic encryption, both [2] and [30] adopt a transmission method (download/upload) of updated weights to parties for decryption and re-encryption after each iteration [2, 30]. This allows to avoid too big multiplicative depth, but, it also incurs computation and communication overhead [2, 5].
- Hao et al. [57] reported that symmetric additively homomorphic encryption has excellent efficiency and could address computational and communication overhead faced by other public-key encryption schemes.

The partial-sharing concept has a positive impact on privacy (indirect leakage) and network traffic load, and it shows good results in terms of accuracy [3, 22]. However, because data is not fully utilized in partial sharing, global/local optimal might be more difficult to achieve [23]. Consequently, it could be interesting to investigate more this concept with differential privacy and other relevant techniques like homomorphic encryption. An appropriate trade-off involving accuracy and privacy should accordingly be explored, taking into consideration potential controlling parameters, like the fraction of sharing and the number of participants, which have a positive impact on accuracy [3].

As for model splitting concept, dropping few activation outputs at local side can protect input data without noticeable performance decrease [8]. Besides, using step-wise activation functions [59] and exploring other similar methods in order to make data fed into cloud side irreversible, is an interesting path to further

investigate.

The approach described in [29] based on Secure Multi-party Computation concept using data masking could fairly deal with common dropouts. However, efficiency in terms of server running time needs to be more investigated as it is significantly impacted by the fraction of dropouts [29].

For data transformation concept, it is noted that applying Repeated Gompertz (RG) nonlinear perturbation function on training data allows to mitigate Maximum A Posteriori (MAP) estimation attacks [20]. Besides, mixup data augmentation technique is interesting to investigate. Mixup could : be used for regularization, help enhancing models against adversarial samples, help models to better generalize, as well as enhance the protection of models against inversion attacks [54].

In addition, the following general observations can also be considered:

- Round-robin parameter exchange strategy achieves better accuracy than asynchronous strategy when aggregating local trainings in collaborative model learning. However, the speed of learning in this case is determined by the slowest participant, which is suitable in case of participants with similar computation capacity [3].
- $\wp$-visibility, enforcing a generalized abstraction of secure aggregation (of local trainings), can be used as another privacy protection layer in the post-processing stage of Local Differential Privacy (LDP). It is also used to mitigate the negative impact of LDP on utility. However, although larger $\wp$ implies higher but not prohibitive training cost, it is advised for low-resource configuration (like limited bandwidth) and large-scale training tasks to consider a trade-off between training efficiency and model accuracy [28].
- Accuracy score calculation of individual trainers could be an interesting approach to explore in order to increase the global training efficiency. In [56], the approach was proposed to reduce the impact of participants with low-quality data on the training process.
- Adaptive redistribution of noise is an interesting technique for accuracy improvement. In [33], better accuracies were achieved in comparison to [19] for both small and large privacy budgets. This was achieved by using noise redistribution on the basis of features relevance to the model output.
- In [58], local models weights are shared instead of gradients for local models aggregation. The authors argued that this method is more robust against information leakage, in the sense that inverting input data from the weights can be considered as a problem of solving a system of nonlinear equations, with a smaller number of equations than variables. However, the authors pointed out that some information might still be leaked.
- Given the importance of the accuracy, a trade-off between accuracy/privacy could make a privacy-preserving model less attractive. Therefore, exploring an efficiency/privacy trade-off can be a more interesting approach. In fact, such a trade-off, besides being without accuracy loss, can be solved by involving more processing units and dedicated programming code, as it is reported in [26].
- Reconstruction rate measure and its utility might need to be more investigated with respect to accuracy. In fact, it was reported in [22] that reconstruction rate could reflect some specific properties such as how performance is affected by decentralized architecture and parameter selection.
- LSTM-CNN outperforms LSTM, DBN and CNN models in [20] and might be a potential adapted deep model for health monitoring as experimental results indicated that it was useful in distinguishing activities performed by multiple participants.
- Low overhead of runtime and communication, and tolerance to a large number of failing devices are among the concerns to be considered for the support of mobile applications [29].
- Dropout and Dropconnect methods are interesting to be used for preventing over-fitting during back-propagation stage [8].
- Transmission energy can be saved by projecting high-dimensional data to lower dimension (compressing data) using row-orthogonal random projection (RP) matrix on input data, knowing that RP does not have any impact on accuracy degradation [20].
- It is interesting to establish TLS/SSL secure channel for communication between user and external

infrastructure in order to protect data integrity [21].

Lastly, in order to inherit the advantages of different key concepts such as strong privacy guarantees of encryption or transmission energy saving through RP data compression, investigating the possibility of combining multiple concepts and their underlying technologies might come with interesting ideas. However, given the impact and drawbacks of different concepts and technologies, such an investigation should take carefully into consideration the performance outcomes of any potential promising solution in terms of effectiveness, efficiency and privacy in order to keep an adequate trade-off matching target applications requirements. Case-based combination where one or another technology is used according to the available resources or network reliability for instance might also be interesting, like exploring the extent of effectiveness of switching between encryption and partial sharing concepts in case of varying network reliability and communication costs.

4. Privacy-preserving (PP) analysis (inference)

Similarly to the model training, ~~involving~~ powerful infrastructures for deep learning analysis phase comes at the expense of privacy, especially for users private data to be processed. In this section, we present the classification that is related to the privacy-preserving inference. Then, we give a summary description of the relevant techniques, and finally we discuss them and derive some learned lessons.

4.1 Classification

PP analysis can be either "server-based" where inference process is performed exclusively on an external infrastructure like a cloud or remote server, or "server-assisted" where inference process is performed cooperatively between the user and infrastructure (see Figure 5). In both cases, mechanisms used for enabling privacy-preserving such as encryption can involve one or both sides depending on the technique.

As depicted below, reviewed techniques are based on different key concepts mainly encryption, multi-party computation, transformation, and model splitting, with which a number of technologies are being employed, as will be described later.

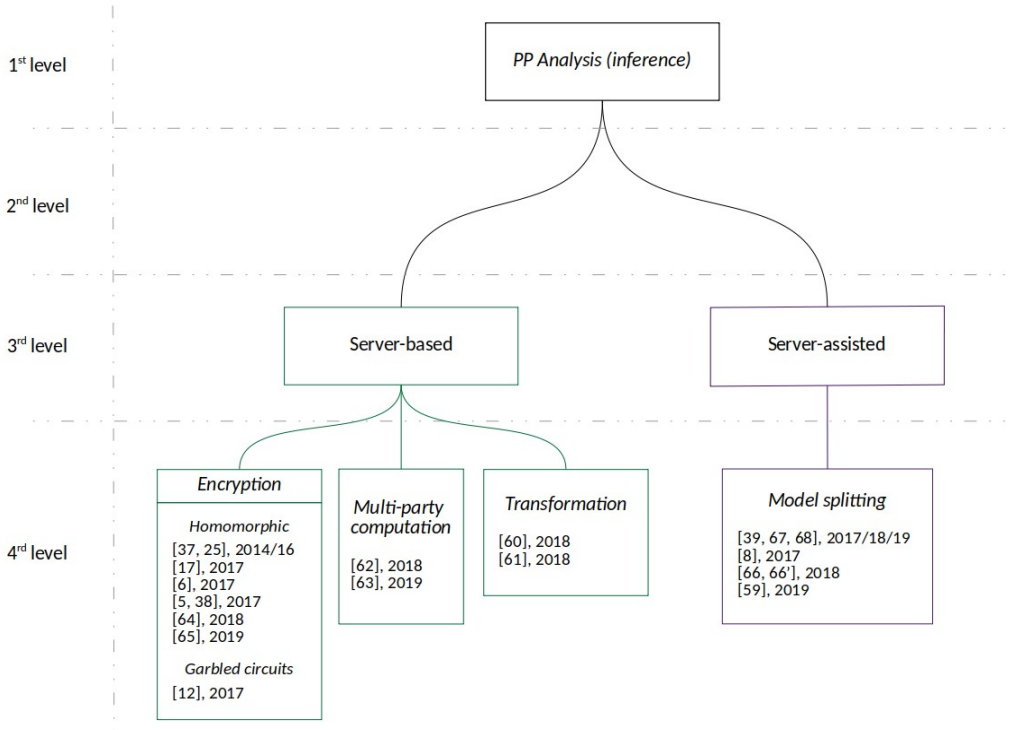

Fig. 5. Taxonomy of PP Analysis techniques

4.2 Solutions review

Based on the classification presented in 4.1, we describe in this section the reviewed solutions (see Tables 8 and 9) along with a summary of evaluation results with respect to the performance metrics (see the below Table 7).

Table 7. Main measures and properties to evaluate PP analysis techniques

Performance metrics	Measures and properties
Effectiveness	Accuracy
Efficiency	Execution time, Communication/Computational overhead, Number of predictions per hour, Number of non-linear layers supported, Independence from dataset, ...
Privacy	(Direct leakage) Protection of clients data, (Indirect leakage) Protection of predictions, Protection against various attacks involving different parties, ...

a) Server-based PP analysis

Table 8. Reviewed solutions for PP Server-based Analysis

Reference *Target task*	**Key concept** Main technologies	**Adversary & Experimentation**
Xie et al. (2014) [37], Gilad-Bachrach et al. (2016) [25] *Applying cloud-based NN*	**Encryption - *Homomorphic*** - LHE (Leveled homomorphic encryption, *a weaker but faster variant of HE) - YASHE' scheme* - Polynomial approximation *of activation function*	- Cloud model trained on unencrypted data - MNIST dataset - DNN model - *Trained network has 9 layers, and the simplified version (for prediction) has 5 layers*
Baryalai et al. (2016) [17] *Neural network based classification*	**Encryption - *Homomorphic*** - Homomorphic encrypting - *Paillier cryptosystem* - Non-colluding dual clouds - Key exchange *based on Diffie and Hellman* - Random salt	- Honest-but-curious (passive) adversaries - Clouds non-colluding
Chabanne et al. (2017) [6] *Classification on cloud with focus on CNN model and depth greater than 2*	**Encryption - *Homomorphic*** - Fully homomorphic encryption - *BGV scheme* Low degree polynomial approximation *of activation function* - Batch normalization principle	- MNIST dataset - Model *with non-linear layers > 2*, (Training) DNN *with 6 activation layers* Comparison : Cryptonets [25]
Hesamifard et al. (2017) [5, 38] *Classification on cloud with focus on CNN model*	**Encryption - *Homomorphic*** - Homomorphic encryption - *leveled HE (LHE)* - Polynomial approximation *of activation function :* approach based on the derivative of ReLU, Sigmoid and Tanh, over a symmetric interval	- MNIST, CIFAR-10, UC Irvine (Crab, Fertility, and Climate Model) datasets - CNN, DNN models - Comparison : Over plain text > [25], [6] / Over encrypted data > [25], [6], [12], [40]
Zhu et al. (2018) [64] *Machine Learning as a Service (MLaaS)*	**Encryption - *Homomorphic*** - *Paillier* homomorphic cryptosystem - Interactive protocol between client and server for ReLU computation	- Model trained in plain-text - MNIST dataset - LeNet-5 (CNN) model Comparison : Model in plain-text, [25]
Vizitiu et al. (2019) [65] *Process client data on an external deep learning-based computing service*	**Encryption - *Homomorphic*** - Fully Homomorphic Encryption - *MORE scheme*	- MNIST, Coronary angiographies datasets - CNN model Comparison : Unencrypted version
Rouhani et al. (2017) [12] *Analyze data of distributed clients by a cloud*	**Encryption – *Garbled circuits*** - Garbled circuits protocol – *Yao's GC* - Data and network preprocessing techniques – *Data projection and DL network distillation*	- Honest-but-curious adversary model - (Visual) MNIST, (Audio) Speech of speakers, (Activities) Smart-sensing datasets - (Visual) CNN, Classical feed-forward NN, (Audio & Activites) fully-connected DNN models Comparison : [25]
Huang et al. (2018) [62] *Edge computing-based CNN feature extraction for mobile sensing*	**Secure Multi-party Computation** - Additive secret-sharing encryption - Edge computing - *non-colluding dual edge servers* - Secure communication channels	- Honest-but-curious model - Edge servers independent and non-colluding - Third party honest and can be trusted - CNN model Comparison : [25], MiniONN, Chameleon
Ma et al. (2019) [63] *Online Neural-network-based prediction*	**Secure Multi-party Computation** - Secure two-party comparison - Additively homomorphic encryption - *ElGamal* - Non-colluding dual servers - Low degree polynomial approximation *of activation function*	- Honest-but-curious model - Cloud servers non-colluding
Leroux et al. (2018) [60] *Offloading deep neural networks to the cloud*	**Transformation** - Generative Adversarial Networks (GANs) - Neural-network-based obfuscation	- CIFAR-10 dataset - ResNets, GoogleNet, VGG (Classifier models) / MobileNet-like (Obfuscator model)
Raval et al. (2018) [61] *Offloading sensors personal data through mobile applications to the cloud for analysis*	**Transformation** - Generative Adversarial Networks (GANs) - Neural-network-based obfuscation	- Honest-but-curious adversary model - Handwriting Recognition (case study), KTH, HAR, Opportunity, StateFarm, CIFAR-10 - CNN, DNN

		Comparison : (Images) obfuscation vs blur, mosaic, ad advrep

Gilad-Bachrach et al. [25] and Xie et al. [37] proposed a technique based on YASHE LHE scheme. The user sends its data encrypted to the cloud, which runs the model and return an encrypted prediction. To make the network compatible with homomorphic encryption, max-pooling is replaced by scaled-mean pool function, and activation functions are approximated with the lowest-degree non-linear polynomial function, which is the square function. The authors noted these modifications should be preferably considered during training, although it is performed over plain-text data. The solution could achieve 99% of accuracy and 59000 predictions per hour on a single PC, without the need to keep constant presence of users during the model running. However, LHE if producing too much noise may decrease accuracy [12, 17], while for models with more than two non-linear layers, it becomes very low [6]. The communication cost is also relatively high compared to [5]. Moreover, because of the high and constant computational cost overhead, practicability in resource-limited settings like smartphones is bounded, and the technique is better suited for large batches processing [12]. Besides, the authors suggested that throughput and latency might be improved through accelerating computation using GPU (Graphical Processing Units) and FPGA (Field Programmable Gate Array), while finding faster homomorphic computation schemes with smaller parameters and more efficient encoding may enhance the computational overhead.

Baryalai et al. [17], in order to address heavy computation overhead of HE, introduced non-colluding dual cloud model, where two clouds A and B collaborate to produce the classification results in a secure way. The cloud A runs the neural network over private data encrypted by the client, but delegates activation functions computations to the cloud B as it holds a shared key with the client. The process is repeated until the last layer. The client A then protects the last output using random salt from cloud B, which applies softmax function and send the encrypted final result to the client. A theoretical scenarios-based security and correctness analysis was presented, explaining how the technique can successfully defend against possible attacks. The studied scenarios comprise attacks : on communication between client and cloud A to obtain input data, on cloud A, on cloud B, on communication between cloud A and cloud B (man in the middle) to intercept data exchanged between them, and on communication between cloud B and client to intercept final result.

Chabanne et al. [6] proposed a solution based on BGV FHE scheme for classification task over CNN model. The key technical innovation is the combination of polynomial approximation with batch normalization. For the training phase, a batch normalization layer is added before each ReLU layer avoiding high accuracy degradation, and max-pooling is replaced by average-pooling which is more FHE-friendly, and has a small impact on accuracy. After training and before the model is ready for classification, ReLU function is replaced with low-degree (2) polynomial approximation, and a batch normalization layer is added before each ReLU layer. Once the model is ready, the user simply needs to encrypt its private data and send it to the model, which performs requested analysis. Evaluation results showed that the solution can achieve low running time with close performance as without privacy. Compared to [25], the solution is furthermore scalable, but for deeper CNN models, a high degradation of the accuracy due to the polynomial approximation of ReLU is noted. For this, the authors proposed to build a new polynomial approximation learned from a distribution closer to the output distribution of the batch normalization, which allowed to achieve 99.30% accuracy.

Hesamifard et al. [5, 38], focused on CNN model, and used LHE scheme instead of FHE for efficiency and practicality of computations. The model is simply run on data encrypted by the user. The authors investigated methods for approximating common activation functions for CNN and the trade-off between polynomial degree and model performance. A method based on the derivative of ReLU function was proposed, and approximations of sigmoid and Tanh over a symmetric interval were studied. In a next work [38], the authors addressed the problem of HE growing noise during computations, and proposed an approach based on noise level threshold and the collaboration between the server and the client. Evaluation over plain-text showed that the achieved accuracy outperforms [6] and [25], and that the behavior of ReLU approximation is robust against structure changes. Using encryption, the achieved accuracy outperforms comparison models over MNIST, and is very close to best non-private version, while over CIFAR-10 dataset, the accuracy is close to the version with original activation function. As for efficiency, the solution is much faster than [25], and can make more predictions per hour, with relatively small communication cost than [12] and [25]. Moreover, the solution is independent from the dataset, and the number of communications and batch size are independent. However, some large datasets like ImageNet require very long time for training and usually need GPU for efficient implementation which was left by authors for future work.

A recent homomorphic encryption-based solution is proposed by Zhu et al. [64]. The user encrypts its private data and sends it to the server for prediction. Linear, convolutional, and pooling transformations are improved on the basis of Paillier algorithm. As for the activation function, the authors choose ReLU, and proposed, instead of using polynomial approximation, an interactive protocol between the client and the server for its computation. The user receives the input data of ReLU, decrypts it and sends to the server the positivity or negativity of this input, so that the server can compute the output. Evaluation results showed that the solution could achieve accuracy close to model in plain-text, and similar to Cryptonet. As for efficiency, the solution provides a much lower time cost than Cryptonet. However, communication overhead needs to be evaluated especially that an interaction between the client and the server is required for every ReLU computation.

Vizitiu et al. [65] proposed a recent solution based on FHE encryption. The standard steps are followed, i.e., the input data is encrypted and then sent to the server for prediction. The approach uses a variant of the MORE scheme that can directly work on floating point values, this allows to practically perform privacy-preserving computations, without prior decryption or the introduction of polynomial approximations. However, the MORE encryption variant was criticized for its weaker security. Evaluation was conducted on a digit recognition problem (MNIST dataset) and a medical application. Results showed that the solution, in comparison to the unencrypted version, could achieve almost similar accuracy over MNIST and identical accuracy over the medical dataset. As for efficiency, it was noted that the encrypted solution increases runtime in comparison to unencrypted version although it is still faster than classic homomorphic encryption schemes. As for privacy, the authors pointed out that although the MORE scheme offers a certain degree of privacy, it is however vulnerable to chosen plaintext attacks.

Rouhani et al. [12], proposed a different technique based on Yao's Garbled Circuit (GC) protocol. The client garbles the boolean circuit of the DL architecture and sends computed garbled tables and input wire label to the cloud. Both client and cloud engage in an oblivious transfer protocol to obliviously transfer the wire labels associated with cloud inputs. The cloud evaluates the garbled circuit and computes encrypted data inference. Encrypted result is sent to the client, which decrypts it using the garbled keys. Besides, transforming input data to a lower-dimensional subspaces ensemble, and DL network distillation are two novel low-overhead preprocessing techniques proposed for optimizing computational and communication overhead of the GC, and reduce the overall runtime. In case of constraint embedded settings like wearables devices, secure mechanisms based on secret sharing to outsource GC protocol to a third party are proposed. Evaluation showed that data and network preprocessing allows to reduce computation and processing times per sample, with no accuracy drop. In comparison to [25], the overall execution time is improved even without preprocessing steps. However, the communication cost is relatively huge (722GB vs 336.7MB in [5]). As for privacy, there is no utility/privacy trade-off, and GC protocol is proven to be secure in honest-but-curious model. The privacy outside GC was demonstrated via formal proof. Besides, the solution is considered ideal for scenarios with small batches in order to have inference with minimal delay, while it is also well-suited to scenarios where data is collected over time and need to be dynamically analyzed without having to queue samples as batches.

Another interesting concept for privacy-preserving inference is the Secure Multi-party Computation (SMC), an interactive paradigm allowing multiple parties to collaboratively compute a function over their input data while keeping them private [62].

In [62], Huang et al. proposed a privacy-preserving CNN feature extractor. The user encrypts input data by randomly splitting it into two secret shares. Each edge server obtains the trained model and one of the data secret shares. A trusted third party generates random values for secure computations, and the edge servers run the CNN feature extractor through secure interaction protocols, to output the encrypted features and send them to the user. A series of secure protocols are proposed for the required operations to run the model collaboratively between the edge servers, including addition, multiplication, ...etc. On their basis, CNN layers adapted to secure computation (called building blocks) are designed to compose the secure CNN feature extractor. The designed building blocks can be used to support a full privacy-preserving inference. Noting that during interactive computations, biases are considered in only one of the edge servers, and set to 0 in the other. Theoretical analysis and empirical experiments of the solution were conducted. Results showed that the output can be recovered correctly as it preserves the additive property, and that the solution can be used with any CNN architecture without accuracy loss, as no approximation is required. Moreover, the client-side latency and network overhead are low, and the solution is secure in the honest-but-curious model.

Ma et al. [63] proposed a recent solution using non-colluding dual servers. Unlike [62], in the solution [63]

each server holds a share of the neural network, split by the network owner, including weights and bias matrices. The client requesting an inference encrypts its data and divides it into two secret shares distributed to the two non-colluding servers. Because of the homomorphic encryption, activation functions are approximated into piece-wise continuous low-degree polynomials. The servers interactively run the model and output the encrypted predictions, which are decrypted by the client to compute the final prediction. The authors presented a theoretical analysis showing that the solution satisfies model and data privacy. As for efficiency, the authors' analysis showed that their solution outperforms comparison solutions in terms of communication and computation complexity.

A last interesting concept for server-based deep learning analysis is data transformation, particularly data obfuscation techniques.

Leroux et al. [60] presented a solution in order to safely offload deep computations to the cloud on private local data. The approach uses Generative Adversarial Networks (GANs) but for the opposite of its original purpose. To that end, two auto-encoder-like networks, an obfuscator and a deobfuscator, are used along with the trained model. The obfuscator transforms the input to an obfuscated version to be fed into the classification network. The deobfuscator tries to recover the original input using the obfuscated version. Training the two new networks at the same time aims to release an obfuscator that produces a transformed image recognizable by the classifier but non-recoverable by the deobfuscator. As the aim is to offload costly computations to the cloud, the obfuscation network need to be as small as possible. In this context, MobileNet inspired architecture was used. Experimental results showed a ~5% accuracy loss between original and obfuscated classification, as well as an overhead due to the obfuscation network. As for privacy, tests showed that it was impossible to reconstruct original images after obfuscation. However, as back-propagation through the classification network is required, the network parameters should be available. To address this issue, the authors tried to train the obfuscator on one classifier and apply it on another. Results showed a large accuracy loss but not to a random level, which suggests more investigation. Besides, the classification network should be assumed static, which might be a problem in case of constant training like publicly available models [70]. However, the obfuscator could recover quickly if retrained with the latest classifier [70].

Raval et al. [61] recently proposed an obfuscation-based solution to prevent disclosing private information when offloading sensors personal data through mobile applications to the cloud. Similarly to [60], two networks are trained competitively, i.e., one to obfuscate input data against a second (the attacker) to undo the obfuscation. Unlike [60], the approach in [61] allows to specify the utility requirements through a parameter λ used in the privacy loss computation. Moreover, so that the obfuscation covers protection against a class of attackers like DNN, SVM, ...etc., the maximum privacy loss is minimized across all the attackers of this target class. The authors proposed to further extend the computation of the maximum privacy loss across all the private attributes to minimize the maximum expected leakage across all the private attributes. Experimentation was conducted on an android app for handwriting recognition, as well as on benchmark datasets. Results showed that the solution could reduce the risk of private information disclosure with a maximum accuracy of inferring private information less than 17%, and an accuracy with a maximum drop of 17%. However, for multiple sensors datasets (multimodal data), more investigation is required. As for efficiency, the obfuscation introduces a time overhead proportional to the input size, considered by the authors as acceptable for real-time processing.

b) Server-assisted PP analysis

Table 9. Reviewed solutions for PP Server-assisted Analysis

Reference *Target task*	**Key concept** Main technologies	**Adversary & Experimentation**
Osia et al. (2017/2018/2019) [39, 67, 68] *Inference task completed collaboratively between local device and cloud system (Hybrid architecture)*	**Model splitting** - DNN model splitting - *1st layer local* - Siamese architecture - Dimensionality reduction - *Principle Component Analysis (PCA) and auto-encoder features* - Multi-dimensional symmetric Gaussian noise	- IMDB and Labeled Face in the Wild (LFW) datasets - CNN - *two common deep models : VGG-16 and VGG-S_RGB based on VGG-S architecture*
Dong et al. (2017) [8] *Model provider to serve a user (inference)*	**Model splitting** - DNN model splitting - Dropping activation outputs - *Hadamard product* and Dropping connections - Dropout and Dropconnect	- MNIST, CIFAR10 (more challenging) datasets - DNN, CNN models

Chi et al. (2018) [66, 66'] *Inference phase of deep learning on a bipartite topology*	**Model splitting** - DNN model splitting *(Bipartite)* - Interactive adversarial deep networks	- Adversary has access to complete remote computing context - MNIST, LFW, CIFAR-10 datasets - Feed-forward NN (MNIST) / CNN (LFW, CIFAR-10)
Yu et al. (2019) [59] *Inference over a cloud-based deep learning model*	**Model splitting** - DNN model splitting - Step-wise activation functions	- Even in case weights are compromised by the adversary - MNIST, CIFAR-10 datasets - CNN model - *MobileNetV2*

Osia et al. [39, 67, 68] described a hybrid architecture for inference based on model splitting concept. The layers are split between local device and the cloud, and the process into feature extraction, performed by the local primary layers, and analysis, performed by secondary layers on the cloud. At training stage, fine tuning using Siamese architecture is performed in order to specialize the model for primary information and make sensitive information unpredictable. At inference stage, the local feature extractor outputs rich feature vector, on which dimensionality reduction is performed. Adding random multidimensional noise to features is also possible in order to increase privacy. The cloud analyzer takes the resulting features as input and performs analysis. Evaluation of the solution was conducted on image processing where gender and emotion are the primary tasks, and face recognition is the privacy one. Results showed that high accuracy could be achieved for primary tasks while decreasing potential sensitive inference. Moreover, using Siamese fine-tuning could make the trade-off between privacy and accuracy significantly better, while dimensionality reduction allows to improve privacy and highly reduces communication overhead, without accuracy loss. Privacy analysis of the feature extractor was performed using the three methods transfer learning, deep visualization and probabilistic modeling.

On the basis of the same concept, Dong et al. [8] proposed a technique where the first layer of the model including its activation function are migrated and applied on the local device. For the inference, the feed-forward propagation stage is performed as described in *Section 3.2.b.2*. Regarding evaluation, as presented in *Section 3.2.b.2*, 98.87% accuracy was achieved for non-invertible functions. More rectifying layers leads to better encryption but it could imply higher local computation. For invertible functions, the input data can be encrypted without noticeable degradation of performance, by only dropping a few activation outputs or adding small noise values to few activation outputs.

Another model splitting or bipartite based solution is proposed by Chi et al. [66] using interactive adversarial deep networks. The idea is make local output irreversible to prevent the recovery of input data. To that end, a defender network is introduced in order to simulate attackers. The training of the two networks is performed concurrently, where the model takes advantage of the recovery performance of the defender as a side information in order to optimize its parameters for a better privacy partitioning efficacy. The authors suggested to extend the defender to a defender suite in order to address different attacker models and provide more robust privacy. Besides, as only a portion of the model is locally-based, which is less-resource requiring, the solution can be deployed on mobiles devices and IoT devices. Experimentation showed that the approach hardens inputs recovery, as well as learning sensitive attributes from recovered inputs like emotion from face recognition. Besides, adding more defenders may allow to harden inputs recovery task, but leads also to more computing resources, rising therefore a trade-off. The number of local layers leads also to another trade-off between model privacy and data privacy.

A more recent solution based on model splitting is the previously described approach proposed by Yu et al. [59]. The first layer is migrated to the local side, where the private data is processed into an irreversible transformed data (called metadata), which is sent to the cloud for analysis (see *Section 3.2.b.2*). The solution was designed for training, and also tested on prediction. As previously presented, evaluation results showed that good accuracy could be achieved while creating considerable difficulties to recover original data. However, a trade-off between accuracy and privacy, controlled by interval value of step-wise activation function, is noted.

4.3 Discussion and learned lessons

We present a summary table of comparison (Table 10) of the privacy-preserving-analysis techniques that are reviewed in this section, then we discuss them, and identify a number of learned lessons and insights with respect to the key concepts.

Table 10. Comparison of reviewed PP analysis solutions

SE: Exclusively on external server(s)
SC: On both client and server(s)

TR : Transformation*
EN : Encryption
SA : Siamese architecture
MC : Multi-party computation
MS : Model splitting
AN : Adversarial networks
IO : Irreversible layer output**

ID: Input data
PR: Prediction

ECS : Extra communications*** required between clients and server(s)
AFI : At each activation function computation
THN: Every time the amount of homomorphic noise reaches a defined threshold

** Including obfuscation, randomization, ...etc. It can be applied on input data, features, ...etc.*
*** Including dropping connections, step-wise activation functions, ...etc.*
**** Apart from initialization, input, and end of training*

** Indicates the information that are protected*

Reference	DL's run location		Main privacy-enabling techniques							Privacy*		Additional server(s)	Training phase	ECS
	SE	SC	TR	EN	SA	MC	MS	AN	IO	ID	PR			
Xie et al. (2014) [37] Gilad-Bachrach et al. (2016) [25]	•			•						•	•		*Preferably train after network modifications*	
Baryalai et al. (2016) [17]	•			•						•	•	•		
Chabanne et al. (2017) [6]	•			•						•	•		*Train after network modifications*	
Hesamifard et al. (2017) [5, 38]	•			•						•	•		*Train after activation functions approximation*	*THN* •
Zhu et al. (2018) [64]	•			•						•	•			*AFI* •
Vizitiu et al. (2019) [65]	•			•						•	•		*Train over encrypted data*	
Rouhani et al. (2017) [12]	•			•						•		*Optional* •	*Retrain for data pre-processing and after network pre-processing*	
Huang et al. (2018) [62]	•			•		•				•	•	•		
Ma et al. (2019) [63]	•			•		•				•	•	•		
Leroux et al. (2018) [60]	•		•					•		•				
Raval et al. (2018) [61]	•		•					•		•				
Osia et al. (2017/2018/2019) [39, 67, 68]		•	•		•		•			•			*Fine tune the model using Siamese*	
Dong et al. (2017) [8]		•					•		•	•			*Apply Dropconnect and Dropout during training*	
Chi et al. (2018) [66, 66']		•					•	•		•			*Train the bipartite model concurrently with the defender*	
Yu et al. (2019) [59]		•					•		•	•			*Train using the step-wise approach*	

Four typical key concepts are used for privacy-preserving inference over a non-local deep model namely: model splitting, multi-party computation, data transformation, and encryption. Below, we discuss the privacy-preserving-learning techniques that are presented in this section, and we identify a number of learned lessons and insights.

Regarding encryption concept, although Homomorphic encryption (HE) is the mostly used technique compared to Garbled circuits (GC), the latter is also an interesting alternative to HE as it was reported to not imply privacy/utility trade-off, nor high computational overhead. However, it might need more exploration as it was noted in [5] that the approach based on GC and described in [12] has a relatively huge communication cost (722GB vs 336.7MB in [5] for the same network). As for homomorphic encryption, a number of notes and observations need to be pointed out :

- It was noted in [25] that modifications performed on the neural network in order to be compatible with homomorphic encryption (like polynomial approximations) should be preferably taken into consideration in the training stage, even if the latter is run over plain-text.

- Similarly to PP model learning, the choice of activation function with appropriate polynomial approximation might still need more investigation, as different results were obtained using different functions and approximations methods.
    - In [25], the lowest-degree non-linear polynomial function, which is the square activation function was used instead of approximating sigmoid and ReLU activation functions.
    - In [6], polynomial approximation was combined with batch normalization to obtain a good low-degree polynomial approximation of ReLU function. This combination seems to be interesting as it helps limiting the need of accurate approximation to a small part of R around the point 0. Besides, it is noted that batch normalization layers were also added to the CNN for training and classification in order to avoid high accuracy degradation between the two phases due to many modifications into the CNN.
    - [5] studied polynomial approximations of Sigmoid and Tanh functions over a symmetric interval using two different orthogonal system of polynomials, and proposed an approach based on the derivative of ReLU function. The latter proposition seems interesting to explore as it was reported that its behavior is robust against structure changes.
- In [25], it was proposed that max-pooling is replaced by scaled-mean pooling for easier computation over encrypted data, while in [6], average-pooling, considered as more FHE-friendly and having a small impact on accuracy, was proposed as a replacement to max-pooling.
- It might be interesting to explore the use of the approach proposed in [38] based on noise level threshold and collaboration between server and client, in order to address the problem of HE growing noise during computations.
- It was reported that feasibility of HE in resource-limited settings is bounded because of the high computational overhead [12] (particularly in FHE [5]), and the introduced noise reducing the inference accuracy [12, 23].
    - As stated in [5], LHE is faster than FHE and it can be improved using Single-Instruction-Multiple-Data (SIMD) techniques.
    - In [25], it was suggested in this context that more efficient encoding schemes with smaller parameters and faster homomorphic encryption should be sought.
- It might be interesting to investigate the non-colluding dual cloud model introduced in [17] where two clouds collaborate to produce the classification results in a secure way in order to address heavy computation overhead of HE.
- Investigating security improvements of the MORE scheme variant might be interesting as it can directly work on floating point values, and thus allows to practically perform privacy-preserving computations, without prior decryption or the introduction of polynomial approximations [65].

As for model splitting concept, dropping few activation outputs at local side can protect input data without noticeable performance decrease [8]. Besides, using step-wise activation functions [59] and exploring other similar methods in order to make data fed into cloud side irreversible, is an interesting path to further investigate.

Besides, the following general observations can also be made :

- Data and neural network preprocessing methods introduced in [12] are interesting to be explored as they allowed reducing computation and processing time per sample, without accuracy drop.
- Experiments showed that performing dimensionality reduction with Principle Component Analysis (PCA) or auto-encoder on intermediate features, allows to improve privacy and highly reduce communication overhead without reducing accuracy [39].

Lastly, the possibility of combining multiple key concepts and their underlying technologies might be worth to be explored in order to inherit their advantages such as data and network preprocessing methods.

5. Releasing a privacy-preserving (PP) model

Sharing a trained model, either as a service (black-box) or as a model with its internals (white-box), exposes its original training data to leakage risks. Membership inference [18] and model inversion [19] are examples of popular attacks used in this context. In this section, we present the classification that is related to the release of a privacy-preserving model. Then, we give a summary description of the relevant techniques, and finally we discuss them and derive some learned lessons.

5.1 Classification

According to the differential privacy mechanism, reviewed existing techniques for releasing a PP model can be categorized into three key concepts namely : (a) differentially private model parameters, (b) differentially private input data, and (c) differentially private mimic learning (see the below Figure 13).

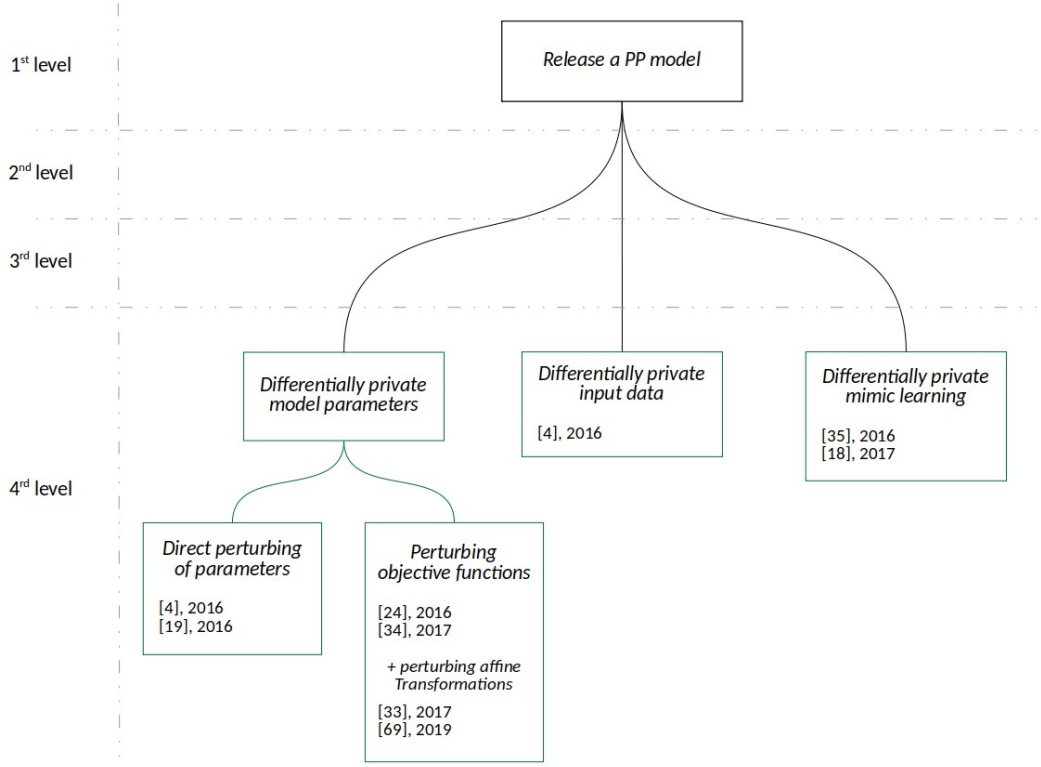

Fig. 13. Taxonomy of Releasing a PP model techniques

5.2 Solutions review

Based on the classification presented in 5.1, reviewed solutions (see Tables 12, 13, and 14) are further described below along with their main evaluation results according to the performance metrics (see the below Table 11).

Table 11. Main measures and properties to evaluate techniques to release a PP model

Performance metrics	Measures and properties
Effectiveness	Accuracy, F-measure
Efficiency	Independence of privacy budget consumption to number of training epochs, Extendability/applicability to different deep learning models/activation functions
Privacy	Privacy budget ε value, Privacy parameters (ε, δ) values, Robustness to privacy budget change, (Indirect leakage) Protection of potential sources (parameters, ...)

a) Differentially private model parameters

In order to come up with differential private model parameters that protect original training data, two main ways are possible, namely (i) directly perturbing model parameters [4,19], and (ii)perturbing objective functions instead of results [24, 34], which might also be coupled with affine transformations perturbation [33, 69].

Table 12. Reviewed solutions for Releasing a PP model through differential private model parameters

Reference *Target task*	**Key concept** Main technologies	**Adversary & Experimentation**
Sei et al. (2016) [4] *Share a trained model (DNN parameters) with other organizations*	**Differentially private model parameters : *Direct perturbing model parameters*** - Differential privacy - *Laplace mechanism*	- ADULT, Salary estimation datasets - DNN model Comparison : (Baseline method) *adding Laplace random variable to each of the trained model parameters*
Abadi et al. (2016) [19] *Training neural networks within a modest ("single-digit") privacy budget*	**Differentially private model parameters : *Direct perturbing model parameters*** - Differential privacy – *Normal (Gaussian) distribution noise*	- MNIST, CIFAR-10 datasets - DNN model Comparison : Non-private baseline model designed and trained by the authors
Phan et al. (2016) [24] *Create a privacy-preserving deep auto-encoder with focus on binomial classification and prediction tasks*	**Differentially private model parameters : *perturbing objective functions*** - Polynomial approximation - *Taylor expansion* - Differential privacy - *Functional mechanism*	- Real health social network, Human behavior prediction datasets - Deep Auto-Encoder model Comparison : Models non enforcing ε-differential privacy , methods for regression analysis under ε-differential privacy
Phan et al. (2017) [34] *Release a CDBN model under privacy protection - Technique to be applied on typical energy-based deep neural networks*	**Differentially private model parameters : *perturbing objective functions*** - Differential privacy – *Functional mechanism* - Polynomial approximation – *Chebyshev expansion, Taylor expansion*	- YesiWell, MNIST datasets - CDBN model Comparison : Prediction: DL models for non enforcing ε-differential privacy, [24] / Classification : state-of-the-art polynomial approximation approaches / [19]
Phan et al. (2017) [33] *Release a privacy-preserving deep neural network model - Should be applicable in different deep neural networks with different activation functions*	**Differentially private model parameters : *perturbing objective functions and affine transformations*** - Differential privacy – *Laplace mechanism adapted to feature relevance* - Layer-wise Relevance Propagation - Polynomial approximation - *Taylor Expansion*	- MNIST, CIFAR-10 datasets - CNN model Comparison : [19]
Adesuyi et al. (2019) [69] *Deliver deep neural network models preserving privacy of personal data*	**Differentially private model parameters : *perturbing objective functions and affine transformations*** - Differential privacy – *Laplace mechanism* - Layer-wise Relevance Propagation - Polynomial approximation - *Maclaurin series*	- Winsconsin Diagnosis Breast Cancer (WDBC) dataset - DNN model Comparison : State-of-the-art classification for WDBC, and non privacy-preserving DNN

Sei et al. [4] proposed three approaches in order to create a privacy-preserved DNN. Two of them named "Learningfirst" and "AnonymizedLearning" are related to direct perturbation of model parameters. In the first approach, the model is trained on the original dataset, then anonymized by adding Laplace noise to its parameters (both weights and bias). Besides, the global sensitivity is also reduced by adding limitations to the model parameters. In the second approach, each value of the model parameters is anonymized in the training stage. The evaluation results showed that the two approaches could achieve high estimation accuracy. For a small privacy budget, accuracy and f-measure of "AnonymizedLearning" approach outperform "LearningFirst", while for a large privacy budget, "LearningFirst" outperforms "AnonymizedLearning". Regarding privacy, using differential privacy ensures that the risk of sensitive information leakage is smaller than the predefined threshold. Authors reported that their approaches outperform the baseline method for most values of privacy budget ε.

Abadi et al. [19] proposed a differential private stochastic gradient descent (SGD) that controls training data influence during the training stage. At each step of the SGD, the gradient of a random subset of examples (called a lot) is computed, and l2 norm of each gradient is clipped. The average of gradients is then computed, and noise is added. Finally, the model is output and the overall privacy loss is computed by the privacy accountant, which accumulates cost at each access to the training data. Besides, tuning the hyper-parameters of the model is proposed in order to balance privacy, accuracy and performance. Its was observed that the model accuracy is less sensitive to the network structure, than the training parameters like noise level and batch size. Evaluation results showed that only a modest total privacy loss is incurred for training the network with (ε, δ)-differential privacy. Accuracy could reach 97% which is 1.3%-less compared to the non-private baseline over MNIST, but for CIFAR-10, the difference was much larger (about 7%). Besides, introducing the moments accountant allows tight automated analysis of the privacy loss. However, adding noise into gradients at every training step leads to dependence on the number of training epochs [33, 34], which needs to be large in order to guarantee accuracy. However, only a small number can be used for a small privacy budget.

Phan et al. [24] proposed a privacy-preserving Deep Auto-encoder, where the objective functions are

perturbed instead of results in order to obtain perturbed parameters and enforce ε-differential privacy. The data reconstruction function is approximated using Taylor expansion then perturbed using Functional mechanism by injecting Laplace noise into its coefficients. After that, the model parameter that minimizes the perturbed function is derived. Gradient descent is used to obtain optimal perturbed parameters by training the perturbed model, resulting in a private auto-encoder (PA). Deep private auto-encoder (dPA) is obtained by stacking multiple instances of PAs where a normalization layer is added on top of the hidden layer of each PA. Moreover, cross-entropy in the softmax layer for classification or prediction is approximated and perturbed using Functional mechanism and back-propagation algorithm is lastly used in order to fine-tune all parameters of the dPA. Evaluation results showed that dPA achieved accurate results, and outperformed comparison solutions. The privacy is preserved through enforcing ε-differential privacy at every layer and training step, and dPA was shown to be robust against privacy budget change. Besides, the authors claimed that their solution could be extended to other deep models, but in [41], it was criticized for lacking the ability for generalization and that no meaningful privacy guarantee is provided. The solution was also considered to be designed for a specific model [33, 34].

Based on the same idea of perturbing objective functions, Phan et al. [34] described another approach for releasing a private Convolutional Deep Belief Network (CDBN). The key idea is the use of Chebyshev expansion for deriving polynomial approximations of non-linear objective functions. In order to satisfy ε-differential privacy in training stages, approximation function is perturbed by injecting Laplace noise into its coefficients. The model is then trained using gradient descent in order to obtain optimal perturbed parameters, which results into private hidden layers. The private CDBN (pCDBN) is consequently obtained by stacking multiple pairs of private hidden layers, and a max-pooling layer. In addition, an output layer is added on top of the pCDBN, and differential privacy is enforced in softmax in the same way as in [24]. The evaluation of accuracy on different settings showed that the pCDBN could outperform all comparison techniques including [19] and [24] in almost all cases and under different dataset cardinalities and different privacy budget values. Privacy is ensured through satisfying ε-differential privacy and using Chebyshev approximation, incurring fewer errors than Taylor and Piecewise approximations, and being more effective in preserving differential privacy in CDBNs. It was shown that pCDBN is robust against privacy budget change, and that large datasets are supported since the budget in pCDBN, unlike [19], is independent from the number of training epochs.

A different approach combining perturbation of objective functions as well as affine transformations of neurons was proposed by Phan et al. [33]. Differentially private relevances of input features are first obtained by injecting Laplace noise into the average relevances computed using Layer-wise Relevance Propagation (LRP) algorithm. LRP can be applied on a simple but accurate pre-trained model. Perturbed affine transformation layer is obtained by injecting adaptive Laplace noise into affine transformation of each of its hidden neurons. The features that are less relevant are injected more noise. The private DNN is then built by stacking hidden layers on differentially private hidden layer where a normalization layer is applied before each stacking for bounding non linear activation functions. Lastly, the labels at the output layer are protected through polynomially approximating loss function and perturbing it by injecting Laplace noise to its coefficients. Evaluation results showed that the solution outperforms [19] for both small and large privacy budget. Although differential privacy is enforced, it was reported in [42] that data after applying noise remains recognizable for human. As for efficiency, even if adding noise adaptively signifies an extra computational overhead requirement [42], evaluation however, showed that average relevances computation is efficient as it only needs 12 extra epochs, and that computational efficiency is also not much affected by noise injection computations. The solution is also reported to be applicable on different deep learning networks with different activation functions, and allows the support of large datasets as privacy budget consumption is independent from number of training epochs.

Adesuyi et al. [69] recently proposed a solution using the same techniques as [33]. The idea is to combine differential privacy with LRP with the aim to strengthen accuracy. LRP is used in order to classify the relevance of neurons into high or low. Laplace noise is then injected to neurons according to their relevance category using corresponding privacy budget. The smaller is the privacy budget, the larger is the produced noise. Moreover, the target value is perturbed at every batch via the loss function, so that each point of data access is protected to offer a reliable privacy preserving model. For this, the loss function is polynomially approximated using Maclaurin series and perturbed using Laplace mechanism. Evaluation results of the solution showed an achieved accuracy close to the non-privacy-preserving version even with large noise injection. The worst case, with heavily dense noise, presented a loss of less than 5% of accuracy. Besides, the

authors pointed out that as neuron relevancy is not considered in the loss function perturbation, the total budget is calculated only from the two privacy budgets corresponding to high and low relevance categories.

b) Differentially private input data

Table 13. Reviewed solutions for Releasing a PP model through differential private input data

Reference *Target task*	**Key concept** Main technologies	**Adversary & Experimentation**
Sei et al. (2016) [4] *Share a trained model (DNN parameters) with other organizations*	**Differentially private model parameters :** ***perturbing model parameters*** - Differential privacy - *Laplace mechanism*	- ADULT, Salary estimation datasets - DNN model Comparison : (Baseline method) *adding Laplace random variable to each of the trained model parameters*

A third method proposed by Sei et al. [4] named "AnonymizingFirst" consists of anonymizing the original dataset so that it satisfies ε-differential privacy, then applying the model on resulting anonymized data. The authors further proposed using a literature technique to mitigate noise impact through adding noise to each set of records, instead of each record. Evaluation results showed that the proposed approach could achieve high estimation accuracy. For a small privacy budget, accuracy and f-measure of "AnonymizingFirst" is higher than the previously described "LearningFirst" approach, while for a large privacy budget, "LearningFirst" outperforms both "AnonymizingFirst" and "AnonymizedLearning" previously described. As for privacy, using differential privacy ensures that the risk of sensitive information leakage is smaller than a predefined threshold.

c) Differentially private mimic learning

Table 14. Reviewed solutions for Releasing a PP model through differentially private mimic learning

Reference *Target task*	**Key concept** Main technologies	**Adversary & Experimentation**
Papernot et al. (2016) [35] *Train a student model (released model) using aggregated predictions of teacher models trained on disjoint data*	**Differentially private mimic learning** - Mimic learning - *Teacher/student* - Differential privacy – *Laplace mechanism* - Semi-supervised learning with GANs	- MNIST, SVHN datasets - CNN model Comparison : [3], [19]
Dehghani et al. (2017) [18] *Learn a student model (released model) using aggregated predictions of privacy-preserving trained teacher models, instead of original sensitive training data*	**Differentially private mimic learning** - Mimic learning - *Teacher/student* - Differential privacy – *Laplace mechanism*	- Experimentation on document re-ranking *as one of the core Information Retrieval tasks*

Mimic learning (teacher/student model) concept introduced in [35] and [18] allows to protect original training data after releasing a deep model. The general idea of mimic learning is to train a first efficient model, called teacher model, on the original training data. Then the teacher annotates a large unlabeled dataset, which will be used to train another model called student model. In case of many applications, the student model was able to make similar predictions as the teacher with similar or better performance [18].

Papernot et al. [35] proposed an approach with no assumption on model details, or constraints for data partitioning or selection. A group of teachers are first trained on different subsets of original sensitive data, then used as an ensemble to annotate unlabeled non-sensitive data. The student, will then learn to accurately mimic the teachers ensemble. To ensure privacy during annotation, the teachers deployed as an ensemble aggregate their distinct predictions into a single prediction, and Laplace noise is added in vote counts to introduce ambiguity. This noisy aggregation-based annotation of student's training data, implies that its training is determined by the number of queries to teachers. This leads to a trade-off with student's model quality. This trade-off is addressed using semi-supervised knowledge transfer described as the most successful technique among others, and which allows to reduce privacy budget. The solution achieved good results in terms of (ε, δ)-differential privacy bound and accuracy of students compared to [3] and [19], as well as in terms of accuracy compared to a model trained with the entire dataset. The private student model also achieved better accuracy than aggregation's output with and without noise. However, the student may not learn as good from data where categories are not designed to be salient in the input space (eg. medical data). As for privacy, noisy aggregation with limited teachers vote allow to protect training data even if the internal parameters of the student model are observed by an attacker. Besides, larger ensembles of teachers allow potentially larger noise and stronger privacy but implies smaller training datasets and thus less accuracy. Appropriate value of number of teachers might therefore need to be found empirically.

Based on the same idea as [35], Dehghani et al. [18] proposed a solution to share a privacy-preserving model

instead of sensitive data in Information Retrieval (IR) applications. Teachers are trained individually using subsets of sensitive training data, and Laplace noise is added to each teacher's prediction, then aggregation is performed using majority voting to obtain a single global prediction. Privacy-preserving transfer knowledge from teacher to student can be thus performed by annotating unlabeled public data using aggregated teacher. The solution was experimented on document re-ranking (one of the core Information Retrieval tasks), and achieved acceptable performance with low privacy risk guarantee. Besides, it was pointed out that the teacher/ student mimic learning concept can be used to overcome the lack of large datasets in IR task.

5.3 Discussion and learned lessons

We present a summary table of comparison (Table 15) of the techniques for releasing a privacy-preserving model that are reviewed in this section, then we discuss them, and identify a number of learned lessons and insights with respect to the key concepts.

Table 15. Comparison of reviewed solutions to release a PP model

MP : Model parameters
OF : Objective functions
AT : Affine transformations
TD : Training data
AP : Aggregated predictions of ensemble teachers
VC : Vote counts of teachers
TP: Teachers predictions

** Indicates the main information to be prevented from leaking information related to training data*

Reference	Main privacy-enabling techniques		Perturbation - *differential privacy*						Information protected*		
	Differential privacy	Mimic learning	MP	OF	AT	TD	VC	TP	TD	MP	AP
Sei et al. (2016) [4]	•		•							•	
						•			•		
Abadi et al. (2016) [19]	•		•							•	
Phan et al. (2016) [24]	•			•						•	
Phan et al. (2017) [34]	•			•						•	
Phan et al. (2017) [33]	•			•	•					•	
Adesuyi et al. (2019) [69]	•			•	•					•	
Papernot et al. (2016) [35]	•	•					•				•
Dehghani et al. (2017) [18]	•	•						•			•

Three typical key concepts are used for privacy-preserving release of a deep model, namely: differential private input data, differential private mimic learning, and differential private model parameters. Below, we discuss the privacy-preserving-learning techniques that are presented in this section, and we identify a number of learned lessons and insights.

Regarding differential private model parameters :

- Adaptive redistribution of noise [33, 69] is an interesting technique for accuracy improvement. In [33], better accuracies were achieved in comparison to [19] for both small and large privacy budgets. This was achieved by using noise redistribution on the basis of features relevance to the model output.
- Independence of privacy budget consumption from the number of training epochs is an important property allowing to keep training without accumulating privacy budget as to guarantee model accuracy in practice and to work with large-scale datasets [33, 34].
- It is noted in [34] that Chebyshev approximation incurs fewer errors than Taylor and Piecewise approximations, and is more effective in preserving differential privacy in CDBNs.

Regarding differential private mimic learning :

- Using mimic learning allows to (1) protect training data even if attackers observe internal parameters of student's model as described in [35], and (2) overcome the lack of large datasets as noted in [18] for

ad-hoc IR task, .

- It is described in [35] how the appropriate number of teachers in mimic learning could be found empirically, as a larger number could imply stronger privacy (due to potentially larger noise) but potentially less accuracy (due to smaller training datasets).
- Semi-supervised knowledge transfer is described in [35] as the most successful technique to address the trade-off between privacy budget and accuracy of the student controlled by the number of queries to teachers for labeling its training dataset.

Besides, it should be noted that introducing moments accountant like in [19, 35, 43] might be useful as it allows tight automated analysis of the privacy loss.

6. Challenges and future directions

Although various solutions were proposed in the literature in order to ensure privacy preservation in deep learning, a certain number of challenging concerns might still deserve to be more explored and investigated whether for privacy-preserving model learning, analysis, or release of a model tasks. Below, we present a summary of the main existing challenges, with respect to the privacy-preserving tasks and performance metrics, and identify some solutions derived from reviewed techniques. We conclude this section with some recommendations and promising future directions.

6.1 PP model learning

a) Effectiveness. Keeping accuracy at an acceptable level seems to be the main challenge to meet when designing a privacy-preserving model learning solution, as applying some techniques might reduce model's accuracy such as activation functions approximation or partial sharing. In this context, the different directions taken by reviewed solutions, such as controlling fraction of gradients in partial sharing or introducing ϱ-visibility to enhance differential privacy, or accuracy score calculation used to reduce impact of low-quality data, are worth to be more investigated separately or in combination given the reported promising results, either to maintain accuracy or mitigate its lost, mostly through an accuracy/privacy trade-off.

b) Efficiency. Training and running times, as well as computation and communication costs are of the main concerns to be considered. Most reviewed works that are based on homomorphic encryption result in increased costs overhead, which suggests that efficiency of homomorphic encryption needs more investigation. The selection and exploration of appropriate encryption schemes like symmetric additively homomorphic, or the possible involvement in combination with other techniques as partial sharing are examples of potential directions to take. However, it is required to study the advantages and drawbacks of each introduced technique and direction taken to build an efficient solution. Employing partial sharing for example is an interesting technique for reducing network traffic. However, it should be kept in mind that partial sharing can make the global optimal more difficult to achieve as data is not fully utilized. Other concerns related to efficiency were also successfully addressed by some reviewed works, particularly, dealing with unreliable networks and saving energy transmission, which are important in case of applications where mobility is a requirement.

c) Privacy. Protection against direct leakage, i.e., privacy of training data, as well as indirect leakage, i.e., privacy of parameters are of the main challenges to be addressed. For techniques where training data is exposed (direct leakage), homomorphic encryption seems to be successful in protecting private data during both collaborative and individual learning process. However, potential efficiency-related issues as described above should be handled. A solution based on combination of data perturbation and projection seems promising and techniques used might be worth to be investigated with other technologies. In contrast, most reviewed solutions not revealing training data (indirect leakage) shift the concern to parameters protection being the element exposed. In this case, a number of techniques were reported as fairly successful such as partial sharing combined with differential privacy or enhancing local differential privacy with ϱ-visibility for gradients protection. However, for some of these methods, more investigation might be useful for better privacy budget consumption and efficiency like considering appropriate trade-off with accuracy. Data masking-based solution for collaborative learning seems interesting but privacy preservation was only evaluated theoretically. On the other hand, other solutions for individual learning based on model splitting with data augmentation or irreversiblity transformation techniques were also reported as fairly successful in preserving privacy of training data. However, model splitting concept requires the deployment of a part of the deep model on user device, while shared model requires the availability of voluntary shared data and local retraining for specializing the

model for user.

6.2 PP analysis

a) Effectiveness. Similarly to deep model training, achieving acceptable accuracy is the main challenge when designing deep-learning-based analysis solutions with privacy-preservation. Homomorphic encryption with polynomial approximation of activation function, are from the most used concepts with good reported evaluation results. Reviewed solutions took different directions to achieve good model utility results such as the combination of polynomial approximation with batch normalization, the selection of appropriate polynomial approximation and search of a trade-off between its degree and model's performance, or the use of scheme variants which do not require network modifications, but at the expense of a certain degree of privacy. On the other hand, the use of Yao's garbled circuit or model splitting concept with Siamese architecture as alternatives to homomorphic encryption seems worth to be more explored given the reported good results. However, other aspects of the above techniques should be more investigated such as efficiency for Yao's garbled circuit or deploying part of the model on the user side.

b) Efficiency. Communication and computation costs are the most faced challenges. Employing a non-colluding dual cloud model was proposed to address heavy computation overhead of homomorphic encryption while a reviewed work suggested that more efficient encoding schemes with smaller parameters, and faster homomorphic computation should be sought. For garbled circuit, some data and network preprocessing techniques was proposed to reduce computation and processing times. Besides, huge communication cost was noted for garbled circuit and needs to be investigated.

c) Privacy. Main concerns addressed refer to the protection of users private data and predictions while sometimes trained model is also considered. Homomorphic encryption based solutions seem successful in protecting input, inference process and predictions through encrypting the overall, whereas garbled circuit is proven to be secure in honest-but-curious model. However, privacy outside of the garbled circuit protocol was only proved formally. On the other hand, removing undesired sensitive information from extracted features in model-splitting solution seems to be successful for achieving privacy of users data, while methods for irrevesibility transformation of local-side ouput like step-wise functions deserves more investigation. However, the method depends on deploying part of the model on the user side although this splitting makes the solution desirable by cloud providers since users do not access to the complete model.

6.3 Release a PP model

a) Effectiveness. Accuracy of the released privacy-preserving model is the main concern to consider. Although differential privacy involves introducing noise, reviewed solutions using this technology could achieve acceptable model's accuracy. However, as high accuracy can be a mandatory requirement in some applications, exploring combining differential privacy with other improvement techniques in order to increase accuracy without compromising privacy might be necessary. Adaptive redistribution of noise based on relevance is a good example in this context.

b) Efficiency. Ensuring that privacy budget consumption is independent from number of training epochs is an important factor to consider allowing to work with large datasets. In this context, it is not advised for example to add noise into gradients in every training step.

c) Privacy. Protecting training data when releasing a privacy-preserving deep model is the main challenge to address. To that end, model's parameters, which are most potential sources of leakage, should be protected, particularly using differential privacy. This was shown to be fairly successful either through perturbing parameters, objective functions or affine transformations. In this context, when it comes to objective functions, it is noted that the selection of appropriate polynomial approximation is an important factor to consider as it was reported that Chebyshev approximation could be more effective in preserving differential privacy. Besides, mimic learning is an interesting concept as teachers models trained on original training data are never directly exposed to end-users. In addition, mimic learning can be combined with other techniques in order to further strengthen privacy guarantees like introducing differential privacy, limiting the number of teachers votes, or revealing only topmost noisy vote for student training. On the other hand, the introduction of moments accountant was shown to be worth for allowing tight automated analysis of the privacy loss.

6.4 Future directions

In this section, we present some recommendations and paths to explore as future research directions in the

design of efficient privacy-preserving deep learning solutions.

An interesting starting point, towards future designs of the solutions, would be by taking benefits from the state-of-the-art research and exploring the different learned lessons derived from reviewed solutions. This includes investigating potential combinations between different concepts as well as their underlying technologies as mentioned previously (see sections 3.3, 4.3 and 5.3, *Discussion and learned lessons*). A performance study in terms of effectiveness, efficiency and privacy should be conducted for potentially promising combinations.

On the other hand, we think that coming up with a real-world solution requires addressing privacy concerns encountered as a whole issue from their different aspects, which might need a combination of different PP solutions classes. For example, designing a solution allowing to collaboratively train a public cloud model in practice, might require not only to protect the training data of participants from the cloud and participants during the learning process, but also to prevent the resulting trained model from leaking sensitive information to future clients. Such a solution covers therefore both PP model learning and release a PP model classes.

Moreover, although various efforts were deployed in order to tackle privacy preservation in deep learning either for training, using or sharing a model, generalizing a solution or preferring one solution over another in an absolute way might not be the right direction in our opinion. In fact, the design of a successful solution that achieves together appropriate effectiveness, efficiency, and privacy depends on different constraints and requirements such as communication costs, time constraints or available resources for typical users devices to perform encryption/decryption for instance or to deploy a local feature extractor provided that user consents [9]. However, it is interesting to design flexible solutions that allow a certain trade-off involving effectiveness, efficiency, privacy, and ability to address constraints and requirements of different applications.

Besides, promising future work in the privacy-preserving deep learning approach, particularly in the context of collaborative or federated model learning, might also involve further exploration of the Blockchain technology, one of the emerging technologies nowadays. Blockchain, initially used in the financial industry, is basically a peer-to-peer distributed ledger technology or database represented by a series of data blocks secured and linked using cryptography [44-46]. In the literature, some Blockchain-related works addressing privacy-preservation in deep learning [44-48] have already been conducted. Jiasi Weng et al. [47] for example presented "Deepchain" a secure framework for distributed deep learning introducing a value-driven incentive mechanism based on Blockchain in order to address malicious adversaries as well as the lack of incentives for distrustful participants. In a more recent work, Lifeng Liu et al. [46] proposed a Blockchain-based federated model learning paradigm in order to address privacy of model and high computation cost of data owner. Such recent works in this context shows that research in the domain is still topical.

From a hardware point of view, a recent study [49] proposed the use of trusted hardware for privacy preserving deep neural network predictions. The study concluded that although using trusted hardware could make privacy preserving machine learning more viable for real-world applications, absolute confidentiality still cannot be guaranteed. However, it was pointed out that such a issue was essentially due to implementation limitations which paves the way for further exploration of this direction. Besides, novel processors technology specialized in the acceleration of machine learning algorithms, i.e., Neural processor or Neural Processing Unit (NPU)[(1)] such as TPU (Tensor Processor Unit) from Google or NVDLA (Nvidia Deep Learning Accelerator) might be helpful towards a better support of privacy techniques requiring local neural network processing or even the design of more efficient ones. In addition, investigating how to use NPUs for non-neural-network processing might be an interesting path to explore and compare with other powerful processors like GPUs in the context of privacy techniques having high local resources requirements like encryption-based ones.

Lastly, in order to address computing and communication costs on the client side, exploring compression methods for neural networks might be an interesting future path to follow given reported promising results [50]. Song Han et al. [51] for example showed that through a three stage pipeline including pruning, trained quantization and Huffman coding, storage requirement of neural networks could be reduced by 35x to 49x without affecting their accuracy.

(1) https://en.wikichip.org/wiki/neural_processor

7. Conclusion

In this paper, we have provided a review of privacy-preservation deep learning solutions, along with a set of learned lessons, challenges, and future recommendations.

We have proposed a novel multi-level taxonomy, which categorizes the current state-of-the-art privacy-preserving deep learning techniques on the basis of privacy-preserving tasks at the top level, and key technological concepts at the base level. The top level privacy-preserving tasks involve three main privacy concerns essentially related to input data, namely: (1) *PP (Privacy-Preserving) model training or learning*, (2) *PP inference or analysis*, and (3) *release a PP model*. At the base level, encryption and more particularly homomorphic encryption (often supported by polynomial approximation) is the most used concept for for PP model learning and PP inference, while for releasing a PP model, differentially privacy particularly that of model parameters is the major key concept used.

We have also identified performance metrics to evaluate the solutions evaluation, which are grouped into three main metrics namely: (1) *effectiveness*, popularly evaluated through accuracy, (2) *efficiency*, mainly including communication/computational overhead and execution time, and (3) *privacy*, mainly evaluated through direct and indirect leakage protection guarantees, and privacy budget consumption for differential privacy.

This work is a ~~in depth~~ privacy-preserving deep learning review that summarizes state-of-the-art solutions along with their evaluations results with respect to the identified performance metrics, ~~discusses their advantages and drawbacks,~~ and derives a set of learned lessons from each privacy-preserving task. In addition, research challenges are highlighted with respect to performance metrics and privacy-preserving tasks.

For the design of future solutions, it is advisable to take into consideration the presented learned lessons and investigate further the combination of the different concepts and reported results of reviewed techniques. In fact, the design of a successful solution that achieves together appropriate effectiveness, efficiency, and privacy should consider different constraints and requirements. Moreover, coming up with a real-world solution requires addressing privacy concerns encountered as a whole issue from their different aspects, which might need a combination of different PP solutions classes. Lastly, emerging techniques and technologies like Blockchain, neural processors, and their potential opportunities, might represent some of the promising future paths to explore.

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] F. Bu, Y. Ma, Z. Chen, and H. Xu, "Privacy Preserving Back-Propagation Based on BGV on Cloud," *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, Aug. 2015.

[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct. 2015.

[4] Y. Sei, H. Okumura, and A. Ohsuga, "Privacy-Preserving Publication of Deep Neural Networks," *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2016.

[5] E. Hesamifard, H. Takabi, and M. Ghasemi. "CryptoDL: Deep Neural Networks over Encrypted Data," *arXiv preprint arXiv:1711.05189v1,* 2017.

[6] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptology ePrint Archive (035)*, 2017.

[7] T. Zhu, G. Li, W. Zhou, S. Y. Philip, "Differentially Private Deep Learning," In *Differential Privacy and Applications,* pp. 67-82, Springer, Cham, 2017.

[8] H. Dong, C. Wu, Z. Wei, and Y. Guo, "Dropping Activation Outputs With Localized First-Layer Deep Network for Enhancing User Privacy and Data Security," *arXiv preprint arXiv:1711.07520v1,* 2017.

[9] D. Zhang, X. Chen, D. Wang, and J. Shi, "A Survey on Collaborative Deep Learning and Privacy-Preserving," *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 2018.

[10] S. Chang and C. Li, "Privacy in Neural Network Learning: Threats and Countermeasures," *IEEE Network*, vol. 32, no. 4, pp. 61–67, 2018.

[11] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable Provably-Secure Deep Learning," *arXiv preprint arXiv:1705.08963,* 2017.

[13] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[14] D. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in bioinformatics*, vol. 18, no. 5, pp. 851-869, 2017.

[15] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2017.

[16] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[17] M. Baryalai, J. Jang-Jaccard, and D. Liu, "Towards privacy-preserving classification in neural networks," *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016.

[18] M. Dehghani, H. Azarbonyad, J. Kamps, and M. de Rijke, "Share your model instead of your data: Privacy preserving mimic learning for ranking," *arXiv preprint arXiv:1707.07605v1,* 2017.

[19] M. Abadi, A. Chu, I. Goodfellow, H. B. Mcmahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," *arXiv preprint arXiv:1607.00133v2,* 2016.

[20] L. Lyu, X. He, Y. W. Law, and M. Palaniswami, "Privacy-Preserving Collaborative Deep Learning with Application to Human Activity Recognition," *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM 17*, 2017.

[21] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning: Revisited and Enhanced," *Applications and Techniques in Information Security Communications in Computer and Information Science*, pp. 100–110, 2017.

[22] M. Liu, H. Jiang, J. Chen, A. Badokhon, X. Wei, and M.-C. Huang, "A Collaborative Privacy-Preserving Deep Learning System in Distributed Mobile Environment," *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2016.

[23] P. Li, J. Li, Z. Huang, T. Li, C. Z. Gao, S. M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems* 74 : 76-85, 2017.

[24] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential Privacy Preservation for Deep Auto-Encoders: an Application of Human Behavior Prediction," *AAAI.* Vol. 16, Feb. 2016.

[25] R. Gilad-Bachrach, N. Dowlin, K. Laine, L. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," *International Conference on Machine Learning,* Jun. 2016.

[26] Y. A. Le Trieu Phong, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption", *IACR Cryptology ePrint Archive (715)*, 2017.

[27] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[28] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, Yet Practical, Multiparty Deep Learning," *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[29] 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," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS 17*, 2017.

[30] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy Preserving Deep Computation Model on Cloud for Big Data Feature Learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, Jan. 2016.

[31] M. Li, L. Lai, N. Suda, V. Chandra, and D. Z. Pan, "PrivyNet: A Flexible Framework for Privacy-Preserving Deep Neural Network Training," *arXiv preprint arXiv:1709.06161v2,* 2017.

[32] S. Servia-Rodriguez, L. Wang, J. R. Zhao, R. Mortier, and H. Haddadi, "Personal Model Training under Privacy Constraints," *arXiv preprint arXiv:1703.00380v2,* 2017.

[33] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning," *arXiv preprint arXiv:1709.05750v1*, 2017.

[34] N. Phan, X. Wu, and D. Dou, "Preserving differential privacy in convolutional deep belief networks," *Machine Learning*, vol. 106, no. 9-10, pp. 1681–1704, 2017.

[35] 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:1610.05755v4,* 2016.

[36] S. Sharma, and K. Chen, "Image Disguising for Privacy-preserving Deep Learning," *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018.

[37] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," *arXiv preprint arXiv:1412.6181v2,* 2014.

[38] E. Hesamifard, H. Takabi, M. Ghasemi, and C. Jones, "Privacy-preserving Machine Learning in Cloud," *Proceedings of the 2017 on Cloud Computing Security Workshop - CCSW 17*, 2017.

[39] S. A. Osia, A. S. Shamsabadi, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi, "Privacy-Preserving Deep Inference for Rich User Data on The Cloud," *arXiv preprint arXiv:1710.01727v3,* 2017.

[40] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[41] M. A. Rahman, T. Rahman, R. Laganiere, N. Mohammed, and Y. Wang, "Membership Inference Attack against Differentially Private Deep Learning Model," *Transactions on Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018.

[42] J. Shen, J. Liu, Y. Chen, and H. Li, "Morphed Learning: Towards Privacy-Preserving for Deep Learning Based Applications," *arXiv preprint arXiv:1809.09968v1* ,2018.

[43] B. Ermis, and A. T. Cemgil, "Differentially Private Variational Dropout," *arXiv preprint arXiv:1712.02629v3,* 2017.

[44] B. Zheng et al., "Scalable and Privacy-Preserving Data Sharing Based on Blockchain", *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 557-567, 2018. Available: 10.1007/s11390-018-1840-5.

[45] X. Zhu, H. Li, Y. Yu, "Blockchain-Based Privacy Preserving Deep Learning," *International Conference on Information Security and Cryptology*, vol. 11449, pp. 370-383, Springer, Cham, 2018.

[46] L. Liu, C. Wu, and J. Xiao, "Blockchain-Based platform for Distribution AI," No. 764. *EasyChair*, 2019.

[47] J. Weng, J. Weng, M. Li, Y. Zhang, W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive", *IACR Cryptology ePrint Archive*, vol. 2018, pp. 679, 2018.

[48] G. Zyskind et al., "Decentralizing privacy: Using blockchain to protect personal data", *Security and Privacy Workshops (SPW) 2015 IEEE. IEEE*, pp. 180-184, 2015.

[49] M. Reuter, "Privacy Preserving Deep Neural Network Prediction using Trusted Hardware, " 2018.

[50] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No Peek: A Survey of private distributed deep learning," *arXiv preprint arXiv:1812.03288*, 2018.

[51] S. Han, H. Mao, W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[52] Zhang, Q., Wang, C., Wu, H., Xin, C., & Phuong, T. V. (2018, July). GELU-Net: A Globally Encrypted,

Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning. In IJCAI (pp. 3933-3939).

[53] On Lightweight Privacy-Preserving Collaborative Learning for Internet-of-Things Objects 2019

[54] Fu, Y., Wang, H., Xu, K., Mi, H., & Wang, Y. (2019, April). Mixup Based Privacy Preserving Mixed Collaboration Learning. In 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE) (pp. 275-2755). IEEE.

[55] Hartmann, V., & West, R. (2019). Privacy-Preserving Distributed Learning with Secret Gradient Descent. arXiv preprint arXiv:1906.11993.

[56] Zhao, L., Zhang, Y., Wang, Q., Chen, Y., Wang, C., & Zou, Q. (2018). Privacy-preserving collaborative deep learning with irregular participants. arXiv preprint arXiv:1812.10113.

[57] Hao, M., Li, H., Xu, G., Liu, S., & Yang, H. (2019, May). Towards Efficient and Privacy-Preserving Federated Deep Learning. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[58] Phong, L. T., & Phuong, T. T. (2018). Privacy-Preserving Deep Learning for any Activation Function. arXiv preprint arXiv:1809.03272.

[59] Yu, C. H., Chou, C. N., & Chang, E. (2019, March). Distributed Layer-Partitioned Training for Privacy-Preserved Deep Learning. In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp. 343-346). IEEE.

[60] Leroux, S., Verbelen, T., Simoens, P., & Dhoedt, B. (2018). Privacy aware offloading of deep neural networks. arXiv preprint arXiv:1805.12024.

[61] Raval, N., Machanavajjhala, A., & Pan, J. (2019). Olympus: Sensor Privacy through Utility Aware Obfuscation. Proceedings on Privacy Enhancing Technologies, 2019(1), 5-25.

[62] Huang, K., Liu, X., Fu, S., Guo, D., & Xu, M. (2019). A Lightweight Privacy-Preserving CNN Feature Extraction Framework for Mobile Sensing. IEEE Transactions on Dependable and Secure Computing.

[63] Ma, X., Chen, X., & Zhang, X. (2019). Non-interactive privacy-preserving neural network prediction. Information Sciences, 481, 507-519.

[64] Zhu, Q., & Lv, X. (2018). 2P-DNN: Privacy-Preserving Deep Neural Networks Based on Homomorphic Cryptosystem. arXiv preprint arXiv:1807.08459.

[65] Vizitiu, A., Niţă, C. I., Puiu, A., Suciu, C., & Itu, L. M. (2019, June). Towards Privacy-Preserving Deep Learning based Medical Imaging Applications. In 2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA) (pp. 1-6). IEEE.

[66] Chi, J., Owusu, E., Yin, X., Yu, T., Chan, W., Liu, Y., ... & Tague, P. (2018, October). Privacy Partition: A Privacy-Preserving Framework for Deep Neural Networks in Edge Networks. In 2018 IEEE/ACM Symposium on Edge Computing (SEC) (pp. 378-380). IEEE.

[66'] Chi, J., Owusu, E., Yin, X., Yu, T., Chan, W., Tague, P., & Tian, Y. (2018). Privacy partitioning: Protecting user data during the deep learning inference phase. arXiv preprint arXiv:1812.02863.

[67] Osia, S. A., Shamsabadi, A. S., Taheri, A., Rabiee, H. R., & Haddadi, H. (2018). Private and Scalable Personal Data Analytics Using Hybrid Edge-to-Cloud Deep Learning. Computer, 51(5), 42-49.

[68] Osia, S. A., Shamsabadi, A. S., Taheri, A., Katevas, K., Sajadmanesh, S., Rabiee, H. R., ... & Haddadi, H. (2017). A hybrid deep learning architecture for privacy-preserving mobile analytics. arXiv preprint arXiv:1703.02952.

[69] Adesuyi, T. A., & Kim, B. M. (2019, February). A layer-wise Perturbation based Privacy Preserving Deep Neural Networks. In 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) (pp. 389-394). IEEE.

[70] Hoefer, N. D., & Monroy, S. A. S. (2018, December). Performance Evaluation of a Differentially-private Neural Network for Cloud Computing. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 2542-2545). IEEE.

[71] Dwork, Cynthia, and Aaron Roth. "The algorithmic foundations of differential privacy." Foundations and

Trends® in Theoretical Computer Science 9.3-4 (2014): 211-407.

[72] Oneto, L. et. al. Pattern Recognition Letters, Pag:31-38 - Differential privacy and generalization: Sharper bounds with applications, Vol:89 – 2017.

[73] Fontaine, C., & Galand, F. (2007). A survey of homomorphic encryption for nonspecialists. EURASIP Journal on Information Security, 2007, 15.