Secure Deep Neural Network Models Publishing Against Membership Inference Attacks Via Training Task Parallelism

Yunlong Mao, *Member, IEEE*, Wenbo Hong, *Student Member, IEEE*, Boyu Zhu, *Student Member, IEEE*, Zhifei Zhu, *Student Member, IEEE*, Yuan Zhang, *Member, IEEE*, and Sheng Zhong, *Member, IEEE*

Abstract—Vast data and computing resources are commonly needed to train deep neural networks, causing an unaffordable price for individual users. Motivated by the increasing demands of deep learning applications, sharing well-trained models becomes popular. The owner of a pre-trained model can share it by publishing the model directly or providing a prediction interface. Either way, individual users can benefit from deep learning without much cost, and computing resources can be saved. However, recent studies of machine learning security have identified severe threats to these model publishing approaches. This article will focus on the privacy leakage issue of publishing well-trained deep neural network models. To tackle this problem, we propose a series of secure model publishing solutions based on training task parallelism. Specifically, we show how to estimate private model parameters through parallel model training and generate new model parameters in a privacy-preserving manner to replace the original ones for publishing. Based on data parallelism and parameter generating techniques, we design another two solutions concentrating on model quality and parameter privacy, respectively. Through privacy leakage analysis and experimental attack evaluation, we conclude that deep neural network models published with our solutions can provide on-demand model quality guarantees and resist membership inference attacks.

Index Terms—Machine learning security, membership inference attack, data parallelism, deep neural network

1 Introduction

Deep Neural Networks (DNNs) have significantly improved the user experience of many applications in recent years, such as personal advertisement recommendations [2], facial recognition [3], and voice-controlled intelligent devices [4]. By learning the hidden relationship between input and output on a substantial dataset, a DNN model is supposed to be sophisticated enough to approximate any function, according to the universal approximation theorem given in [5]. However, training such a DNN model requires vast computing resources and massive training data. Thus, even aided by powerful devices, training a DNN model is still expensive [6].

To meet the rising demands of DNN applications, many designs of machine learning as a service (MLaaS) have emerged, such as Google AI Platform, AWS Deep Learning, and Azure Machine Learning Service. Among all kinds of MLaaS, sharing well-trained DNN models by publishing model parameters (or weights, interchangeably) directly [7] and sharing inference interfaces [8] are two common approaches. However, DNN models trained with private datasets are intellectual properties. Therefore, it is hazardous to share private DNN models, especially those trained with sensitive data (e.g., financial data or diagnostic data). Furthermore, recent research has found that publishing a well-trained DNN model could cause serious privacy leakage, including membership inference [9], [10], [11], task property leakage [12], [13], and representative data reconstruction [14], [15].

Severe data privacy issues in DNN model publishing have attracted plenty of studies. Generally, recent studies can be categorized into three types, i.e., adversarial training [16], secure computing [17], and differential privacy [18]. Adversarial training solutions will have a task model and an adversarial model trained together in a game way. When the training task ends, a trained task model can be more robust against privacy leakage threats. Secure computing solutions basing on cryptographic tools can provide strong security guarantees. Differentially private solutions could provide a strong privacy guarantee because the artificial noise is introduced. These solutions have a good performance in the defense against data leakage threats.

However, there exists a paradox between learning and privacy protection. Learning algorithms are designed to capture knowledge hidden in the data, while privacy protection aims to preserve individual information. If a DNN

• *The authors are with the State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: {maoyl, zhongsheng}@nju.edu.cn, {wenbo.hong, 181860167}@smail.nju.edu.cn, zhuby@outlook.com, zhangyuan05@gmail.com.*

Manuscript received 31 Aug. 2021; revised 11 Nov. 2021; accepted 16 Nov. 2021. Date of publication 22 Nov. 2021; date of current version 23 May 2022.
This work was supported in part by the National Key R&D Program of China under Grants 2020YFB1005900, NSFC-61902176, BK20190294, NSFC-61872179, and NSFC-61872176, and in part by the Leading-edge Technology Program of Jiangsu NSF under Grant BK20202001.
(Corresponding author: Sheng Zhong.)
Recommended for acceptance by A. J. Peña, M. Si and J. Zhai.
Digital Object Identifier no. 10.1109/TPDS.2021.3129612

model seeks desirable usability, then sufficient data diversity inside categories is necessary. However, contributing to data diversity could lead to users' data being distinguished. Hence, it is rather challenging for the existing learning solutions to achieve multiple desired features, like model quality, training efficiency, and model privacy at the same time.

For the purpose of sharing DNN models with model quality and model privacy preserved, we propose a private DNN model publishing solution on the basis of data parallelism. Specifically, we collect intermediate training results of multiple tasks for the same DNN architecture with different training data in a parallel manner. Those models learned with differential data parallelism (one or more data samples may vary between parallelisms) compose a new dataset. We name this new dataset as "parameter collection", which has all parameters of a DNN model as one entry. Thus, each parameter in an entry could be seen as a feature of the whole dataset. Then by performing a specific learning algorithm on parameter collection, we could obtain an approximate distribution of model parameters in stable states. Once we have learned an approximate shape of parameter distribution, we can construct a DNN model that is entirely different from the real private model by generating parameters on estimated distributions.

In the previous study [1], we have implemented a parameter generating solution as an alternative way for model publishing. This solution has been proved to be effective when dealing with a membership inference attack in the black-box setting [9]. However, membership inference attacks have been evolved recently. The previous solution cannot cover a new membership inference attack in the white-box setting [11]. Meanwhile, we have made a further step towards decreasing privacy loss for model publishing. As a result, we construct a new model publishing solution dedicating on saving privacy budget by combining deep compression techniques with a differentially private parameter generating method. We prove that our new solution can significantly reduce privacy loss compared to previous work. Moreover, according to experimental results, this new solution has a better performance in defending against membership inference attacks in both black-box and white-box settings. The attack mitigated by our new solution will behave like random guesses.

1) We find parameter similarity in well-trained DNN models between separate training tasks. Based on this observation, we propose a private DNN model publishing solution by generating approximate parameters.
2) To trade privacy for model quality, we propose a quality-aware private model publishing solution by combining the original solution with a private parameter grouping method, which preserves crucial connections between parameters.
3) To further reduce the privacy loss, we propose a strictly privacy-preserved model publishing solution by promoting private parameter generating solution with model compression techniques. In this way, the privacy loss can be sharply reduced at the cost of a slight model quality loss.
4) We prove that DNN models published by our solutions are differentially private through detailed analysis. We also prove that published models are resistant to threatening membership inference attacks in both black-box and white-box settings.

2 Related Work

Previous studies have proved that DNN model publishing can break data privacy seriously, such as label-representative information [14], [15], membership information [9], [19], [20], [21], task-relevant properties [12], [13] and even human genomic data [22]. These security threats pose urgent demands for a secure solution for DNN models publishing. To meet this demand, many research efforts have been made from different angles. Previous studies on this topic can be summarized into three categories according to how model privacy is preserved [23]: gradient level, object level, and label level.

Gradient-level noise based solutions have been studied widely. In [24], sparse vector technique (SVT) is introduced into distributed DNN learning. SVT provides distributed learning with a selective gradients method which could guarantee sharing gradients with other participants will not leak private training information. In [25], authors have given a differentially private stochastic gradient descent (DP-SGD) algorithm which use moments accountant analysis to track the privacy loss in the batch-wise updates. To balance model utility and privacy preservation, the authors of [26] have estimated privacy loss by concentrated differential privacy and developed a dynamic privacy budget allocation to improve model training accuracy.

Objective-level artificial noise based solutions commonly introduce noise into the objective function. For example, a previous study [27] proposes a polynomial approximation of objective function for calculating the sensitivity then an objective-level differentially private deep learning solution is constructed based on functional mechanism. Another work [28] uses Chebyshev expansion to approximate the objective function to the polynomial form for higher accuracy. It proposes an adaptive Laplace mechanism to construct a differentially private convolutional deep belief network.

A label-level privacy-preserving DNN learning approach PATE is proposed in [29]. PATE transfers knowledge from the ensemble of teacher models which are trained on horizontal partitions of private data to a public student model. In this solution, training data privacy could be ensured by adding artificial noise to teachers' answers. Aiming at the application of PATE in more complex situations, an improved solution is given in [30], which proposes a noisy aggregation mechanism with a selective procedure to achieve a tighter privacy budget.

Given all these previous works which depend on additional artificial noise at different levels, this paper aims to give a model publishing solution without any interference in the original training phase. Specifically, we will use the ambiguously generated DNN model to replace the real one. In this way, two significant advantages can be obtained. First, there will be no need to worry about the convergence of noisy DNN training algorithms. Second, compared with previous works, a smaller privacy budget can be achieved by our solution when we publish a DNN model with the same level of model utility.

Meanwhile, training DNN models in a parallel manner has been studied intensively. As a result, excellent parallel training solutions have been proposed for data parallelism [31], model parallelism [32], and pipeline parallelism [33], providing strong practicability to our task parallelism-based publishing solution. In our work, we will train parallel models using neighboring datasets with the same DNN architecture to construct parameter collection.

3 Preliminary

For practical concerns, we will take into account both black-box and white-box settings, which means that the publisher could choose to share a DNN model through providing a query interface or raw model parameters.

3.1 Model Publishing

First we briefly review essential notations in deep learning. Given a training dataset X and DNN model parameters $\boldsymbol{\theta}$, a training task is to find approximately optimal parameters $\boldsymbol{\theta}$ by minimizing a pre-defined loss function $\mathcal{L}$ regarding input X. Assume that the optimizer used is a mini-batch stochastic gradient descent (SGD) algorithm, which updates $\boldsymbol{\theta}$ with a batch input iteratively. Assuming the batch size is N, then the total loss of $\boldsymbol{\theta}$ for a batch input $\boldsymbol{x} = \{x_i | x_i \in \boldsymbol{X}, i \in [1, N]\}$ should be $\sum_{x \in \boldsymbol{x}} \mathcal{L}(\boldsymbol{\theta}, x)$ in the tth training iteration. The gradients of $\boldsymbol{\theta}$ for model updating should be estimated by $\frac{1}{N} \sum_{x \in \boldsymbol{x}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, x)$ approximately. Hence, parameters $\boldsymbol{\theta}$ can be updated as $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \frac{1}{N} \sum_{x \in \boldsymbol{x}} \nabla_{\boldsymbol{\theta}} \mathcal{L} \;\; (\boldsymbol{\theta}, x)$. Generally, we assume that when the training reaches the Tth iteration, the model will be ready for publishing.

3.2 Differential Privacy

Differential privacy (DP) [34] has been a defacto standard for privacy-preserving data publishing and analysis. DP provides a general framework for building privacy-preserving mechanisms for specific applications by following DP mechanism designs.

Definition 1 (Differential Privacy). *A random mechanism $M : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subsets of outputs $S \subset \mathcal{R}$ it holds that*

$$\Pr[M(d) \in S] \leq e^{\epsilon} \Pr[M(d') \in S] + \delta. \tag{1}$$

This DP definition proposed in [35] ensures that original ϵ-DP can be broken with δ probability. An exponential mechanism (EM) proposed in [36] is also needed for our solution. EM is generally designed for query function $q : \mathcal{D}^n \times \mathcal{R} \rightarrow \mathcal{R}$, which will assign a real-valued score to a sample (d, r) drawn from $\mathcal{D}^n \times \mathcal{R}$. Given $d \in \mathcal{D}^n$, EM will return an $r \in \mathcal{R}$, maximizing the score $q(d, r)$ approximately. To have a higher score occur more frequently, a base measure μ associated with r is required.

Definition 2. *For any function $q : \mathcal{D}^n \times \mathcal{R} \rightarrow \mathcal{R}$, and base measure μ over $\mathcal{R}$, we define $\varepsilon_q^{\epsilon}(d)$ as choosing r with probability proportional to* $\exp(\epsilon q(d, r)) \times \mu(r)$.

Following the theorem in [36], ε_q^{ϵ} satisfies $(2\epsilon\Delta_q)$-DP.

3.3 Threat Model

Generally, once a pre-trained model is published, any client who has been granted to access the model could be an adversary and is capable of performing arbitrary computation on model parameters. Yet various threats have been identified against a pre-trained DNN model, such as model inversion attack [14], [37] and membership inference attack [9], [11]. In this paper, We consider an adversary focusing on the membership inference attack. In this kind of attack, an adversary aims to distinguish whether a data participated in published model training or not. According to different adversarial abilities, we consider two inference attack modes, black-box attack and white-box attack.

In the black-box attack, the adversary infers private information of the published model through observing input samples and the corresponding output of the model. During the observation, the adversary could make an estimation of more test data. By training multiple shadow models which are supposed to be equivalent to the published model using data drawn from the estimated data distribution, the adversary could obtain an attack model F. Given the published model $\boldsymbol{\theta}$, we give the adversarial goal of a black-box membership inference attack [9] as

$$\mathcal{A}_{bb}(\boldsymbol{X}) = \arg\min_{F} \sum_{x_i \in \boldsymbol{X}} \mathcal{L}_F(F(\boldsymbol{\theta}, x_i), l_i), l_i \in \{0, 1\},$$

where $\mathcal{L}_F$ is the loss function used for attack model F.

In the white-box attack, the adversary have the entire knowledge of the published model, including model parameters $\boldsymbol{\theta}$. Hence, the adversary can use intermediate calculation to assist inference attacks. We consider a state-of-the-art white-box attack [11]. The adversary distinguishes members by discriminating whether the model's prediction relies on characteristic features or uncharacteristic features. The characteristic features are actually used for prediction while uncharacteristic features are merely caused by overfitting and coincidence. In other words, characteristic features are also used by other classifiers trained on other datasets sampled from the identical distribution. But uncharacteristic features will not appear in other classifiers.

To classify the features, the adversary need to analyze whether a feature has effect on the final decision in an uncharacteristic way. To this end, the adversary handle intermediate output as features and slice each layer of the published model into two parts $\boldsymbol{\theta}(\boldsymbol{X}) = g(\boldsymbol{X}) \circ h(\boldsymbol{X})$. Then the adversary use internal influence $\boldsymbol{\chi}(g \circ h, \boldsymbol{X})$ to infer feature representations and the corresponding influence. Furthermore, the adversary trains a proxy model $\tilde{g}$ using $\boldsymbol{X}'$, which is sampled from an auxiliary data distribution. Since simple element-wise subtraction is not suitable for measuring distance of data distributions, a distance function $Dist(\boldsymbol{X}, \boldsymbol{X}')$ is trained for each layer. We briefly interpret the combination of each layer in the original attack model as a meta model F. Thus, the adversarial goal of white-box attack $\mathcal{A}_{wb}$ is to obtain appropriate $Dist$ and F

$$\mathcal{A}_{wb}(\boldsymbol{X}) = \arg\min_{F, Dist} \sum_{x_i \in \boldsymbol{X}} \mathcal{L}_F(F(Dist(g(x_i), \tilde{g}(x_i)), h(x_i)), l_i), l_i \in \{0, 1\}.$$

TABLE 1
Notation Table

Π	the universal set of DNN models
$\boldsymbol{\theta}$	all parameters of a certain DNN model
$\boldsymbol{\theta}^i$	parameters of the ith DNN model of Π
θ_j	model parameter in the jth position
Θ	parameter collection
$\boldsymbol{\theta}[j]$	the jth slice of the parameter collection
$\boldsymbol{\theta}(l)$	parameters within the lth layer
p_j	pruning ratio of the jth parameter
p_c	precision of centroid representations
p_r	precision of residual representations

4 Secure Model Publishing Solutions

When a provider shares a DNN model, it is critical to ensure that model privacy is not disclosed. To this end, we will introduce a private model parameter generating algorithm, composed of parallel training tasks. The basic idea of the generating algorithm is to use the approximate estimation of private models to construct a more generalized DNN model. Then we design two different model publishing solutions focusing on model quality and privacy respectively. Some necessary notations in this section are summarized in Table 1 for quick reference.

4.1 Private Parameter Generating

We assume that a set of model $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ is constructed by training the same DNN architecture separately with data samples drawn from identical data distribution. Assuming the total parameter amount in this DNN architecture is N, we denote by $\boldsymbol{\theta}^i$, $|\boldsymbol{\theta}^i| = N$ the set of all parameters in model $\pi_i, i \in [1, M]$. Although parameters are organized in layers, we can simply flatten them to be a vector, i.e., $\boldsymbol{\theta}^i = \{\theta^i_1, \theta^i_2, \ldots, \theta^i_N\}$. Then we define "parameter collection" Θ as a new dataset, whose entries are $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots, \boldsymbol{\theta}^M$. Then any parameter θ^i_j located in position j in $\boldsymbol{\theta}^i$ of model π_i can be seen as an attribute of entry $\boldsymbol{\theta}^i, i \in [1, M]$. If we slice Θ vertically, we can get parameters located in the same position for all models. We use $\boldsymbol{\theta}[j], j \in [1, N]$ to indicate vertical slices of Θ, which are elementary datasets that we are going to protect.

Generally, the optimizing process may vary for repetitive training tasks because of the randomness in training. With the help of training task parallelism, we can exploit the randomness in depth for generating a generalized model. Now we need to introduce some essential definitions.

Neighboring datasets for client's query of each parameter are two neighboring slices $\boldsymbol{\theta}[j]$ and $\boldsymbol{\theta}'[j]$ of Θ. Two slices only differ on one element, $|\boldsymbol{\theta}[j]| = M$, $|\boldsymbol{\theta}'[j]| = M - 1$. In this way, the parameter collection contains private information no less than any single DNN model.

Query function varies widely in differentially private mechanism designs. We give a new query function design for DNN model publishing here. Given parameter collection Θ, there is no need for clients to query the original training data, since Θ has sufficient information to construct a full-functional DNN model. Instead, Θ will be queried by a composite function, which is composed of a statistical procedure f and a sampling procedure g. Specifically, we will use kernel density estimation (KDE) to implement f and use the exponential mechanism (EM) to construct g. When Θ is queried, each slice $\boldsymbol{\theta}[j]$ for any $j \in [1, N]$ is queried independently.

Model Privacy of the published model is defined in a differentially private manner. If a client cannot tell the query result of $\boldsymbol{\theta}[j]$ is obtained from $\boldsymbol{\theta}[j]$ or its neighboring dataset $\boldsymbol{\theta}'[j]$, $j \in [1, N]$, we say the model privacy regarding this query is preserved.

Since elements in slice $\boldsymbol{\theta}[j], j \in [1, N]$ are collected from training task parallelism, each element can be seen as a data sample drawn from a parameter distribution. Based on this conjecture, we use KDE method [38] to approximately estimate the distribution of elements in $\boldsymbol{\theta}[j]$. Specifically, we estimate elements in $\boldsymbol{\theta}[j]$ by

$$f_{j,b}(\boldsymbol{\theta}[j]) = \frac{1}{M \times b} \sum_{\theta \in \boldsymbol{\theta}[j]} \phi\left(\frac{\theta_j - \theta}{b}\right), \tag{2}$$

where b is a bandwidth (also known as smoothing parameter) of the estimator, ϕ is normal density function. Please note that the smoothing parameter b should be set empirically. Without causing any ambiguity, we will ignore subscript b in the rest. Having distribution of any slice $\boldsymbol{\theta}[j], j \in [1, N]$ in parameter collection Θ approximately estimated as $f_j(\boldsymbol{\theta}[j])$, the next step is to design a sampling procedure g to output parameters to construct a fully-functional DNN model. Since more precise parameters we sample, higher probability is to reveal model privacy. To solve this problem, we will use EM to construct our sampling procedure. We will also show that the KDE sampling approach can be perfectly integrated into EM.

To design an EM based private publishing solution, it is important to define a proper score function $u : \mathcal{R}^M \times \mathcal{R} \rightarrow \mathcal{R}$, mapping pairs of parameter collection slice and output parameter to real-valued scores. Since EM tries to output some element of $\mathcal{R}$ with the maximum possible score, we can give the formal definition of u by associating KDE result with output score. Then we have

$$u_j(\boldsymbol{\theta}[j], \theta_j) = \int_{\theta_j - \frac{\delta}{2}}^{\theta_j + \frac{\delta}{2}} f_b(\boldsymbol{\theta}[j]), \tag{3}$$

where δ is a small window for random sampling which could be optimized empirically. As proposed in [34], score function for EM can be arbitrarily sensitive in its range, which means the sensitivity of u_j should be defined as

$$\Delta_{u_j} = \max_{\theta_j \in \mathcal{R}} |u_j(\boldsymbol{\theta}[j], \theta_j) - u_j(\boldsymbol{\theta}'[j], \theta_j)|, j \in [1, N]. \tag{4}$$

For parameter $\theta_j, j \in [1, N]$, we set privacy budget ϵ_j. Then the publisher performs parallel training tasks to obtain M DNN models, composing Π. Parameter collection Θ is constructed by flattening and stacking parameters $\boldsymbol{\theta}^i$ of each model in Π, $|\boldsymbol{\theta}^i| = N$, $i \in [1, M]$. The publisher slices Θ vertically to get isolated datasets $\boldsymbol{\theta}[j]$, $j \in [1, N]$ and performs KDE on each $\boldsymbol{\theta}[j]$ to obtain approximate estimation $f(\boldsymbol{\theta}[j])$ for the jth parameter. If a privacy budget ϵ_j is set for each $\boldsymbol{\theta}[j]$, then the publisher can generate each parameter θ_j with probability proportional to $\exp(\frac{\epsilon_j u(\boldsymbol{\theta}[j], \theta_j)}{2\Delta_{u_j}})$ using $f(\boldsymbol{\theta}[j])$ and $u(\boldsymbol{\theta}[j], \theta_j)$. For the concern of model usability, we can set a quality threshold δ_d. If the test accuracy of the generated model is not larger than δ_d, then we can deny the model and repeat the algorithm. Otherwise, we get a model ready to be published.

We summarize the above steps as our differentially private parameter generating (DP-PG) algorithm and give its sketch in Algorithm 1. Keyword *Test* and *Sample* used in Algorithm 1 are functions which give test accuracy of a model and parameter samples following EM respectively.

Algorithm 1. DP-PG Algorithm

```
1: initial $\boldsymbol{\theta} = 0$;
2: while $\texttt{Test}(\boldsymbol{\theta}) \leq \delta_d$ do
3:     for $j = 1$ to $N$ do
4:         $f_b(\theta_j) = \texttt{KDE}(\boldsymbol{\theta}[j])$;
5:         $u_j(\boldsymbol{\theta}[j], \theta_j) = \int_{\theta_j - \frac{\delta_g}{2}}^{\theta_j + \frac{\delta_g}{2}} f_j(\boldsymbol{\theta}[j])$;
6:         $\theta_j = \texttt{Sample}(\exp(\frac{\epsilon_j u_j(\boldsymbol{\theta}[j], \theta_j)}{2\Delta_{u_j}}))$;
7:     end for
8: end while
```

Algorithm 1 takes as input the parameter collection Θ. We will focus on the efficiency of the algorithm in the evaluation. But we note that Θ can be effectively constructed using data parallelism. Particularly, for each $\boldsymbol{\theta} \in \Theta$, we can train the model using the prepared data separately. Since training datasets of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j \in \Theta$ are slightly different, it is also possible to utilize the hybrid parallelism technique, achieving a more efficient parameter collection purpose. Each model is trained using partitioned data in data parallelism, while different parts of a model are trained using the same dataset in model parallelism. We find that data parallelism and model parallelism are beneficial to achieving differential privacy in our solution since we need data parallelism for training different models and model parallelism for faster harvesting parameters. We note that hybrid parallelism is feasible for the parameter collecting process because we handle parameters element-wise. Therefore, each parameter could be trained in a vertically or horizontally partitioned model.

4.2 Model Publishing With Parameters Grouping

In our basic publishing solution, we handle each parameter of a DNN model separately, which poses a threat to the usability of the published model. One immediate result is that model quality could be frustrated because hidden relations between parameters are broken. Fortunately, we find this circumstance can be relieved by relaxing the assumption of individual parameter generating. It has been shown that DNN model parameters have potential connections and different importance in design space [39], [40]. Based on this result, we propose a quality-aware publishing solution, which combines the DP-PG algorithm with parameters grouping. The core idea is to preserve connections between parameters with a selected portion of the model. Parameters outside the selected portion will be published using the original DP-PG.

Since parameters within the same DNN layer are tightly connected [41], we perform parameters grouping for each layer separately. Assume a specific DNN model π_0 to be published consisting of L layers. We denote by $\boldsymbol{\theta}_{(l)}$, all parameters in the lth layer, $l \in [1, L]$. If we flatten $\boldsymbol{\theta}_{(l)}$ into a vector, we can denote by $\theta_{(l,i)}$ the ith parameter of $\boldsymbol{\theta}_{(l)}$, $i \in [1, N_l]$ where N_l is amount of parameters in the lth layer. Now, we sort $\boldsymbol{\theta}_{(l)}$ by parameter significance and get the result $\boldsymbol{\theta}'_{(l)}$ in descending order. A predefined selection ratio γ_l will be used to control the scale of parameter selection. Specifically, parameters stored in top γ_l of $\boldsymbol{\theta}'_{(l)}$ will be selected for $\boldsymbol{\theta}_{(l)sel} = \{\theta_i | 1 \leq i \leq \gamma_l N_l\}$. Other parameters will be handled with the DP-PG algorithm separately.

To capture parameter connections in $\boldsymbol{\theta}_{(l)sel}$, we use a clustering algorithm for parameters grouping in a private manner. After running a private k-means algorithm [42] on $\boldsymbol{\theta}_{(l)sel}$, k clusters $\boldsymbol{K} = \{K_1, K_2, \ldots, K_k\}$ and the corresponding centroids $\boldsymbol{c} = \{c_1, c_2, \ldots, c_k\}$ can be obtained. We denote by $\theta_j^{K^i}$ the parameter in cluster K^i, $j \in [1, |K^i|]$. Clearly, $K^i \subset \boldsymbol{\theta}_{(l)sel}$, $i \in [1, k]$. Replacing parameters within the cluster with a centroid will lead to the loss of model quality. To tackle this problem, we construct a new differentially private mechanism to publish parameters grouped in the same cluster. Specifically, we design an obfuscated distance function $o_i : \mathcal{R}^{|K^i|} \to \mathcal{R}$ for any $\theta_j^{K^i} \in K^i$

$$o_i(\theta_j^{K^i}) = \theta_j^{K^i} - c_i + Laplace\left(\frac{\Delta_{o_i}}{\epsilon_{K^i}}\right), \tag{5}$$

where ϵ_{K^i} is the privacy budget of distance function for cluster K^i while Δ_{o_i} is the sensitivity of o_i regarding neighboring datasets in $\mathcal{R}^{|K^i|}$ (e.g., K^{i1}, K^{i2}). Then we have

$$\Delta_{o_i} = \max_{K^{i1} - K^{i2}} |o_i(\theta_j^{K^{i1}}) - o_i(\theta_j^{K^{i2}})|. \tag{6}$$

We summarize this differentially private parameter grouping and publishing (DP-PGP) solution in Algorithm 2. Please note that parameter connections' preserving is controlled by selection ratio γ_l. When $\gamma_l = 0, \forall l \in [1, L]$, the DP-PGP algorithm will be degenerated to the DP-PG algorithm.

Algorithm 2. DP-PGP Algorithm

```
1: initial $\boldsymbol{\theta} = 0$;
2: while $\texttt{Test}(\boldsymbol{\theta}) \leq \delta_d$ do
3:     for $i = 1$ to $N$ do
4:         $f_b(\theta_i) = \texttt{KDE}(\boldsymbol{\theta}[i])$;
5:         $u_i(\boldsymbol{\theta}[i], \theta_i) = \int_{\theta_i - \frac{\delta_g}{2}}^{\theta_i + \frac{\delta_g}{2}} f_i(\boldsymbol{\theta}[i])$;
6:     end for
7:     for $l = 1$ to $L$ do
8:         $\boldsymbol{\theta}'_{(l)} =$ sort $\boldsymbol{\theta}_{(l)}$ in descending order;
9:         for $i = 1$ to $N_l$ do
10:            if $i \leq \gamma_l N_l$ then
11:                $\boldsymbol{\theta}_{(l)sel} = \boldsymbol{\theta}_{(l)sel} \cup \{\theta_i\}$;
12:            else
13:                $\theta_i = \texttt{Sample}(\exp(\frac{\epsilon u_i(\boldsymbol{\theta}[i], \theta_i)}{2\Delta_{u_i}}))$;
14:            end if
15:        end for
16:        $(\boldsymbol{K}, \boldsymbol{c}) = \texttt{DP} - \texttt{Kmeans}(\boldsymbol{\theta}_{(l)sel})$;
17:        for $i = 1$ to $k$ do
18:            for $j = 1$ to $|K^i|$ do
19:                $\theta_j = c_i + o_i(\theta_j^{K^i})$;
20:            end for
21:        end for
22:    end for
23: end while
```

4.3 Model Publishing With Compressing

As the basic parameter generating solution, the DP-PG yields acceptable privacy loss. To improve model quality, the DP-PGP trades additional privacy budget for the accuracy. However, these solutions are not dedicated to a strict privacy requirement. Hence, we design an alternative publishing solution for further reducing the privacy loss of the original DP-PG. In particular, we propose a differentially private parameter generating and compressing (DP-PGC) method for model publishing under a tight privacy budget. DNN model compressing method [43] aims to decrease the redundancy of model parameters. Generally, there are two main approaches for DNN compressing, pruning [44] and quantization [45]. Pruning technique is commonly used to reduce the amount of parameters, which will change the structure of a DNN model while quantization technique is used to reduce the representing bits of parameters in place.

Intuitively, we perform a compressing procedure as the post processing right after the original DP-PG algorithm. Recall that the output of the DP-PG algorithm is a model $\boldsymbol{\theta}$ generated under the privacy budget $\epsilon_g = \max\{\epsilon_j | j \in [1, N]\}$. Given $\boldsymbol{\theta}$, the compressing procedure is straightforward. First, each parameter $\theta_j, j \in [1, N]$ is assigned with an importance weight w_j, which can be obtained through the importance evaluation [46]. The importance weight indicates how much significant effect θ_j has on the inference phase. It should be noted that the calculation of importance weights will not introduce additional knowledge to $\boldsymbol{\theta}$. Instead, a new function p_j is constructed for transforming each importance weight into a probability. Since negative importance weights are allowed, we construct the probability with a min-max standardization. Hence, $p_j(\theta_j, \boldsymbol{\theta}) = \frac{w_j - \min\{w_i | i \in [1,N]\}}{\max\{w_i | i \in [1,N]\} - \min\{w_i | i \in [1,N]\}}, j \in [1, N]$. Then θ_j will be pruned as

$$\theta_j = \begin{cases} \theta_j, \ with\ probability\ of\ p_j(\theta_j, \boldsymbol{\theta}), \\ 0, \ otherwise. \end{cases} \tag{7}$$

Next, a k-means clustering algorithm is used after pruning. Similar with the DP-PGP, clustering can find similar parameters in the same layer and let them share the same representative values. Specifically, if we put parameters in the lth layer into k_l clusters, we are going to find the argument of the minimum sum of squares by solving $\arg\min \sum_{i \in [1,k_l]} \sum_{\theta \in K^i, c_i \in \boldsymbol{c}} (\theta - c_i)^2$, where $\boldsymbol{c} = \{c_1, c_2, \ldots, c_{k_l}\}$ is the set of clustering centroids and K^i is a set of parameters sharing the same centroid c_i, $i \in [1, k_l]$, $l \in [1, L]$. On the basis of the clustering result, we can obtain a new formula of parameter θ_j as $\theta_j = c_i + r_j$, for any $\theta_j \in \boldsymbol{\theta}$, when $\theta_j \in K^i$. Denoted by r_j the residual part of θ_j. In this way, the quantization can be defined as a function q_j, mapping each generated parameter into a pair of centroid and residual. Assuming that $\theta_j \in \boldsymbol{\theta}_{(l)}$, then $q_j(\theta_j, \boldsymbol{\theta}_{(l)}) = c_i + r_j$, $i \in [1, k_l]$, $j \in [1, N]$.

We allow parameters to have the same precision after compressing. We assume that all centroids are represented in p_c precision (e.g., 1e-4) while all residual parts are represented in p_r precision (e.g., 1e-6), $p_c \le p_r \le p_t$. In this way, θ_j can be reconstructed by $c_i + r_j$ without any precision loss, where p_t is the precision of model parameters. After model parameters are pruned, the model quality and privacy loss will be further affected by the precision of clustering centroids and residual parts. We give an example of parameter clustering in Fig. 1 to

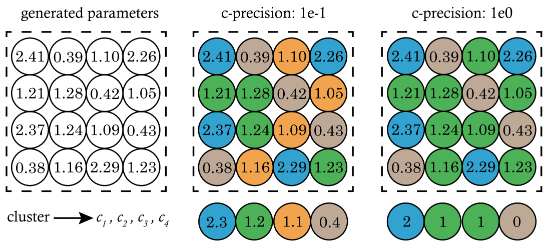


Fig. 1. Cluster model parameters with centroids in different precision.

show how parameter precision affects the clustering result. When the precision of centroids changes, a parameter may be assigned to a different cluster. Meanwhile, the number of clusters may be reduced when the precision decreases. If we use fixed-point centroids to replace parameters for publishing, the expected value of a specific parameter will be obfuscated once more. We obtain less privacy loss, but the model quality will be frustrated. Hence, a proper residual part for each parameter is necessary.

Once the precision of centroids is determined, the residual part becomes crucial. The original compressing methods use a fine-tuning step to calibrate centroids and residuals after quantization. But this additional training step will cause another privacy leakage. To maintain the usability of the publishing model during the compressing, the precision of residuals should be handled carefully. We denote by $\zeta = \frac{p_r}{p_t}$ the precision ratio between the residual and its corresponding parameter. Specifically, when $\zeta = 0$, it is a special case where no residuals are published. When $\zeta = 1$, the quantization of compressing will not have any effect on model publishing because residuals published can reconstruct original parameters precisely. In case of $\zeta \in (0, 1)$, the generated parameters will be further obfuscated through residual bits truncation. Summing up all these steps, we give DP-PGC in Algorithm 3.

Algorithm 3. DP-PGC Algorithm

```
1:  initial $\boldsymbol{\theta} = 0$;
2:  while Test($\boldsymbol{\theta}$) $\le \delta_d$ do
3:      for $i = 1$ to $N$ do
4:          $f_b(\theta_i) =$ KDE($\boldsymbol{\theta}[i]$);
5:          $u_i(\boldsymbol{\theta}[i], \theta_i) = \int_{\theta_i - \frac{\delta_g}{2}}^{\theta_i + \frac{\delta_g}{2}} f_i(\boldsymbol{\theta}[i])$;
6:          $\theta_i =$ Sample(exp($\frac{\epsilon_i u_j(\boldsymbol{\theta}[i], \theta_i)}{2\Delta_{u_i}}$));
7:      end for
8:      for $i = 1$ to $N$ do
9:          $w_i =$ Importance($\theta_i$);
10:         $p_i = \frac{w_i - \min\{w_j | j \in [1,N]\}}{\max\{w_j | j \in [1,N]\} - \min\{w_j | j \in [1,N]\}}$;
11:         $\theta_i =$ Prune($\theta_i$);
12:     end for
13:     for $l = 1$ to $L$ do
14:         $(\boldsymbol{K}, C) = $ K $-$ means($\boldsymbol{\theta}_{(l)}$);
15:     end for
16:     for $i = 1$ to $k$ do
17:         for $\theta_j$ in $K^i$ do
18:             $r_j =$ Truncate($\theta_j - c_i, p_r$);
19:             $\theta_j = c_i + r_j$;
20:         end for
21:     end for
22: end while
```

In Algorithm 3, we denote by `Importance()` an importance weight evaluation method, `Prune()` the pruning process defined in Equation (7), and `Truncate()` a function truncating the argument into a specified precision.

5 Privacy and Quality Guarantees

Our main purpose is to keep DNN model privacy leakage within a limited privacy budget while providing high model quality. The private dataset directly accessed by any legal client is Θ. For any parameter θ_j, we define a mechanism for the DP-PG algorithm as $M_{\text{DP-PG}}(\boldsymbol{\theta}[j], f_j, g_j) = \theta_j$, $j \in [1, N]$. The corresponding privacy loss caused at output θ_j is

$$b(\theta_j; M_{\text{DP-PG}}, \boldsymbol{\theta}[j], \boldsymbol{\theta}'[j]) = \frac{\Pr[M_{\text{DP-PG}}(\boldsymbol{\theta}[j], f_j, g_j) = \theta_j]}{\Pr[M_{\text{DP-PG}}(\boldsymbol{\theta}'[j], f_j, g_j) = \theta_j]}.$$

Definition 3 (Model Privacy). *For a model collection Π, given any neighboring datasets $\boldsymbol{\theta}[j]$ and $\boldsymbol{\theta}'[j]$ for any parameter θ_j, $j \in [1, N]$, if $b(\theta_j; M_{\text{DP-PG}}, \boldsymbol{\theta}[j], \boldsymbol{\theta}'[j])$ can be bounded by a fixed privacy budget, then the published model $\boldsymbol{\theta} = \{\theta_j | j \in [1, N]\}$ preserves model privacy under this privacy budget.*

Now we will show how to determine the privacy loss of DP-PG. By following the privacy accountant theorem of the EM proposed in previous work [47], we can give the privacy guarantee of each parameter published with the DP-PG algorithm by proving mechanism $M_{\text{DP-PG}}$ preserves differential privacy guarantee.

Corollary 1. *Given a score function $u : (\mathcal{R}^M \times \mathcal{R}) \to \mathcal{R}$, parameter θ_j published with the DP-PG algorithm is $(\epsilon_j, 0)$-DP, if θ_j is chosen with probability proportional to $\exp(\frac{\epsilon_j u(\boldsymbol{\theta}[j], \theta_j)}{2\Delta_{u_j}})$, for any $j \in [1, N]$.*

Proof. proof Bounding the sensitivity of query function is crucial for determining privacy loss. Now we will prove that sensitivity Δ_{u_j} is always within [0,1] for any neighboring datasets. Recall that $\forall j \in [1, N]$

$$\begin{aligned}\Delta_{u_j} &= \max_{\theta_j \in \mathcal{R}} \max_{\boldsymbol{\theta}[j], \boldsymbol{\theta}'[j]} |u_j(\boldsymbol{\theta}[j], \theta_j) - u_j(\boldsymbol{\theta}'[j], \theta_j)| \\ &= \max_{\theta_j \in \mathcal{R}} \max_{\boldsymbol{\theta}[j], \boldsymbol{\theta}'[j]} \left| \int_{\theta_j - \frac{\delta}{2}}^{\theta_j + \frac{\delta}{2}} f_j(\boldsymbol{\theta}[j]) - \int_{\theta_j - \frac{\delta}{2}}^{\theta_j + \frac{\delta}{2}} f_j(\boldsymbol{\theta}'[j]) \right| \\ &< \max_{\theta_j \in \mathcal{R}} \max_{\theta_0 \in \boldsymbol{\theta}[j], \notin \boldsymbol{\theta}'[j]} \left| \int_{\theta_0 - \frac{\delta}{2}}^{\theta_0 + \frac{\delta}{2}} \frac{1}{M \times (M-1) \times b} \right|.\end{aligned}$$

Actually, if we choose $\delta < b$, we can have $\Delta_{u_j} < \frac{1}{M \times (M-1)}$. Since Δ_{u_j} is no more than 1, we can say that $M_{\text{DP-PG}}$ preserves $(\epsilon_j, 0)$-differential privacy as long as we choose θ_j with probability proportional to $\exp(\frac{\epsilon u(\boldsymbol{\theta}[j], \theta_j)}{2\Delta_{u_j}})$, for any fixed privacy budget ϵ_j. For the whole publishing DNN model, we have N parameters in total. According to the parallel composition theorem proposed in [34], we can conclude that the privacy budget of publishing $\boldsymbol{\theta}$ is $\max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_N\}$. □

Meanwhile, the EM is supposed to give a strong utility guarantee because the output decreases exponentially when the quality score falls off [34]. In the DP-PG algorithm, we treat each θ_j in $\boldsymbol{\theta}$ as an individual query and design an independent publishing function respectively. For the slice $\boldsymbol{\theta}[j]$ of a parameter collection, we let $\text{OPT}_u(\theta_j) = \max_{\theta \in \mathcal{R}} u(\boldsymbol{\theta}[j], \theta_j)$ denote the maximum utility score of any possible $\theta_j \in \mathcal{R}$. Thus, we can measure the utility of parameter θ_j in terms of $\text{OPT}_u(\theta_j)$. Assume that the sampling interval of the KDE output distribution is β, which means any two neighboring samples on estimated distribution have β spatial distance. Then the feasible range $\mathcal{R}$ of model parameters can be quantified. To ensure range $\mathcal{R}$ finite in the analysis, we assume long tails of the estimated distribution are truncated and the area within $(\text{MAX}_{\theta_j}, \text{MIN}_{\theta_j})$ will be kept. This will lead to $|\mathcal{R}| = (\text{MAX}_{\theta_j} - \text{MIN}_{\theta_j})/\beta$. On the basis of this result, we can bound parameter utility of the DP-PG algorithm by following theorem and corollary about the EM given in [34] directly. Please note that the proof of our corollary is straightforward and will be omitted here.

Corollary 2. *If $M_{\text{DP-PG}}(\boldsymbol{\theta}[j], f_j, g_j)$ outcomes θ_j with probability proportional to $\exp(\frac{\epsilon u(\boldsymbol{\theta}[j], \theta_j)}{2\Delta_{u_j}})$, utility of the published parameter θ_j can be bounded by $\Pr[u(M_{\text{DP-PG}}(\boldsymbol{\theta}[j], f_j, g_j)) \le \text{OPT}_u(\theta_j) - \frac{2\Delta_u}{\epsilon_j}(ln(|\mathcal{R}|) + t)] \le e^{-t}, \forall \theta_j \in \boldsymbol{\theta}$.*

5.1 Privacy Analysis of DP-PGP

The main difference between DP-PG algorithm and DP-PGP algorithm is parameter grouping. For the lth layer, $l \in [1, L]$, no parameters will be selected if we set $\gamma_l = 0$, which means there is no parameter to be grouped. In this case, the DP-PGP algorithm will be equivalent to the DP-PG algorithm while publishing the lth layer.

Corollary 3. *If $\gamma_l = 0, l \in [1, L]$, the DP-PGP algorithm will be equivalent to the DP-PG algorithm.*

For any $l \in [1, L]$, if $0 < \gamma_l \le 1$, then parameter grouping will happen. Two parts of private parameter grouping may disclose private information, i.e., clustering and tuning. In the clustering phase, total privacy leakage caused by clustering algorithm may vary from the specific privacy-preserving clustering implementations. For conciseness, we will simply denote total privacy leakage caused by private clustering by ϵ_c and treat ϵ_c as a constant in the rest. In the tuning phase, parameters within each cluster will be handled by an independent Laplace mechanism. Given a privacy budget ϵ_{K^i} for parameters $\theta_j, j \in [1, |K^i|]$ in K^i, total privacy budget of tuning grouped parameters will be the maximal privacy budget across all clusters, i.e., $\max_{i=1}^{k} \epsilon_{K^i}$. For the parameters not selected for grouping, we can give privacy budget for each of them by following the corollary of DP-PG algorithm directly, which is $\max_{\theta_j \in \boldsymbol{\theta}_{(l)} - \boldsymbol{\theta}_{(l)sel}} \epsilon_j$. Now we can give total privacy budget of the lth layer as

$$b(\boldsymbol{\theta}_{(l)}) = \max\left\{ \left(\epsilon_c + \max_{i \in [1,k]} \epsilon_{K^i} \right), \max_{\theta_j \in \boldsymbol{\theta}_{(l)} - \boldsymbol{\theta}_{(l)sel}} \epsilon_j \right\}. \tag{8}$$

Taking all layers of a DNN model into account, we can conclude a generalized case of the first corollary.

Corollary 4. *For a DNN model consisting of L layers, total privacy budget of the DP-PGP algorithm will be $\max_{l=1}^{k} b(\boldsymbol{\theta}_{(l)})$ while publishing the entire model.*

5.2 Privacy Analysis of DP-PGC

As a post-processing step of the DP-PG algorithm, the privacy loss of DP-PGC algorithm can be derived from Corollary 1. Recall that there are two main operations in the DP-PGC, i.e., model pruning and parameter quantization. Since we do not take into account the privacy budget allocation in the view of the entire model, the pruning operation will not cause extra privacy loss. Because a parameter will be pruned to be zero or kept unchanged as it is generated. On the other side, the quantization operation will decrease the precision of parameters. Hence, the model quality will be thwarted by this operation but the privacy loss can be further reduced. Generally, when we assume that model parameters vary in the range of (0,1), then the sensitivity of parameter publishing should be in the same range, even if parameter precision is decreased. But the privacy loss should be re-calculated regarding the decreased precision.

Specifically, we define a mechanism for the DP-PGC algorithm as $M_{DP-PGC}(\boldsymbol{\theta}[j], p_j, q_j) = \hat{\theta}_j$. For clarity, we denote by $\hat{\theta}_j$ the truncated parameter with p_r bits and denote by $\tilde{\theta}_j$ the generated parameter with p_t bits, which has no precision loss. Moreover, we use the expression $x \gg k$ to indicate a k-bit logical right shift operation on variable x. Now the privacy loss caused at output θ_j is

$$\begin{aligned} b(\theta_j; M_{\text{DP-PGC}}, \boldsymbol{\theta}[j], \boldsymbol{\theta}'[j]) &= \frac{\Pr[M_{\text{DP-PGC}}(\boldsymbol{\theta}[j], p_j, q_j) = \hat{\theta}_j]}{\Pr[M_{\text{DP-PGC}}(\boldsymbol{\theta}'[j], p_j, q_j) = \hat{\theta}_j]} \\ &= \frac{\Pr[M_{\text{DP-PG}}(\boldsymbol{\theta}[j], f_j, g_j) = \tilde{\theta}_j | \tilde{\theta}_j \gg (p_t - p_r) = \hat{\theta}_j]}{\Pr[M_{\text{DP-PG}}(\boldsymbol{\theta}'[j], f_j, g_j) = \tilde{\theta}'_j | \tilde{\theta}'_j \gg (p_t - p_r) = \hat{\theta}_j]} \\ &= b(\tilde{\theta}_j; M_{\text{DP-PG}}, \boldsymbol{\theta}[j], \boldsymbol{\theta}'[j]) \times \frac{1}{\log \frac{2^{p_t}}{2^{p_r}}}. \end{aligned}$$

Proof of the above equation is straightforward. Since shift operation and generating model parameters with M_{DP-PG} are two independent events, we can have calculate the probability separately. When two parameters $\tilde{\theta}_j$ and $\tilde{\theta}'_j$ are generated using different parameter collections $\boldsymbol{\theta}[j]$ and $\boldsymbol{\theta}'[j]$, $\Pr[\tilde{\theta}_j = \tilde{\theta}'_j] = \epsilon_j$ according to Corollary 1. The probability of yielding the identical result after logical right shift operation for two independent variables can be determined by binary-encoding capability. Hence, we can conclude the following result for the DP-PGC algorithm.

Corollary 5. *The privacy loss of the parameter published by the DP-PGC algorithm is no more than* $\frac{\epsilon_j}{p_t - p_r}$*, when* θ_j *is generated by the DP-PG algorithm with privacy budget* ϵ_j *for any* $j \in [1, N]$*, and* $p_r \leq p_t$.

6 Evaluation

We implement and evaluate three solutions on three popular DNN architectures in Keras source code [48]. According to the training dataset of each model, we name three models *MNIST-Net*, *CIFAR-Net* and *SVHN-Net*. In particular, MNIST-Net consists of two convolutional layers and two fully connected layers, with 1,199,882 parameters in total. CIFAR-Net consists of four convolutional layers and two fully connected layers, with 1,250,858 parameters in total, and the SVHN-Net uses a similar network architecture to CIFAR-Net with 314,394 parameters. More details about DNN architectures can be found in Keras documents.

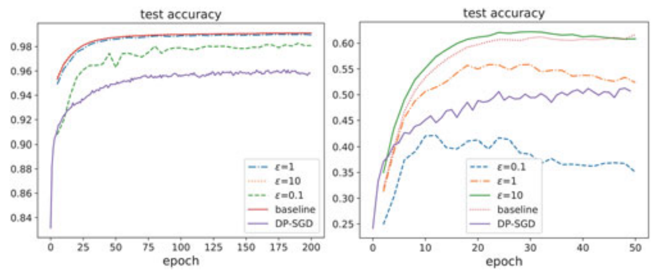


Fig. 2. Model quality evaluation for the DP-PG with MNIST-Net (left) and CIFAR-Net (right) in different model states.

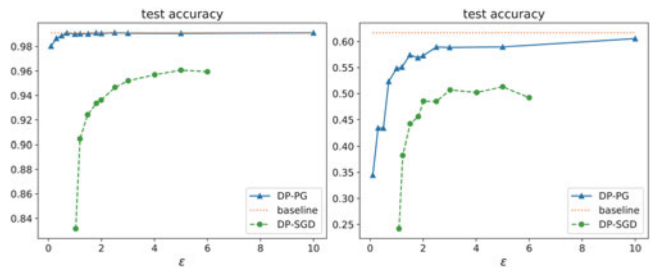


Fig. 3. Model quality evaluation for the DP-PG with MNIST-Net (left) and CIFAR-Net (right) using different privacy budgets.

The datasets we use are MNIST, CIFAR-10 and SVHN datasets. MNIST [49] is a standard handwritten digits dataset including numbers from 0 to 9. CIFAR-10 [50] is a popular image classification dataset consisting of 50,000 training images and 10,000 test images for 10 classes. SVHN is a real-world image dataset incorporating over 600,000 digit images for recognizing digits and numbers in natural scene images. To construct parameter collection for our solutions, we train each DNN for 50 times to get different models in a task-parallelism way and collect the intermediate results. All hyperparameters are the same for 50 training tasks. The batch size is 128 and the learning rate is 0.001. The baseline models are also trained in this default setting.

6.1 Model Quality Evaluation

To give a thorough evaluation on the model quality, we compare our solutions with DP-SGD [25] and a baseline without any privacy protection. In Fig. 2, we evaluate DP-PG by publishing models in different states. Since the DP-PG is the basis of the DP-PGP and DP-PGC, they share a similar performance regarding different model states. We use three fixed privacy budgets to evaluate the DP-PG when comparing with the DP-SGD and the baseline. The DP-PG can achieve commensurate result on training and testing metrics in all stages. When the privacy budget is large, publishing model will benefit from model aggregation to achieve a better performance than a single DNN model. This phenomenon has been studied in recent research work [51].

To investigate how privacy budget affects model quality, we generate DNN models in well-trained states with various budget values. As shown in Fig. 3, the DP-PG introduces less interference in model quality when compared with the DP-SGD. The model quality is affected by a small privacy budget (e.g., $\epsilon \leq 0.1$) of the DP-PG, which is rather conservative for practical use. According to this result, we

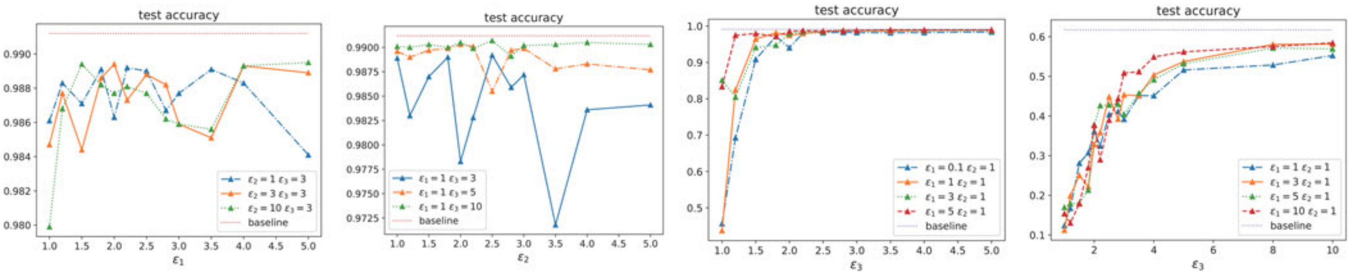


Fig. 4. Model quality evaluation for the DP-PGP with MNIST-Net (left three) and CIFAR-Net (right) using various privacy budgets.

recommend $\epsilon \leq 1$ for MNIST-Net for a balance between privacy and quality. For CIFAR-Net, the privacy budget for acceptable model quality is relatively large because training data in the dataset has less similarity than MNIST. Even though, the DP-PG approximates the baseline when $\epsilon \geq 2$.

The DP-PGP solution has additional privacy budgets for DP-Kmeans and Laplace mechanism, which should also be taken into account. We set $\epsilon_j = \epsilon_1$, $\epsilon_c = \epsilon_2$, $\forall \theta_j \in \boldsymbol{\theta}_{(l)} \setminus \boldsymbol{\theta}_{(l)sel}$, $\forall l \in [1, L]$ for DP clustering process, $\epsilon_{K^i} = \epsilon_3$ for all clusters, $i \in [1, k]$. Grouping selection ratio is 0.2. Quality evaluation of MNIST-Net and CIFAR-Net regarding ϵ_1, ϵ_2, ϵ_3 are shown in Fig. 4. The privacy budget of the EM has a small influence on model quality because it has been proved to be sufficiently small for an acceptable accuracy in the DP-PG evaluation. So there is no significant trend regarding ϵ_1. Privacy budgets of DP-Kmeans and Laplace mechanism have main influences on model quality for the DP-PGP. Specifically, when $\epsilon \geq 2$, the model quality can be commensurate with the baseline. Thus, the privacy budget of the Laplace mechanism should be at least 1.5 to ensure an acceptable model quality.

For the DP-PGC, we evaluate model quality in two main aspects, pruning ratio and parameter precision. We will denote by p_j the pruning ratio, p_c and p_r the centroid precision and the residual precision of parameters. Particularly, we will represent each model parameter using a centroid and a residual part after quantizing. Thus, centroid precision shown in the evaluation is the bit length of centroids of parameter clusters, while residual precision is the bit length for counting the distance between a centroid and a parameter's actual value. Based on the result given in Fig. 5, we can conclude that pruning ratio affects model quality directly and this result may vary regarding different DNN architectures. When we prune MNIST-Net, we cannot observe pruning effect until the pruning ratio is larger than 0.6. When we use a pruning ratio less than 0.45 for CIFAR-Net, we can observe a significant drop of model quality. For SVHN-Net, the test accuracy in Fig. 6 is relatively stable until the pruning ratio gets larger than 0.55.

On the other side, the result of parameter precision after compressing for the DP-PGC is shown in Fig. 7. We find a limit of parameter precision in the DP-PGC. When p_c and p_r are both larger than 8, model quality will not improve any more. When $p_c = 3$, $p_r = 4$, we can still yield an acceptable test accuracy. But a further precision compression will be dangerous for model quality. Since SVHN dataset is larger and harder than MNIST and CIFAR-10, the pruning procedure has a stronger influence on model usability. Fig. 6 gives a privacy budget evaluation result of DP-PG for DP-PGC with SVHN-Net. When the budget is smaller than 1, model accuracy is significantly affected. However, when the budget is larger than 1, we can have acceptable model accuracy higher than 80%. That means pruning ratio has more influences than the privacy budget for SVHN-Net. Based on our study, we recommend 35% or lower pruning ratios and precisions $p_r = 4$, $p_c = 4$ for the SVHN-Net.

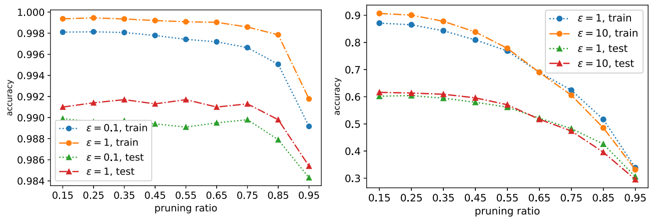


Fig. 5. Model quality evaluation for the DP-PGC of different pruning ratios with MNIST-Net (left) and CIFAR-Net (right).

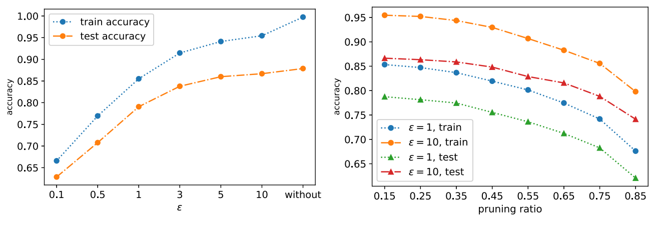


Fig. 6. Model quality evaluation for the DP-PG (left) and the DP-PGC (right) with SVHN-Net using different privacy budgets and pruning ratios.

6.2 Privacy Evaluation

Membership inference attack has been proved effective against published DNN models [9], [11], [52]. To verify privacy protection of our solutions, we investigate the adversarial effect of two kinds of membership inference attacks (black-box attack [52] and white-box attack [11]). Since membership inference attacks achieve preferable performance on CIFAR-10 dataset [52], we will mainly evaluate privacy with CIFAR-Net. Besides, as reported in [9], the attack accuracy largely depends on the over-fitting phenomenon, we will evaluate models all in well-trained states instead of different training stages.

We show the attack accuracy[1] of models published by our solutions in Fig. 8 and Fig. 9. The black-box membership inference attack can achieve near 90% accuracy on class 3. However, when we apply the DP-PG with $\epsilon = 1$ for model

1. Please note that the membership inference attack accuracy plotted here may have differences with the original work. Because experimental settings are different. But we will use the same setting strictly for all attack experiments in the paper for a fair comparison.

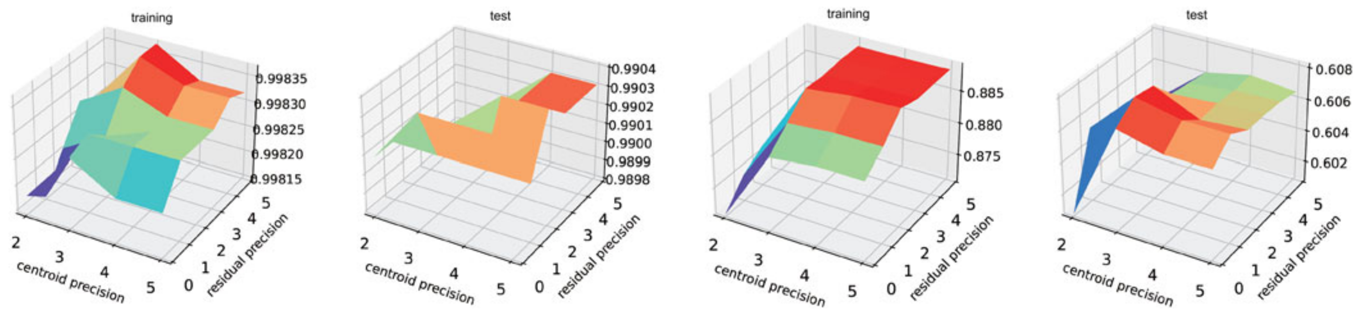


Fig. 7. Model quality evaluation for the DP-PGC with MNIST-Net (left two) and CIFAR-Net (right two) using different precision.

TABLE 2
Evaluation Results of Membership Inference Attack Against Published SVHN-Net Models

defense	white-box attack			black-box attack			test accuracy
	recall	precision	accuracy	recall	precision	accuracy	
without defense	0.867	0.566	0.602	0.764	0.563	0.586	0.879
DP-PG($\epsilon = 1$)	0.695	0.531	0.540	0.589	0.529	0.532	0.791
DP-PG($\epsilon = 10$)	0.815	0.543	0.564	0.686	0.540	0.552	0.867
DP-PGP($\epsilon_1 = 3, \epsilon_2 = 1, \epsilon_3 = 2$)	0.083	0.534	0.505	0.280	0.521	0.511	0.437
DP-PGP($\epsilon_1 = 10, \epsilon_2 = 2, \epsilon_3 = 8$)	0.761	0.539	0.555	0.6	0.529	0.532	0.833
DP-PGC($\epsilon = 1, p_j = 0.25, p_r = 2, p_c = 2$)	0.788	0.540	0.558	0.676	0.539	0.549	0.856
DP-PGC($\epsilon = 1, p_j = 0.75, p_r = 2, p_c = 2$)	0.628	0.519	0.523	0.476	0.516	0.515	0.782
DP-PGC($\epsilon = 10, p_j = 0.25, p_r = 2, p_c = 2$)	0.813	0.543	0.565	0.684	0.543	0.554	0.866
DP-PGC($\epsilon = 10, p_j = 0.75, p_r = 2, p_c = 2$)	0.620	0.515	0.519	0.462	0.513	0.511	0.807

publishing, the adversary can only obtain an inference accuracy around 65% on all classes. According to a recent study [52] on membership inference attack, when model test accuracy is 60.7% on CIFAR-10 dataset, attack accuracy can be almost 70% on two classes against the DP-SGD. When model test accuracy is 45%, attack accuracy can achieve higher than 60% on one class against the DP-SGD. However, the DP-PG can get test accuracy higher than 55% when attack accuracy is lower than 60% on all classes. The attack against models published with the DP-PGP may get better accuracy because it trades more privacy budget for model quality. When the DP-PGP uses a total privacy budget 5 ($\epsilon_1 = 5, \epsilon_2 = 1, \epsilon_3 = 4$), attack accuracy of an black-box membership inference attack is no more than 70% on any class while test accuracy is about 50% on CIFAR-10 dataset.

When we average attack accuracy of black-box inference and white-box inference across all classes, we can get an approximate attack accuracy for each solution and give the result in Fig. 9. When the privacy budget increases, defense performance of our basic solution DP-PG will drop off in both attack modes. However, defense performance of the DP-PGP and DP-PGC with various privacy budgets are

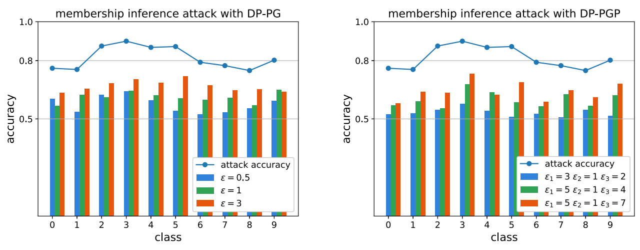


Fig. 8. Membership inference attack against different classes.

different from the DP-PG. When we use a relatively small privacy budget 3 ($\epsilon_1 = 3, \epsilon_2 = 1, \epsilon_3 = 2$) for the DP-PGP, inference accuracy will be kept under 60% in both attack modes. But we note that model quality will also be suppressed under 50%. For a better model quality, the DP-PGP has to maintain a total privacy budget around 5. As a solution dedicating on privacy, the DP-PGC can achieve better defense performance even if we use a relatively large privacy budget like 10. It is important for the DP-PGC to have a rational pruning ratio. As shown in Fig. 10, model test accuracy is above 60% if we use a pruning ratio no larger than 55%. Meanwhile, we find there is a positive correlation between parameter precision and inference accuracy, which can be observed from Fig. 11. Higher precision of centroids and residuals lead to a more accurate inference result. This phenomenon appears in both attack settings.

Moreover, we provide the evaluation result of another dataset SVHN to verify that the defense effect of our solutions is not by coincidence. In fact, the experimental result of protecting SVHN-Net is more satisfying. If we attack SVHN-Net without any protection, the accuracy reaches 60% in the white-box setting. We perform evaluations of each publishing solution in white-box and black-box settings. Table 2 gives the evaluation results. We have noticed that the attack has unstable performances in different runs. Although we average results across multiple runs, fluctuations are still recognizable. A reasonable conjecture is that the SVHN dataset has better data samples diversity, raising the difficulty of inferring specific samples. Combined with the model quality evaluation result of SVHN-Net, we can conclude that DP-PGC can preserve model privacy with the negligible expense of model quality.

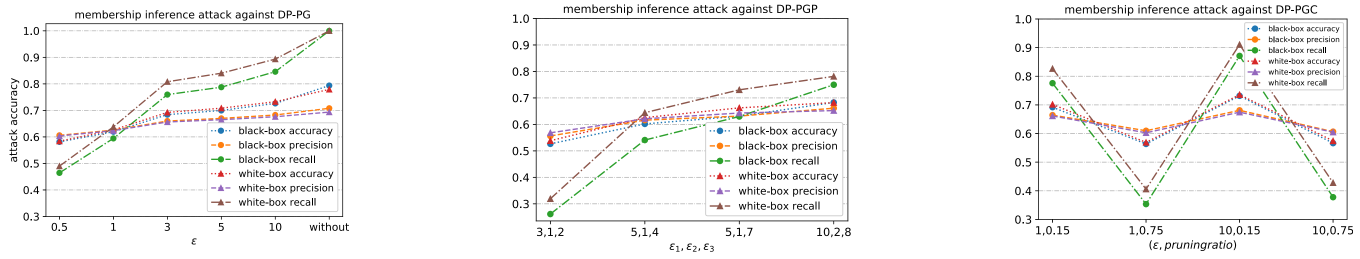


Fig. 9. Membership inference attack against the DP-PG, DP-PGP, and DP-PGC.

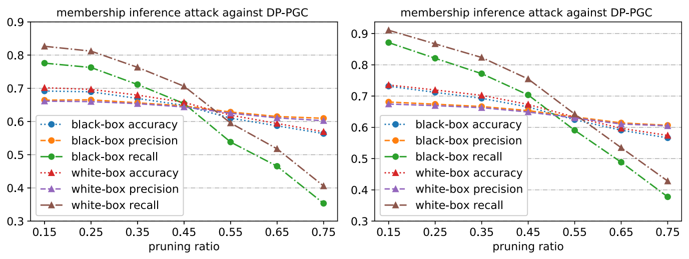


Fig. 10. Membership inference attack against published models.

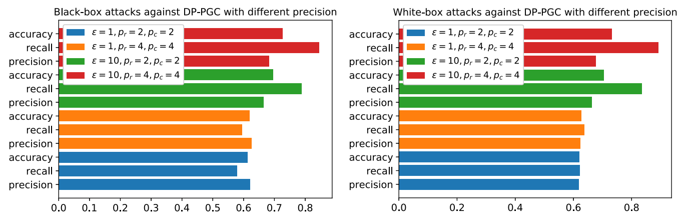


Fig. 11. Membership inference attack with different precision.

7 Conclusion

We investigate the private DNN model leakage issue in MLaaS, particularly membership inference attacks. We have observed that DNN model parameters have similar patterns in separate training tasks. Based on this observation, we propose the DP-PG for private model publishing. The DP-PG is a basic algorithm for model parameter generating, which provides differential privacy for model publishing with a customized budget. But the privacy budget of the DP-PG affects the model quality (measured by test accuracy) significantly. To moderate this situation, we design two solutions on the basis of the DP-PG for different requirements. The DP-PGP solution can achieve higher model quality growth when the privacy budget increases, while the DP-PGC solution dedicates to shrinking privacy loss. Hence, we conclude that the DP-PGP solution gives desirable model quality with a sufficient privacy budget, and the DP-PGC solution provides the most robust defense performance against both black-box and white-box inferences when model quality is not the primary requirement.

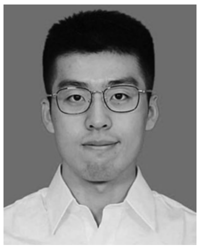
Acknowledgments

The authors would like to thank the anonymous reviewers for the time and efforts they have kindly made. A preliminary version [1] has been published in the proceedings of the 28th International Symposium on Quality of Service (IEEE/ACM IWQoS 2020).

References

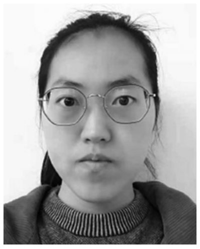
[1] Y. Mao, B. Zhu, W. Hong, Z. Zhu, Y. Zhang, and S. Zhong, "Private deep neural network models publishing for machine learning as a service," in *Proc. IEEE/ACM 28th Int. Symp. Qual. Service*, 2020, pp. 1–10.
[2] V. N. Ioannidis, A. S. Zamzam, G. B. Giannakis, and N. D. Sidiropoulos, "Coupled graphs and tensor factorization for recommender systems and community detection," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 909–920, Mar. 2021.
[3] Y. Mao, J. Feng, F. Xu, and S. Zhong, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Top. Edge Comput.*, 2018, pp. 1–6.
[4] S. Zhou *et al.*, "Inferring emotion from large-scale internet voice data: A semi-supervised curriculum augmentation based deep learning approach," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 2–9.
[5] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6231–6239.
[6] Y. You, A. Buluç, and J. Demmel, "Scaling deep learning on GPU and knights landing clusters," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2017, pp. 1–12.
[7] PyTorch, "PyTorch Hub for research models." Accessed: Jun. 20, 2021. [Online]. Available: https://pytorch.org/hub/research-models
[8] Google, "Cloud inference API." Accessed: Jun. 20, 2021. [Online]. Available: https://cloud.google.com/inference
[9] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[10] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 634–646.
[11] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.
[12] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 691–706.
[13] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 619–633.
[14] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.
[15] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.
[16] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and K. Ren, "GANobfuscator: Mitigating information leakage under GAN via differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2358–2371, Sep. 2019.
[17] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2020.
[18] Y. Mao, W. Hong, H. Wang, Q. Li, and S. Zhong, "Privacy-preserving computation offloading for parallel deep neural networks training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1777–1788, Jul. 2021.
[19] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *Proc. 28th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–17.

[20] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1964–1974.
[21] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.
[22] Z. He and J. Zhou, "Inference attacks on genomic data based on probabilistic graphical models," *Big Data Mining Analytics*, vol. 3, no. 3, pp. 225–233, 2020.
[23] H. Bae *et al.*, "Security and privacy issues in deep learning," 2021, *arXiv:1807.11655*.
[24] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[25] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[26] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 332–349.
[27] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: An application of human behavior prediction," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1309–1316.
[28] N. Phan, X. Wu, and D. Dou, "Preserving differential privacy in convolutional deep belief networks," *Mach. Learn.*, vol. 106, pp. 1681–1704, 2017.
[29] N. Papernot, M. Abadi, U. Erlingsson, I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–16.
[30] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with PATE," in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–34.
[31] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, "Measuring the effects of data parallelism on neural network training," *J. Mach. Learn. Res.*, vol. 20, pp. 1–49, 2019.
[32] J. H. Park *et al.*, "HetPipe: Enabling large DNN training on (Whimpy) heterogeneous GPU clusters through integration of pipelined model parallelism and data parallelism," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 307–321.
[33] F. Zhang, Z. Chen, C. Zhang, A. C. Zhou, J. Zhai, and X. Du, "An efficient parallel secure machine learning framework on GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2262–2276, Sep. 2021.
[34] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, pp. 211–407, 2014.
[35] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[36] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci.*, 2007, pp. 94–103.
[37] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 225–240.
[38] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *J. Roy. Statist. Soc. Ser. B Methodol.*, vol. 53, pp. 683–690, 1991.
[39] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1990, pp. 598–605.
[40] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[41] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu, "Compressing convolutional neural networks via factorized convolutional filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3972–3981.
[42] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private k-means clustering," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy*, 2016, pp. 26–37.
[43] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–14.
[44] J. Liu *et al.*, "Discrimination-aware network pruning for deep model compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2021.3066410.
[45] A. Abdi and F. Fekri, "Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3105–3112.
[46] J. Chen, S. Chen, and S. J. Pan, "Storage efficient and dynamic flexible runtime channel pruning via deep reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 14747–14758.
[47] K. Talwar and F. McSherry, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci.*, 2007, pp. 94–103.
[48] Keras, "Keras code examples: Computer vision." Accessed: Jun. 20, 2021. [Online]. Available: https://keras.io/examples/
[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[50] A. Krizhevsky, "Learning multiple layers of features from tiny images." Accessed: Jun. 20, 2021. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[51] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5330–5340.
[52] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model," *Trans. Data Privacy*, vol. 11, pp. 61–79, 2018.



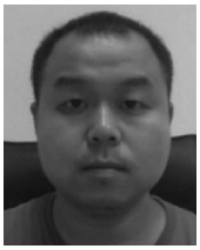
Yunlong Mao (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2013 and 2018, respectively. He is currently an assistant researcher with the Department of Computer Science and Technology, Nanjing University. His current research interests include security, privacy, and machine learning.

Wenbo Hong (Student Member, IEEE) is currently working toward the graduate degree with the Department of Computer Science, Nanjing University, Nanjing, China. His research interests include data privacy and deep learning.



Boyu Zhu (Student Member, IEEE) received the BS degree in software engineering from Donghua University, Shanghai, China, in 2016. She is currently working toward the PhD degree with the Department of Computer Science and Technology, Nanjing University, Nanjing, China. Her research interests include cryptography and algorithms.

Zhifei Zhu (Student Member, IEEE) is currently working toward the undergraduate degree with the Department of Computer Science, Nanjing University, Nanjing, China. His research interests include security and machine learning.



Yuan Zhang (Member, IEEE) received the BS degree in automation from Tianjin University, Tianjin, China, in 2005, the MSE degree in software engineering from Tsinghua University, Beijing, China, in 2009, and the PhD degree in computer science from the State University of New York at Buffalo, Buffalo, New York, in 2013. His research interests include security, privacy, and economic incentives.

Sheng Zhong (Member, IEEE) received the BS and MS degrees from Nanjing University, Nanjing, China, in 1996 and 1999, respectively, and the PhD degree from Yale University, New Haven, Connecticut, in 2004, all in computer science. His research interests include security, privacy, and economic incentives.