A Differentially Private Framework for Deep Learning with Surrogate Loss Functions

Zhigang Lu, Hassan Jameel Asghar, Dali Kaafar, Darren Webb, and Peter Dickinson

Abstract—Differential privacy (DP) has been applied in deep learning for preserving privacy of the underlying training datasets. Existing DP practice falls into three categories [1] - objective perturbation (injecting DP into objective function), gradient perturbation (injecting DP into the process of gradient descent) and output perturbation (injecting DP into the trained neural networks, scaled by the global sensitivity of the trained model parameters). They suffer from three main problems. First, conditions on objective functions limit objective perturbation in general deep learning tasks. Second, gradient perturbation does not achieve satisfactory privacy-utility trade-off due to over-injected noise in each epoch. Third, high utility of the output perturbation method is not guaranteed because of the loose upper bound of the global sensitivity of the trained model parameters as the noise scale parameter. To address these problems, we analyse a tighter upper bound of the global sensitivity of the model parameters. Under a black-box setting, based on this global sensitivity, to control the overall noise injection, we propose a novel output perturbation framework by injecting DP noise into a randomly sampled neuron (by the exponential mechanism of DP) at the output layer of a baseline non-private neural network trained with a convex surrogate loss function. We empirically compare the privacy-utility trade-off, measured by accuracy loss to baseline non-private models and the privacy leakage against black-box membership inference (MI) attacks [2], between our framework and the open-source differentially private stochastic gradient descent (DP-SGD) approaches on six commonly used real-world datasets. The experimental evaluations show that our approach, injecting DP noise into an individual neuron at the output layer, outperforms DP-SGD on privacy-utility trade-off when the baseline models have observable privacy leakage under MI attacks.

Index Terms—Differential privacy, Membership inference attack, Deep learning, Surrogate loss function.

1 Introduction

Membership Inference (MI) attacks are pose a serious privacy risk to Machine Learning-as-a-Service (MLaaS). In a nutshell, MI attacks apply a binary classifier using prediction vectors of an observation from a non-private machine learning (ML) model to infer membership of the data samples. In this paper we specify the baseline ML technique as deep learning implemented by artificial neural networks. There are two general types of MI attacks, black-box MI attacks [2] and white-box MI attacks [3], depending on adversary's access to the target ML model trained on some training dataset. Compared to the white-box MI attacks, the black-box attacks require less [3] (or even no [4]) auxiliary information about the training data to the online MLaaS, which does not release the model details but only the queried prediction output. Our focus is on providing privacy against black-box MI attacks.

To prevent membership disclosure, differential privacy (DP) [5] is a promising technique, which shields the existence of any arbitrary data sample in a dataset, i.e., preserving membership privacy for each member in the training dataset. At this stage, there are three categories of injecting DP into deep learning - objective perturbation, gradient perturbation and output perturbation- according to Jayaraman et al. [1]. In particular, objective perturbation injects DP noise into the objective function of a learning task [6]; gradient perturbation injects DP noise into each epoch during gradient descent [7]; output perturbation injects DP noise to the elements (edges or nodes) of a trained non-private neural network [8], [9] or into the prediction results following the sample-and-aggregate mechanism of DP [10]. Due to the non-convexity of the loss function, applying objective perturbation and output perturbation mainly rely on convexification techniques. For example, Phan et al. [11], [12] convexify the loss function of convolutional deep belief networks [13], then inject DP noise into the coefficients via the functional mechanism (originally for non-deep learning tasks) [6]. More generally, we could implement output perturbation by training a baseline non-private neural network with a universal convexified surrogate loss function [14], [15], then injecting DP noise into the elements of the trained network.

However, we found some weaknesses that limit the application/performance of existing DP applications. Existing objective perturbation approaches (combining convexified loss function [11], [12] and DP objective function [6]) only work for a specified learning task - convolutional deep belief network [13], which makes it hard to apply it to general deep learning tasks. The gradient perturbation approaches (including different DP variants and composition theorems) suffer from over-injected noise, mainly since the overall noise injection depends on the number of training epochs, which are usually large in deep learning [16]. PATE [10], the only work implementing sample and aggregate mechanism of DP for output perturbation, works in a special configuration requiring additional publicly available data to assist differentially private aggregation of the distributed learning outputs. The output perturbation framework (uni-

- *Z. Lu, H. J. Asghar and D. Kaafar are with Department of Computing, Macquarie University, Sydney, Australia.*
  E-mail: {zhigang.lu, hassan.asghar, dali.kaafar}@mq.edu.au
- *D. Webb and P. Dickinson are with Cyber & Electronic Warfare Division, Defence Science & Technology Group, Edinburgh, Australia.*
  E-mail: {darren.webb,peter.dickinson}@dst.defence.gov.au

Manuscript received Month DD, YYYY; revised Month DD, YYYY.

TABLE 1: Comparison between the Upper Bounds of the L_2 Global Sensitivity of Trained Model Parameters. (ρ-Lipschitz loss functions, λ-strongly convex L_2 regularisers, η: constant SGD step, C: number of labels, $\|\mathbf{x}\|_2$: maximum L_2 norm of a data sample in the training data space, n: size of the training dataset.)

Work	Upper Bound	Conditions				
		Convex loss	Smooth loss	Step size in SGD	Normalised data	Binary labels
Chaudhuri et al. [8]	$2\rho/\lambda n$	✓	-	-	✓	✓
Chaudhuri et al. [8]	$C\|\mathbf{x}\|_2\rho/\lambda n$	✓	-	-	-	-
Wu et al. [9]	$2\rho/\lambda n$	✓	✓	decreasing	-	-
Wu et al. [9]	$2\eta\rho/(1-(1-\eta\lambda)^n)$	✓	✓	constant	-	-
This Work	$2\rho/\lambda n$	✓	-	-	-	-

versal convexification plus DP noise injection) relies on tight upper bound of the DP noise scale parameter (the global sensitivity). Existing theoretical results [8], [9] provide loose upper bounds assuming convexity of the loss function. To tighten their results, more conditions should be introduced, in addition to convex loss function, such as normalised training datasets with binary label [8] or smooth loss functions with a decreasing step size during the process of stochastic gradient descent (SGD) [9]. We show the conditions around existing upper bounds in Table 1.

Contributions. To address the problems in existing DP practice in deep learning, we propose a novel DP framework in the black-box setting, where the model parameters are not released. In summary, we list our main contributions below.

1) We mathematically analyse a tight upper bound of the global sensitivity of the trained model parameters, which assumes a convex loss function as existing upper bounds, but without other conditions (see Table 1 for a brief comparison). Different from existing works, we bound the maximum change of the model parameters via studying the stability of a trained model by removing an arbitrary data sample from a training dataset.
2) We propose a novel framework in the black-box settings, where DP noise is injected into an individual neuron at the output layer of a trained neural network. Specifically, at the training stage (for a baseline non-private model), we apply a universal convexification for loss function [14], [15] to meet the convexity condition. At the prediction stage (for a differentially private prediction probability vector), we first sample a neuron with the exponential mechanism of DP (Exp-DP) [17] at the output layer then inject DP noise into the sampled neuron, where we scale the noise by the global sensitivity of an individual neuron (bounded by the global sensitivity of the trained model parameters).
3) We empirically compare the privacy-utility trade-off of our framework with existing open-source differentially private stochastic gradient descent (DP-SGD) approaches for classification on six commonly used real-world datasets, following the same experimental configurations as existing studies. The experimental results show that our framework achieves better privacy-utility trade-off than existing DP-SGD implementations when the baseline non-private models have observable privacy leakage under MI attacks.

2 Related Work

Currently, DP implementations in deep learning are mainly categorised into objective perturbation, gradient perturbation and output perturbation [1]. The difference between the three categories is at the point where we inject DP noise. Objective perturbation injects DP noise into the objective function; gradient perturbation injects DP noise into the process of gradient descent; output perturbation injects DP noise to the output, i.e., the elements of a trained neural network.

Specific to deep learning, there are two works achieving DP for the objective perturbation [1]. Phan et al. [11], [12] replace the non-convex loss function with polynomial approximations via Taylor expansions, then inject DP noise into the coefficients of the approximate function via the functional mechanism [6]. However, such a framework only works on a specific machine learning task - convolutional deep belief network [13], which makes it hard to apply in general deep learning tasks. Additionally, it is hard to find the optimal order of polynomial degree to achieve accepted trade-off between utility and privacy in practice.

Gradient perturbation is the most famous DP practice in the field of deep learning (DP-SGD) [7]), including an open-source Python library provided by Google's tensorflow-privacy [18]. DP-SGD perturbs the minibatch stochastic optimisation process, i.e., injecting DP noise into the gradients of each epoch during the training process. Specifically, in each epoch, DP-SGD bounds the global sensitivity of the gradients (maximum change of the gradients by an individual data samples) by norm clipping. However, since DP-SGD injects noise into all the epochs and the number of epochs is usually a large number in deep learning, the overall performance of DP-SGD would not be guaranteed.

For output perturbation, there are two main streams of techniques. One line of works applies the sample-and-aggregate mechanism of DP to produce differentially private prediction probability vectors. For example, Papernot et al. [10] propose a framework for differentially private aggregate machine learning, called Private Aggregation of Teacher Ensembles (PATE). Specifically, PATE splits the original dataset into several disjoint subsets which are inputs for training the sub-models. When new observation comes for prediction, PATE aggregates the predicted labels of each sub-model with injected noise following Gaussian distribution for the final DP prediction output. However, since the splitting scheme of PATE makes the size of each subset smaller than the original dataset, each sub-model would have higher chance to be overfitted, which impacts the prediction performance of PATE. Additionally, PATE works on a special assumption that some publicly accessible data following the same distribution as the private data should be prepared. Such an assumption further limits the use of PATE in general privacy-preserving tasks where data curators do not have publicly available data [16].

The other line of work in output perturbation is to inject DP noise into the baseline non-private neural networks trained with a convexified surrogate loss function. This way, we could analyse the upper bound of the global sensitivity for noise injection with the property of convexity. At this stage, there are two main theoretical results [8], [9] for the upper bound of the global sensitivity of the trained model parameters. Chaudhuri et al. [8] and Wu et al. [9] provide loose upper bound meeting the convexity condition of the loss function. To tighten their results, more conditions should be introduced, such as normalised training data for binary classification tasks [8] or smooth loss function with decreasing step size during the SGD process [9]. Injecting noise, scaled by the loose upper bounds of the global sensitivity, cannot guarantee a satisfying privacy-utility trade-off.

Based on the above analysis, we conclude that existing DP approaches in objective perturbation, gradient perturbation and output perturbation suffer from three problems. First, conditions on objective functions limit the objective perturbation in general deep learning tasks. Second, gradient perturbation does not achieve a satisfactory privacy-utility trade-off due to over-injected noise in each epoch. Third, the utility of the output perturbation is not guaranteed because of the loose upper bounds of the global sensitivity of the trained model parameters as the noise scale parameter. Therefore, in this paper, we aim to provide a DP framework of output perturbation to control the overall noise injection, achieving better trade-off between utility and privacy, in general deep learning tasks.

3 Preliminaries

In this section, we briefly introduce the privacy notion used in this paper, i.e., differential privacy, machine learning concepts and membership inference (MI) attacks. A dataset X is a collection of *rows*: $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each $\mathbf{x}_i$ is from some fixed data domain. Two datasets X and X' are called neighbouring datasets, denoted $X \sim X'$, if one is obtained from the other by adding or removing a single row. Given a dataset $X = (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n)$, $X^{(-i)}$ represents the dataset $X^{(-i)} = (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n)$, i.e., the neighbouring dataset of X with an arbitrary row $\mathbf{x}_i$ removed. Similarly, $X^{(i)}$ denotes the dataset $X \setminus \{\mathbf{x}_i\} \cup \{\mathbf{x}'\} = (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}', \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n)$, i.e., the row $\mathbf{x}_i$ replaced with an arbitrary row $\mathbf{x}'$. Note that the Hamming distance between X and $X^{(-i)}$, $d(X, X^{(-i)}) = 1$ and datasets X and $X^{(i)}$ are *not* neighbouring datasets according to our definition.

3.1 Differential Privacy

Informally, differential privacy (DP) [5] ensures that two outputs, produced by a randomised algorithm on two neighbouring datasets are almost indistinguishable.

Definition 1 (ϵ-DP [5]). *A randomised mechanism $\mathcal{T}$ is ϵ-differentially private if for all neighbouring datasets X and $X^{(-i)}$, $i \sim U(n)$, and for all outputs $S \in Range(\mathcal{T})$, $\mathcal{T}$ satisfies:*

$$\Pr[\mathcal{T}(X) = S] \leq \exp(\epsilon) \times \Pr[\mathcal{T}(X^{(-i)}) = S], \tag{1}$$

where $\epsilon > 0$ is the privacy budget.

Two prototypical ϵ-DP mechanisms considered in this paper are the Laplace mechanism (Lap-DP) [5] and the exponential mechanism (Exp-DP) [17]. The former adds additive noise drawn from the Laplace distribution of scale $\Delta f/\epsilon$ to the numeric computation $f(\cdot)$, where Δf is the L_1 global sensitivity of f. The L_p global sensitivity of f is:

$$\Delta_p f = \max_{\forall X \in \mathcal{D}, d(X, X^{(-i)})=1} \|f(X) - f(X^{(-i)})\|_p. \tag{2}$$

where $\|\cdot\|_p$ is the L_p-norm. For simplicity, in this paper, Δf denotes the L_1 global sensitivity.

The Exp-DP mechanism is a weighted sampling scheme to generate the output from a set of (arbitrary) candidates, and is well-suited for non-numeric computations. The mechanism simply outputs a given candidate r from a set of candidates R computed over a dataset X, with probability proportional to $\exp(\epsilon q(X, r)/2\Delta q)$, where $q(X, r)$ is the score/quality function of the candidate r, and Δq is the maximum possible change in q over all neighbouring datasets.

An important property of differential property is that the mechanisms compose [19]:

Theorem 1 (Sequential Composition [19]). *Let $M_i(X)$ provide ϵ_i-DP. Then the sequence of $M_i(X)$ provides $\sum_i \epsilon_i$-DP.*

A recently proposed ϵ-Gaussian DP (ϵ-GDP), injecting noise following a Gaussian distribution $\mathcal{N}(0, (^{\Delta f}/_{\epsilon})^2)$, is a relaxation of traditional ϵ-DP. We have

Theorem 2. *[20] A mechanism is ϵ-GDP if and only if is is $(\epsilon, \delta(\epsilon))$-DP, $\forall \epsilon > 0$, where $\delta(\epsilon) = \Phi(-1 + {}^{\epsilon}/_{2}) - \exp(\epsilon)\Phi(-1 - {}^{\epsilon}/_{2})$, $\Phi(\cdot)$ is the cumulative distribution function of $\mathcal{N}(0, 1)$.*

The composition of GDP is shown as follow.

Theorem 3 (Composition of GDP [20]). *The n-fold composition of ϵ_i-GDP is $\sqrt{\sum_{i=1}^{n} \epsilon_i^2}$-GDP, where each ϵ_i-GDP injects Gaussian noise following $\mathcal{N}(0, (^{\Delta f}/_{\epsilon_i})^2)$.*

3.2 Machine Learning Concepts

Let A be a learning algorithm, $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a training dataset drawn from a distribution $\mathcal{D}$ ($X \sim \mathcal{D}^n$), $W_X (= (\omega_1, \ldots, \omega_{|E|}) \in \mathbb{R}^{|E|}) = A(X)$ be the output of the learning algorithm A on the training dataset X, i.e., trained model parameters, where $|E|$ is the number of model parameters. Once the model parameters are set after training, we predict the label y of an observation $\mathbf{x}$ as $y = \arg\max\{p_1, \ldots, p_C\}$, where $\mathbf{p} = \{p_1, \ldots, p_C\} = h(W_X, \mathbf{x})$, h is the learnt hypothesis and C is the number of labels.

We use a loss function $l(W_X, \mathbf{x}_i)$ on an observation $\mathbf{x}_i \in X \sim \mathcal{D}^n$ to measure the empirical error of W_X on $\mathbf{x}_i$. Note that each $\mathbf{x}_i$ is a composition of a set of features and an accompanying class label. Accordingly, we have the following equation to measure the training error of the learning algorithm A on the dataset X. The learning algorithm A aims to minimise the training error.

$$L_X(W_X) = \frac{1}{|X|}\sum_{i=1}^{|X|} l(W_X, \mathbf{x}_i) \tag{3}$$

Measurement for overfitting. A fundamental issue to avoid in machine learning is overfitting of a learning algorithm. One way to measure overfitting of a learning algorithm is the notion of On-Average-Replace-One stability of a model [21]. That is, replacing a single data sample does not result in a large difference on the value of the loss function.

Definition 2 (On-Average-Replace-One Stability [21]). *Let $\sigma : \mathbb{N} \to \mathbb{R}$ be a monotonically decreasing function. We say that a learning algorithm A is on-average-replace-one stable with rate $\sigma(n)$ if for every distribution $\mathcal{D}$,*

$$\mathbb{E}_{(X,\mathbf{x}')\sim\mathcal{D}^{n+1}, i\sim U(n)}[l(W_{X^{(i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)] \le \sigma(n), \tag{4}$$

where $X^{(i)} = X \setminus \{\mathbf{x}_i\} \cup \{\mathbf{x}'\}$, and $U(n)$ is the uniform distribution over $[n]$.

We extend On-Average-Replace-One stability to On-Average-Remove-One stability to fit the neighbouring dataset requirement of Definition 1.

Definition 3 (On-Average-Remove-One (OARO) Stability). *Given a monotonically decreasing function $\sigma : \mathbb{N} \to \mathbb{R}$, an arbitrary distribution $\mathcal{D}$, then a learning algorithm A is on-average-remove-one stable with rate $\sigma(n)$ if*

$$\mathbb{E}_{X\sim\mathcal{D}^n, i\sim U(n)}[l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)] \le \sigma(n), \tag{5}$$

where $U(n)$ is the uniform distribution over $[n]$.

Topology of deep neural networks. A deep neural network is a graph, $G = (V, E)$, formed by connecting $T+1$ layers. A layer V_t ($t \in [0, T]$) is a set of disconnected vertices, $V_t = \{v_{t,1}, v_{t,2}, \ldots, v_{t,|V_t|}\}$. Each vertex is a neuron in the neural network, one of the neurons serves as the bias term. The vertices set V is a union of disjoint layers, $V = \bigcup_{t=0}^{T} V_t$ and the edges set E is the set of weighted edges connecting vertices in two adjacent layers. For example, an edge $e_{t,i,j}$ (with a weight $\omega_{t,i,j}$) connects $v_{t-1,j}$ and $v_{t,i}$. We call V_0 input layer, V_t ($t \in [1, T-1]$) hidden layers and V_T output layer. Note that $|V_0| = m + 1$, where m is the dimension of a data sample, $|V_T| = C$, where C is the number of labels, $|E| = |W_X|$ is the number of model parameters.

When using a trained neural network (trained model parameters/weights and given topology) to make prediction for a given data sample $\mathbf{x} = (x_1, \ldots, x_n)$, we calculate the output of each vertex/neuron, $a_{t,i}$ at layer V_t as follows.

$$\begin{cases} z_{t,i} &= \sum_{\forall e_{t,i,j}} \omega_{t,i,j} a_{t-1,j}, \\ a_{t,i} &= \phi(z_{t,i}), \end{cases} \tag{6}$$

where $\phi(\cdot)$ is an activation function for the neurons, $a_{0,i} = x_i$ and $a_{t-1,|V_{t-1}|}$ is also known as the bias term. At the output layer, the final prediction probability vector of the given data sample $\mathbf{x}$ would be $\mathbf{p} = (a_{T,1}, \ldots, a_{T,C})$, where $\sum_{i=1}^{C} a_{T,i} = 1$. In our experiments in Section 5, $\phi(\cdot)$ is Tanh for $t < T$, $\phi(\cdot)$ is Softmax for $t = T$.

Furthermore, we also use the concepts of *convex set*, *convex function*, *λ-strong convexity* and *ρ-Lipschitzness* in this paper. We follow the standard definitions of these concepts.

Definition 4 (Convex set [21]). *X is a convex set if and only if for any two vector $\mathbf{u}$, $\mathbf{v} \in X$ and an $\alpha \in [0, 1]$, we have $\alpha\mathbf{u} + (1-\alpha)\mathbf{v} \in X$.*

Definition 5 (Convex function [21]). *A function $f : X \to \mathbb{R}$ is convex if X is convex and for any two vectors $\mathbf{u}$ and $\mathbf{v}$ in X, we have*

$$f(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \le \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v}). \tag{7}$$

Definition 6 (λ-Strong convexity [21]). *A convex function $f : X \to \mathbb{R}$ is λ-strongly convex if for any two vectors $\mathbf{u}$ and $\mathbf{v}$ in X and $\alpha \in [0, 1]$, we have*

$$f(\alpha\mathbf{u}+(1-\alpha)\mathbf{v}) \le \alpha f(\mathbf{u})+(1-\alpha)f(\mathbf{v})-\frac{\lambda}{2}\alpha(1-\alpha)\|\mathbf{u}-\mathbf{v}\|_2^2. \tag{8}$$

Definition 7 (ρ-Lipschitzness [21]). *A function $f : X \to \mathbb{R}^d$ is ρ-Lipschitz over X if for any two vectors $\mathbf{u}$ and $\mathbf{v}$ in X, we have*

$$\|f(\mathbf{u}) - f(\mathbf{v})\| \le \rho\|\mathbf{u} - \mathbf{v}\|. \tag{9}$$

3.3 Membership Inference Attacks

The goal of Membership Inference (MI) Attacks is to infer the membership of a given data sample by the prediction output of the data sample. In this paper we specify MI attacks as black-box MI attacks implemented in shadow model as Shokri et al. [2]. That is, the MI adversaries only have access to the distribution of the target/private dataset and the prediction output of a given data sample.

For simplicity, a black-box MI attack in shadow model is a function $\mathcal{A} : \mathcal{D}^{kn+1} \to \{0, 1\}$. The adversary performs MI attacks (shadow model approach) in three steps. Firstly, the adversary trains k shadow models with shadow dataset $X'_i \sim \mathcal{D}^n$, $i \in [1, k]$, mimicking the behaviour of the target model trained with the target/private dataset X. Then the adversary uses the k shadow models to make predictions for training data (members) and test data (non-members). These prediction vectors will then be the training data to train an attack model (a binary classification). Finally, for a given data sample $\mathbf{x} \sim \mathcal{D}$, the adversary queries the prediction probability vector $\mathbf{p}$ of $\mathbf{x}$ from the target model, then feeds $\mathbf{p}$ to the trained attack model to predict the membership ($\{0, 1\}$) of $\mathbf{x}$.

Finally, Table 2 shows the summary of notations of this paper.

4 Differential Privacy via Output Perturbation and Theoretical Results

In this section, we show our approach of implementing output perturbation, then mathematically analyse the properties of the proposed approach, such as global sensitivity of model parameters and neurons, effect of differential privacy against membership inference attacks.

TABLE 2: Summary of Notations.

Notation	Description
A	Learning algorithm
C	Number of classes/labels
$\mathcal{D}$	Data distribution
ϵ	Privacy budget
$e_{t,i,j}$	Edge connecting $v_{t-1,j}$ and $v_{t,i}$
$G=(V,E)$	Neural network
λ	Strongly convex constant
$l(W_X, \mathbf{x})$	Loss function on $\mathbf{x}$
$L_X(W_X)$	Training error on X
m	Dimension of the training dataset
n, $\lvert X\rvert$	Size of the training dataset
ρ	Lipschitzness constant
$U(n)$	Uniform distribution over $[1,n]$
V_t	Neurons set at layer t
W_X, $A(X)$	Model parameters trained on X
$\Delta_2 W$	L_2 global sensitivity of weights vector
$\Delta\omega$	L_1 global sensitivity of an individual weight
$\omega_{t,i,j}$	Weight on edge $e_{t,i,j}$
$v_{t,i}$	ith neuron at tth layer in G
X	Training dataset
$X^{(-i)}$	Neighbouring dataset of X
$\mathbf{x}$	Data sample (vector) of X
$\mathbf{x}\sim\mathcal{D}$	$\mathbf{x}$ drawn from distribution $\mathcal{D}$
Δz_T	L_1 global sensitivity of a neuron (output layer)
$\lVert\mathbf{a},\mathbf{b}\rVert_p$	L_p distance between $\mathbf{a}$ and $\mathbf{b}$
$\lvert S\rvert$	Cardinality of a set S

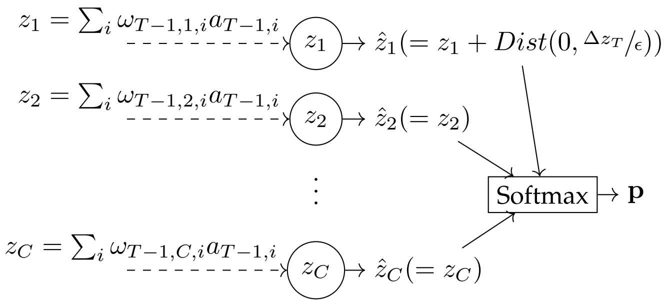

$z_1 = \sum_i \omega_{T-1,1,i} a_{T-1,i}$
z_1
$\hat{z}_1(= z_1 + Dist(0, \Delta z_T/\epsilon))$
$z_2 = \sum_i \omega_{T-1,2,i} a_{T-1,i}$
z_2
$\hat{z}_2(= z_2)$
Softmax
$\mathbf{p}$
$z_C = \sum_i \omega_{T-1,C,i} a_{T-1,i}$
z_C
$\hat{z}_C(= z_C)$

Fig. 1: Idea Overview (assume z_1 is the sampled neuron).

4.1 Idea Overview

Based on our analysis in Section 2, output perturbation provides us a way to control the amount of injected noise while satisfying DP guarantee in general deep learning tasks. To minimise the noise injection while guaranteeing DP of the prediction probability vector of any given data sample, we propose injecting DP noise into a randomly sampled neuron at the output layer. Figure 1 depicts our general idea, say neuron z_1 is the sampled neuron, where the noise distribution is $Dist(0, {}^{\Delta z_T}/_{\epsilon})$).

Specifically, our DP prediction works in three steps. Firstly, for a given data sample, we feed this data sample to a trained neural network to calculate the values for the neurons at the output layer. Next, we inject DP noise into a randomly sampled neuron at the output layer. After noise injection, we apply Softmax function on the noisy neuron vector to produce a differentially private probability vector. Algorithm 1 shows the implementation details, where Line 1 to Line 7 are the first step, Line 8 to Line 10 are the second step, and Line 11 and Line 13 are the third step.

Algorithm 1: DP Prediction Probability Vector for Non-Private Deep Learning Models.

```
Input : G = (V, E): an artificial neural network; C: number of labels; W_X = {ω_{t,i,j} | t ∈ [1,T], i ∈ [1,|V_t|], j ∈ [1,|V_{t-1}|]}: trained model parameters on dataset X; φ(·): activation function at hidden layers; ε: privacy budget for neuron sampling and noise injection; Δz_T: L_1 global sensitivity of the neurons at layer T; x: an observation for predicting label.
Output: p = (p_1, ..., p_C): Differentially Private prediction probability vector.
1  $\mathbf{a}_0 \leftarrow \mathbf{x}$;
2  for $t \leftarrow 0$ to $T-1$ do
3      for $i \leftarrow 1$ to $|V_{t+1}|$ do
4          if $t \neq T-1$ then
5              $a_{t+1,i} \leftarrow \phi(\sum_{j=1}^{|V_t|} \omega_{t+1,i,j} a_{t,j})$;
6          else
7              $z_{t+1,i} \leftarrow \sum_{j=1}^{|V_t|} \omega_{t+1,i,j} a_{t,j}$;
8  $\mathbf{p} \leftarrow$ Softmax$(z_{T,1}, \dots, z_{T,C})$;
9  $v \leftarrow$ Exp-DP$(\{z_{T,1}, \dots, z_{T,C}\}, \Pr[v] \propto \exp({}^{\epsilon p_v}/_{2\Delta p}))$;
10 $\hat{z}_{T,v} \leftarrow z_{T,v} + Dist(0, {}^{\Delta z_T}/_{\epsilon})$;
11 $(\hat{z}_{T,1}, \dots, \hat{z}_{T,C}) \leftarrow (z_{T,1}, \dots, \hat{z}_{T,v}, \dots, z_{T,C})$;
12 $\mathbf{p} \leftarrow$ Softmax$(\hat{z}_{T,1}, \dots, \hat{z}_{T,C})$;
13 return: $\mathbf{p}$;
```

The key point of implementing Algorithm 1 is to find a tight upper bounds of Δz_T for deep learning models. In the following sections, we show our method to calculate the upper bounds of Δz_T by analysing the upper bound of the global sensitivity of the model parameters, $\Delta_2 W$, which is tighter than existing studies [8], [9] under the condition of convex loss function. In addition, we show the DP guarantee of Algorithm 1 and the effect of DP against the black-box MI attacks.

4.2 Analysis of the Upper Bound of Global Sensitivity

This section studies the upper bounds of the L_2 global sensitivity of trained model parameters and the L_1 global sensitivity of an individual neuron at the output layer. In brief, we analyse these upper bounds by the fundamental properties of convex and strongly convex functions.

Lemma 1. *[21] The L_2-norm regulariser $2\lambda\|W_X\|_2^2$ is λ-strongly convex.*

Lemma 2. *[21] Let function h_i be convex, function g be λ-strongly convex. Their linear composition $f = \frac{1}{n}\sum_{i=1}^{n} h_i + g$ is λ-strongly convex.*

Lemma 3. *[21] Let $\mathbf{u}$ be a minimiser of a λ-strongly convex function f ($f'(\mathbf{u}) = 0$), then for any $\mathbf{v}$, we have*

$$f(\mathbf{v}) - f(\mathbf{u}) \geq \frac{\lambda}{2}\|\mathbf{v}-\mathbf{u}\|_2^2. \tag{10}$$

Based on Lemma 1, Lemma 2 (when h is the loss function and g is the L_2-norm regulariser) and Lemaa 3, we have the following theorem to calculate the global sensitivity of the model parameters, where we measure the global sensitivity of the model parameters by L_2-norm (a.k.a. L_2-sensitivity [19]) and the upper bound of the On-Average-Remove-One stability of the loss function.

Theorem 4. *Given a convex and ρ-Lipschitz loss function and a λ-strongly convex regulariser, then the upper bound of the L_2 global sensitivity of the model parameters $\Delta_2 W$ is $2\rho/\lambda n$; the On-Average-Remove-One stability of the loss function $l(W_X, \mathbf{x})$ is bounded by $2\rho^2/\lambda n$.*

Proof. Based on Lemma 1, Lemma 2 and Lemma 3, we have

$$\begin{aligned}
&\|W_{X^{(-i)}} - W_X\|_2^2 \\
\leq&\frac{2}{\lambda}\left(L_X(W_{X^{(-i)}}) + \frac{\lambda}{2}\|W_{X^{(-i)}}\|_2^2\right) \\
&- \frac{2}{\lambda}\left(L_X(W_X) + \frac{\lambda}{2}\|W_X\|_2^2\right) &(11)\\
=&\frac{2}{\lambda}\left(L_{X^{(-i)}}(W_{X^{(-i)}}) + \frac{\lambda}{2}\|W_{X^{(-i)}}\|_2^2\right) \\
&- \frac{2}{\lambda}\left(L_{X^{(-i)}}(W_X) + \frac{\lambda}{2}\|W_X\|_2^2\right) \\
&+ \frac{2}{\lambda n}\left(l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)\right) &(12)\\
\leq&\frac{2}{\lambda n}\left(l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)\right) &(13)\\
\leq&\frac{2\rho}{\lambda n}\|W_{X^{(-i)}} - W_X\|_2, &(14)
\end{aligned}$$

where Inequality (11) is based on Lemma 3; we change the measurement domain for the training error in Equation (12), such that one more term is added for each training error; since $W_{X^{(-i)}}$ is the minimiser of $L_{X^{(-i)}}(W_{X^{(-i)}}) + \frac{\lambda}{2}\|W_{X^{(-i)}}\|_2^2$, we have Inequality (13); Inequality (14), $\forall \mathbf{x}_i \in X$ is a result of ρ-Lipschitz loss function. Because $\|W_{X^{(-i)}} - W_X\|_2 \geq 0$, $i \sim U(|X|)$, we immediately have

$$\Delta_2 W = \|W_{X^{(-i)}} - W_X\|_2 \leq \frac{2\rho}{\lambda n}. \tag{15}$$

Furthermore, since the loss function $l(W_X, \mathbf{x})$ is ρ-Lipschitz, we have

$$l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i) \leq \rho\|W_{X^{(-i)}} - W_X\|_2 \leq \frac{2\rho^2}{\lambda n}. \tag{16}$$

Since this inequality holds for any X and $\mathbf{x}_i$ ($i \sim U(|X|)$), we then have

$$\mathbb{E}_{(X,\mathbf{x}_i)\sim\mathcal{D}^{n+1}, i\sim U(n)}[l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)] \leq \frac{2\rho^2}{\lambda n}. \tag{17}$$

Combining Inequality (15) and Inequality (17), we conclude this theorem. □

We notice that Chaudhuri et al. [8] provided similar results as Theorem 4. Their upper bounds of the L_2 global sensitivity of the model parameters are $2\rho/\lambda n$, specifically for a binary classification task on a normalised training dataset ($\|\mathbf{x}_i\| \leq 1$). In addition, Wu et al. [9] also provide the same upper bound of the global sensitivity of the model parameters as ours. However, their result needs the loss function to be β-smooth with a decreasing step size during the process of stochastic gradient descent. Once removing those conditions, their upper bounds become loose. Compared to [8] and [9], when achieving the same tight bound, our result does not rely on such assumptions/conditions; when following the same condition (convex loss function), our result is tighter.

Since $\Delta_2 W$ is the L_2 global sensitivity, we have $(\Delta_2 W)^2 = \sum_{i=1}^{|W_X|}(\Delta\omega_i)^2$, where $|W_X|$ is the number of edges in a deep neural network. Since each $\Delta\omega_i \leq \Delta\omega$, we have the global sensitivity (L_1-norm) of an individual model parameter as

$$\Delta\omega = \frac{\Delta_2 W}{\sqrt{|W_X|}} \leq \frac{2\rho}{\lambda n\sqrt{|W_X|}}. \tag{18}$$

Once having the L_1 global sensitivity of each single model parameter/weight, we could further calculate the upper bound of the global sensitivity of the value (input to the Softmax function) of the output layer.

Corollary 1. *Given an L_1 global sensitivity of an individual model parameter $\Delta\omega$, a fully connected $(T+1)$-layer neural network, $G = (V, E)$, where the output layer V_T has no activation function, the activation function at the hidden layers is bounded by a_u, then the L_1 global sensitivity of a neuron v_T at the output layer is bounded by $a_u|V_{T-1}|\Delta\omega$.*

Proof. Let z_T be the value of an arbitrary neuron at the output layer v_T, then we have $z_T = \sum_{j=1}^{|V_{T-1}|} a_{T-1,j}\omega_{T,j}$, where $|V_{T-1}|$ is the incoming degree of v_T in a fully connected neural network. Since the activation function is bounded by a_u. We achieve the maximum difference between z_T and $z_T^{(-i)}$ when $\omega_{T,j} > 0 > \omega_{T,j}^{(-i)}$ and $a_{T-1,j} = a_{T-1,j}^{(-i)} = a_u$, $\forall j \sim U(|V_{T-1}|)$, that is

$$\begin{aligned}
\Delta z_T &= \sum_{j=1}^{|V_{T-1}|} a_u\omega_{T,j} - \sum_{j=1}^{|V_{T-1}|} a_u\omega_{T,j}^{(-i)} \\
&\leq a_u\sum_{j=1}^{|V_{T-1}|}\left(\omega_{T,j} - \omega_{T,j}^{(-i)}\right) \\
&\leq a_u|V_{T-1}|\Delta\omega. &(19)
\end{aligned}$$

Hence this corollary holds. □

In practice, we have the upper bound of the activation function $a_u = 1$ for Tanh, Sigmoid, Binary step and Gaussian functions. In our experiments in Section 5, we use Tanh as the activation function in the hidden layers.

Corollary 2. *Given the global sensitivity of an individual neuron at the output layer of a neural network Δz_T, the upper bound of the global sensitivity of $p \in \mathbf{p}$, Δp, is $\min\{\exp(2\Delta z_T) - 1, 1\}$, where $\mathbf{p}$ is the prediction probability vector provided by the Softmax function.*

Proof. For an arbitrary neuron v at the output layer, we have $p_v = \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})}$, where C is the number of labels. Since $z_{T,j} - \Delta z_T \leq z_{T,j}^{(-i)} \leq z_{T,j} + \Delta z_T$, the global sensitivity of p is

$$\begin{aligned}
\Delta p =& \sup\left|p_v^{(-i)} - p_v\right| \\
=& \sup\left|\frac{\exp(z_{T,v}^{(-i)})}{\sum_{j=1}^{C}\exp(z_{T,j}^{(-i)})} - \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})}\right| \\
=& \frac{\exp(z_{T,v} + \Delta z_T)}{\exp(z_{T,v} + \Delta z_T) + \sum_{j\neq v}\exp(z_{T,j} - \Delta z_T)}
\end{aligned}$$

$$\begin{aligned}
&- \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})} \\
=&\frac{\exp(z_{T,v})}{\exp(z_{T,v}) + \sum_{j\neq v}\exp(z_{T,j} - 2\Delta z_T)} \\
&- \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})} \\
\leq&\frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j} - 2\Delta z_T)} - \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})} \\
=&(\exp(2\Delta z_T) - 1) \times \frac{\exp(z_{T,v})}{\sum_{j=1}^{C}\exp(z_{T,j})} \\
<&\exp(2\Delta z_T) - 1. \qquad (20)
\end{aligned}$$

Since both p_v and $p_v^{(-i)}$ are less than 1, we have $\Delta p = \min\{\exp(2\Delta z_T) - 1, 1\}$ to conclude this corollary. □

4.3 Lipschitzness Constant of Cross-entropy Loss Function

In this section, we show how to calculate the Lipschitzness constant ρ to calculate the upper bound of Δz, where other variables of Δz are fixed when configuring the training algorithm.

Lemma 4. *[22], [23] Given a fully connected neural network containing T layers, $f_T : X \to \mathbb{R}^C$, where the hidden layers apply 1-Lipschitz activation functions (e.g., ReLU, Tanh and Sigmoid), the output layer applies Softmax function, $\mathbf{X}_i$ is the values of neurons at Layer i, then the Lipschtiz constant (with respect to the model parameters) of f_T is bounded by $\prod_{i=1}^{T}\|\mathbf{X}_i\|_2$.*

Lemma 5. *[24] For a one layer neural network with cross-entropy loss function, the Lipschitz constant of the cross-entropy loss function (with respect to the model parameters) is $\frac{C-1}{C|V|}\|\mathbf{X}\|_2$, where C is the number of classes, $|V|$ is the number of the neurons at the input layer and $\mathbf{X}$ is a given data point.*

Based on Lemma 4 and Lemma 5, we have Proposition 1 to calculate the upper bound of the Lipschitz constant of the cross-entropy loss function (with respect to the model parameters) for a $(T + 1)$-layer neural network.

Proposition 1. *Using cross entropy function as the loss function of the aforementioned $(T + 1)$-layer neural network, the upper bound of Lipschitz constant (with respect to the model parameters) is*

$$\rho \leq \frac{(C-1)\prod_{t=0}^{T-1}\sqrt{|V_t|}x_{u,t}}{C|V_{T-1}|}, \qquad (21)$$

where C is the number of classes, $|V_t|$ is the number of neurons at layer t and $x_{u,t}$ is the upper bound of the absolute value of neurons at layer t.

4.4 Surrogate Loss Function

In our proofs in former sections, a key assumption is that the objective function of the learning algorithm is a combination of a convex loss function and a λ-strongly convex regulariser. Lemma 1 guarantees that the strong convexity for an L_2-norm regulariser. However, the loss functions of deep learning models are non-convex due to the large number of model parameters [25].

In this paper we follow existing results in multi-layer (more than one hidden layers) neural networks convexification [14], [15] by risk-averse optimisation to meet the convexity requirement to the loss function.

Theorem 5. *[14], [15] For a non-convex loss function $l(W_X, \mathbf{x})$ and its training error $L_X(W_X) = \frac{1}{n}\sum_{i=1}^{n} l(W_X, \mathbf{x}_i)$, their risk-averting error functions, $l^{(\alpha)}(W_X, \mathbf{x})$ and $L_X^{(\alpha)}(W_X)$, are convex, where*

$$\begin{aligned}
l^{(\alpha)}(W_X, \mathbf{x}) &= \exp(\alpha \times l(W_X, \mathbf{x})), \\
L_X^{(\alpha)}(W_X) &= \frac{1}{\alpha}\ln\left[\frac{1}{n}\sum_{i=1}^{n} l^{(\alpha)}(W_X, \mathbf{x}_i)\right]. \qquad (22)
\end{aligned}$$

α is the risk-factor which measures the size of convex region. Larger α indicates larger convex region.

So we have

$$\begin{aligned}
&L_X^{(\alpha)}(W_X) + 2\lambda\|W_X\|_2^2 \\
=&\frac{1}{\alpha}\ln\left[\frac{1}{n}\sum_{i=1}^{n}\exp(\alpha \times l(W_X, \mathbf{x}_i))\right] + 2\lambda\|W_X\|_2^2, \qquad (23)
\end{aligned}$$

where $l(W_X, \mathbf{x})$ could be those conventional/commonly used non-convex loss functions, such as quadratic loss and cross-entropy loss. We use the convexified surrogate loss function rather than the traditional non-convex loss function in the experimental evaluations.

4.5 Effect of Differential Privacy

In this section, we study the effect of differential privacy, including the DP guarantee of Algorithm 1 and the success of MI adversaries attacking DP deep learning models.

Theorem 6. *The prediction probability vector $\mathbf{p}$ of a given observation $\mathbf{x}$, produced by Algorithm 1 (injecting Laplace noise) is $(2C+1)\epsilon$-differentially private, where C is the number of labels.*

Proof. Let the activation function with the output layer be Softmax and the values of the neurons prior to feeding Softmax be $\mathbf{z}_T = (z_{T,1}, \ldots, z_{T,C})$, then we have the prediction probability vector $\mathbf{p} = (p_1, \ldots, p_C)$, where $p_i = \frac{\exp(z_{T,i})}{\sum_{i=1}^{C}\exp(z_{T,i})}$ and C is the number of labels.

Consider two sets of model parameters W_X and $W_{X^{(-i)}}$ trained from two neighbouring dataset X and $X^{(-i)}$, respectively. We run Algorithm 1 on W_X and $W_{X^{(-i)}}$ with the same privacy budget, the same data sample $\mathbf{x}$ for prediction and the same topology of the neural network. Based on Algorithm 1, we analyse the DP guarantee for the weighted sampling step and the noise injection step below.

Let the score function of the Exp-DP for sampling the neuron v be $q(X, v) = \exp(p_v)$. Based on Exp-DP and Lemma 2, the sampling weights of a neuron v is $\Pr[\text{sample}(X, q, \epsilon) = v] = \frac{\exp(\epsilon q(X,v)/2\Delta q)}{\sum_{i=1}^{C}\exp(\epsilon q(X,i)/2\Delta q)}$, where $\Delta q = \exp(\Delta p)$. Following the standard proof of the Exp-DP [17], we have

$$\frac{\Pr[\text{sample}(X, q(X, v), \epsilon) = v]}{\Pr[\text{sample}(X^{(-i)}, q(X^{(-i)}, v), \epsilon) = v]} \leq \exp(\epsilon). \qquad (24)$$

For a neuron v where noise has been injected and an arbitrary $r_v \in (0, 1)$, we have the worst-case upper bound of the probability ratio for $p_u \in \mathbf{p}$ ($\mathbf{p}$ is the noisy prediction probability vector) as follows.

$$
\begin{aligned}
&\frac{\Pr[\hat{p}_v = r_v | \text{sample}(X, q(X,v), \epsilon) = v]}{\Pr[\hat{p}_v^{(-i)} = r_v | \text{sample}(X^{(-i)}, q(X^{(-i)},v), \epsilon) = v]} \\
=& \frac{\Pr\left[\frac{\exp(\hat{z}_{T,v})}{\exp(\hat{z}_{T,v}) + \sum_{j\neq v}\exp(z_{T,j})} = r_v\right]}{\Pr\left[\frac{\exp(\hat{z}_{T,v}^{(-i)})}{\exp(\hat{z}_{T,v}^{(-i)}) + \sum_{j\neq v}\exp(z_{T,j}^{(-i)})} = r_v\right]} \\
=& \frac{\Pr\left[\hat{z}_{T,v} = \ln\left(\frac{r_v}{1-r_v}\sum_{j\neq v}\exp(z_{T,j})\right)\right]}{\Pr\left[\hat{z}_{T,v}^{(-i)} = \ln\left(\frac{r_v}{1-r_v}\sum_{j\neq v}\exp(z_{T,j}^{(-i)})\right)\right]} \\
=& \frac{\exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{r_v}{1-r_v}\sum_{j\neq v}\exp(z_{T,j})\right) - z_{T,v}\right|\right)}{\exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{r_v}{1-r_v}\sum_{j\neq v}\exp(z_{T,j}^{(-i)})\right) - z_{T,v}^{(-i)}\right|\right)} \\
\leq& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{\sum_{j\neq v}\exp(z_{T,j}))}{\sum_{j\neq v}\exp(z_{T,j}^{(-i)}))}\right) - z_{T,v} + z_{T,v}^{(-i)}\right|\right) \\
\leq& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{\sum_{j\neq v}\exp(z_{T,j}^{(-i)} + \Delta z_T))}{\sum_{j\neq v}\exp(z_{T,j}^{(-i)}))}\right) + \Delta z_T\right|\right) \\
=& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\exp(\Delta z_T)\frac{\sum_{j\neq v}\exp(z_{T,j}^{(-i)}))}{\sum_{j\neq v}\exp(z_{T,j}^{(-i)}))}\right) + \Delta z_T\right|\right) \\
=& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\Delta z_T + \Delta z_T\right|\right) \\
=& \exp(2\epsilon). \qquad (25)
\end{aligned}
$$

For an arbitrary neuron u where noise has not been injected (we inject noise into a neuron v) and an arbitrary $r_u \in (0,1)$, we have the worst-case upper bound of the probability ratio for $\hat{p}_u \in \mathbf{p}$ ($\mathbf{p}$ is the noisy prediction probability vector) under the condition of $\hat{p}_v = r_v$ and $\hat{p}_v^{(-i)} = r_v$.

$$
\begin{aligned}
&\frac{\Pr[\hat{p}_u = r_u | \hat{p}_v = r_v]}{\Pr[\hat{p}_u^{(-i)} = r_u | \hat{p}_v^{(-i)} = r_v]} \\
=& \frac{\Pr\left[\frac{\exp(z_{T,u})}{\sum_{j=1}^C \exp(z_{T,j})} = r_u \middle| \frac{\exp(\hat{z}_{T,v})}{\sum_{j=1}^C \exp(z_{T,j})} = r_v\right]}{\Pr\left[\frac{\exp(z_{T,u}^{(-i)})}{\sum_{j=1}^C \exp(z_{T,j}^{(-i)})} = r_u \middle| \frac{\exp(\hat{z}_{T,v}^{(-i)})}{\sum_{j=1}^C \exp(z_{T,j}^{(-i)})} = r_v\right]} \\
=& \frac{\Pr\left[r_v \cdot \frac{\exp(z_{T,u})}{\exp(\hat{z}_{T,v})} = r_u\right]}{\Pr\left[r_v \cdot \frac{\exp(z_{T,u}^{(-i)})}{\exp(\hat{z}_{T,v}^{(-i)})} = r_u\right]} \\
=& \frac{\Pr\left[\hat{z}_{T,v} = \ln\left(\frac{r_v \exp(z_{T,u})}{r_u}\right)\right]}{\Pr\left[\hat{z}_{T,v}^{(-i)} = \ln\left(\frac{r_v \exp(z_{T,u}^{(-i)})}{r_u}\right)\right]} \\
=& \frac{\exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{r_v \exp(z_{T,u})}{r_u}\right) - z_{T,v}\right|\right)}{\exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{r_v \exp(z_{T,u}^{(-i)})}{r_u}\right) - z_{T,v}^{(-i)}\right|\right)} \\
\leq& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{\exp(z_{T,u})}{\exp(z_{T,u}^{(-i)})}\right) - z_{T,v} + z_{T,v}^{(-i)}\right|\right) \\
\leq& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\frac{\exp(z_{T,u}^{(-i)} + \Delta z_T)}{\exp(z_{T,u}^{(-i)})}\right) + \Delta z_T\right|\right) \\
=& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\ln\left(\exp(\Delta z_T)\frac{\exp(z_{T,u}^{(-i)}))}{\exp(z_{T,u}^{(-i)}))}\right) + \Delta z_T\right|\right) \\
=& \exp\left(\frac{\epsilon}{\Delta z_T}\left|\Delta z_T + \Delta z_T\right|\right) \\
=& \exp(2\epsilon). \qquad (26)
\end{aligned}
$$

Then we give the upper bound of the privacy guarantee for the final prediction result.

$$
\begin{aligned}
&\frac{\Pr[\hat{\mathbf{p}} = \mathbf{r}]}{\Pr[\hat{\mathbf{p}}^{(-i)} = \mathbf{r}]} \\
=& \frac{\Pr[\text{sample}(X, \mathbf{p}, \epsilon) = v]}{\Pr[\text{sample}(X^{(-i)}, \mathbf{p}^{(-i)}, \epsilon) = v]} \times \prod_{j=1}^{C} \frac{\Pr[\hat{p}_j = r_j]}{\Pr[\hat{p}_j^{(-i)} = r_j]} \\
\leq& \exp(\epsilon) \times \frac{\Pr[\hat{p}_v = r_v]}{\Pr[\hat{p}_v^{(-i)} = r_v]} \times \prod_{j\neq v} \frac{\Pr[\hat{p}_j = r_j | \hat{p}_v = r_v]}{\Pr[\hat{p}_j^{(-i)} = r_j | \hat{p}_v^{(-i)} = r_v]} \\
\leq& \exp(\epsilon) \times \prod_{j=1}^{C} \exp(2\epsilon) \\
=& \exp((2C+1)\epsilon). \qquad (27)
\end{aligned}
$$

Therefore, we conclude this theorem. □

Corollary 3. *The differentially private prediction probability vector increases the difficulty of an MI adversary inferring membership with $X^{(-i)} = X \setminus \{x_i\}$, $i \sim U(|X|)$.*

Proof. Since injecting DP into the values of the neurons at the output layer (prior to the Softmax activation function). We take such a noise injection as the same effect of modifying the model parameters to be $\hat{W}_X$. Note that $\hat{W}_X$ is not a minimiser of $L_X(W_X) + \frac{\lambda}{2}\|W_X\|_2^2$, replacing W_X in Inequality (11) by $\hat{W}_X$ decreases the value of Inequality (11) and Equation (12). Following the same proof as Theorem 4, we have a tight upper bound for $\|W_{X^{(-i)}} - \hat{W}_X\|_2$, say, with a $c > 0$, $\|W_{X^{(-i)}} - \hat{W}_X\| = \|W_{X^{(-i)}} - W_X\| - c \leq 2\rho/\lambda n - c$. Since we assume the loss function is ρ-Lipschitz, we have

$$
\begin{aligned}
l(W_{X^{(-i)}}, \mathbf{x}_i) - l(\hat{W}_X, \mathbf{x}_i) &\leq \rho\|W_{X^{(-i)}} - \hat{W}_X\|_2 \\
&= \rho\|W_{X^{(-i)}} - W_X\|_2 - c\rho \\
&\leq \frac{2\rho^2}{\lambda n} - c\rho. \qquad (28)
\end{aligned}
$$

Since this is valid for any X and $\mathbf{x}_i$, $i \sim U(|X|)$, $|X| = n$, we immediately have

$$
\mathbb{E}_{(X,\mathbf{x}_i)\sim\mathcal{D}^{n+1}, i\sim U(n)}[l(W_{X^{(-i)}}, \mathbf{x}_i) - l(\hat{W}_X, \mathbf{x}_i)] \leq \frac{2\rho^2}{\lambda n} - c\rho. \qquad (29)
$$

Since $c\rho > 0$ and the upper bound of $\mathbb{E}_{(X,\mathbf{x}_i)\sim\mathcal{D}^{n+1}, i\sim U(n)}[l(W_{X^{(-i)}}, \mathbf{x}_i) - l(W_X, \mathbf{x}_i)]$ is $2\rho^2/\lambda n$ by Theorem 4, injecting differential privacy into the model parameters makes the learning model more On-Average-Remove-One stable, which mitigates the overfitting of a model, then further makes MI adversaries more difficult to infer membership privacy [26]. Hence this corollary holds. □

5 Experimental Evaluation

In this section, we show experimental evaluations of the proposed algorithm, including the dataset statistics, performance metrics, experimental configurations and the evaluation results and analyses.

5.1 Datasets, Metrics and Configurations

5.1.1 *Datasets*

In the experiments, we use the same datasets as Shokri et al. [2] and Jayaraman et al. [1], i.e., US Adult (Income)[1], MNIST[2], Locations (Bangkok restaurants check-ins)[3], Purchases[4], CIFAR[5] and Texas Hospital[6]. Since these datasets are commonly used in the field of machine learning and MI attacks, we only show the statistics of them in Table 3, where #Rec is the number of records (randomly sampled from the raw datasets) in training sets. For the details of these datasets, please refer to Section IV-a of [2] and Section 4.1 of [1].

TABLE 3: Datasets Statistics.

Dataset	#Rec.	#Feat.	#Labels	#Shadow Models
US Adult	10,000	14	2	20
MNIST	10,000	784	10	50
Locations	1,200	446	30	30
Purchases-2	10,000	600	2	20
Purchases-10	10,000	600	10	20
Purchases-20	10,000	600	20	20
Purchases-50	10,000	600	50	20
Purchases-100	10,000	600	100	20
CIFAR-10	10,000	3,072	10	100
CIFAR-100	10,000	3,072	100	100
Texas Hospital	10,000	6,169	100	10

5.1.2 *Performance Metrics*

Metrics for overfitting of the baseline non-private models. In our experiments, we use the upper bound of the On-Average-Remove-One (OARO) Stability (Definition 3 and Equation (17) in Theorem 4) to measure the overfitting of baseline non-private models, that is,

$$\mathbb{E}_{(X,\mathbf{x}_i)\sim\mathcal{D}^{n+1},i\sim U(n)}[l(W_{X^{(-i)}},\mathbf{x}_i)-l(W_X,\mathbf{x}_i)]\le\frac{2\rho^2}{\lambda n}, \tag{30}$$

where ρ is the Lipschitz constant calculated by Proposition 1, λ is the strongly convex constant shown in Lemma 1 and n is the size of the training set.

Metrics for DP models. Following existing studies [1], [26] on measuring DP performance against MI attacks, we use the same metrics in this paper, i.e., accuracy loss, a DP model's accuracy loss on the test set with respect to the baseline non-private model and privacy leakage, the difference between the true positive rate and the false negative rate of the MI attacks (as binary classifiers).

$$\text{Acc\_Loss}=1-\frac{\text{Test\_Acc}_{\text{DP}}}{\text{Test\_Acc}_{\text{Baseline (S)}}}, \tag{31}$$

1. http://archive.ics.uci.edu/ml/datasets/Adult
2. http://yann.lecun.com/exdb/mnist/
3. https://sites.google.com/site/yangdingqi/home/foursquare-dataset
4. https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data
5. https://www.cs.toronto.edu/~kriz/cifar.html
6. https://www.dshs.texas.gov/THCIC/Hospitals/Download.shtm

$$\text{Priv\_Leak}=\frac{|\text{TP}_{\text{MI}}|}{|\text{P}|}-\frac{|\text{FP}_{\text{MI}}|}{|\text{N}|}, \tag{32}$$

where Baseline (S) indicates the baseline non-private model trained on the surrogate loss function, TP_{MI} and FP_{MI} are the true positive and false positive of the MI attacks (specifying as the black-box shadow model approach from Shokri et al. [2] in this paper), P and N are positive/member and negative/non-member labels of a test data sample.

5.1.3 *Configurations*

Since existing upper bounds of the global sensitivity of the trained model parameters are looser than ours under the same condition, in our experiments, we compare the performance achieved by Algorithm 1 and Google's open-source implementation [18] of DP-SGD approaches on the real-world datasets.

Hyper-parameters. In general, we mainly follow the configurations in existing studies on MI attacks (the shadow model approach from Shokri et al.) [2] and DP-SGD [1] but train baseline non-private models, MI shadow models and DP models with a convexified surrogate loss function. Specifically, for training baseline non-private models and MI shadow models [2], we keep the same configurations as Shokri et al. [2]; to implement the DP-SGD (injecting Gaussian noise) approaches, we call Google's open-source tensorflow-privacy package following the same configurations as Jayaraman et al. [1]. We also follow the same computation as Jayaraman et al. [1] to plot the theoretical upper bound on the privacy leakage of ϵ-DP, which is $\exp(\epsilon)-1$. When plotting it, we bound it by the privacy leakage of the baseline non-private model. We tune the value of convexification constant (α in Eq. (23)) to ensure the surrogate loss function achieves similar training outcomes as the original loss function. Table 4 shows the detailed configurations, where *RDP*, *zCDP*, *AC* and *NC* represent Rényi DP [27], zero-Concentrated DP [28], DP with advanced composition [19] and DP with naïve composition [29], respectively, $|X|$ is the size of the training dataset.

Privacy budget. When implementing the noise injection of Algorithm 1, we apply both Laplace distribution and Gaussian distribution for generating DP noise. Note that existing DP-SGD approaches implement DP with Gaussian noise. In particular, when applying Laplace distribution, according to the sequential composition of privacy budget (Theorem 1), we split the overall privacy budget $\epsilon_{\text{overall}}$ as Theorem 6, that is, $\epsilon_{\text{neuron}}(=\epsilon_{\text{sampling}})=\epsilon_{\text{overall}}/2C+1$. When applying Gaussian distribution, according to the relationship between ϵ-GDP and (ϵ,δ)-DP (Theorem 3) and the composition theorem proposed in Gaussian differential privacy (Theorem 3), we split the privacy budget as $\epsilon_{\text{neuron}}(=\epsilon_{\text{sampling}})=\epsilon_{\text{overall}}/\sqrt{2^2\times C+1}$. The overall privacy budget values in this paper are the same as Jayaraman et al. [1], i.e., $\epsilon_{\text{overall}}=\{0.01,0.05,0.1,0.5,1,5,10,50,100,500,1000\}$.

Experimental setup. To evaluate the DP models (Algorithm 1 and DP-SGD models) on a given dataset with a privacy budget ϵ, we first randomly sample two disjoint sets from the dataset, one for the training set, the other for the test set. In the experiments, the test set size is always the same as the training set size for all the models. Using the

TABLE 4: Configurations in the Experiments.

Non-Private/DP-SGD/MI Shadow Models						
#hidden neurons	L_2 reg.	learning rate	#hidden layer	optimiser	batch size	#epoch
128	*0.001*	*0.001*	*1*	*ADMA*	*100*	*100*
activation func.	loss func.	ϵ	δ	DP Implementation		
Tanh, Softmax	*cross-entropy*	*[0.01, 1000]*	$1/10 \times \lvert X \rvert$	*RDP, zCDP, AC, NC*		
Convexification Constant (α in Eq. (23))						
$\alpha = 1$: Locations, MNIST, US Adults, Purchases-2, CIFAR-100					$\alpha = 2$: CIFAR-10	
$\alpha = 4$: Texas Hospital		$\alpha = 5$: Purchases-100, Purchases-50, Purchases-20, Purchases-10				
MI Attack Models						
#hidden neurons	L_2 reg.	learning rate	Other Configurations			
64	*0.01*	*0.01*	*The Same as MI Shadow Models*			

TABLE 5: Performance of the Baseline Non-Private Models
(sorted in decreasing order of Acc. Gap/Priv. Leak. of overfitting and fitting models).

Dataset	Original Loss Func.			Surrogate Loss Func.			Priv. Leak.	OARO Stability
	Training Acc.	Test Acc.	Acc. Gap	Training Acc.	Test Acc.	Acc. Gap		
Locations	1.0000	0.6068	0.3932	0.9916	0.6484	0.3432	0.3600	10.8532
Texas Hospital	0.7990	0.5689	0.2301	0.8770	0.5364	0.3406	0.2295	18.8945
Purchases-100	0.9992	0.7985	0.2007	0.9942	0.7723	0.2219	0.1664	1.8193
Purchases-50	0.9994	0.8636	0.1358	0.9985	0.8315	0.1670	0.1236	1.7827
Purchases-20	0.9986	0.9022	0.0964	0.9796	0.8759	0.1037	0.0727	1.6750
Purchases-10	0.9973	0.9203	0.0770	0.9735	0.8903	0.0832	0.0292	1.5036
Purchases-2	0.9963	0.9642	0.0321	0.9951	0.9642	0.0309	0.0073	0.4641
MNIST	0.9863	0.9528	0.0335	0.9494	0.9297	0.0197	0.0035	1.9845
US Adult	0.8310	0.8300	0.0010	0.8260	0.8262	0.0002	0.0023	0.0109
CIFAR-10	0.6198	0.4453	0.1745	0.5731	0.4236	0.1495	0.0111	7.7760
CIFAR-100	0.3224	0.1677	0.1647	0.2026	0.1389	0.0637	0.0099	9.4090

training set, we train the baseline non-private model and the ϵ-DP-SGD models and implement Algorithm 1 with ϵ. We calculate the accuracy loss of each DP model based on the test accuracy obtained from the test set. Then we perform the shadow model approach from Shokri et al. to attack the baseline non-private model and the DP models to calculate the privacy leakage. Finally, we repeat the training and attacking process 10 times to report the average accuracy loss and privacy leakage.

5.2 Experimental Results and Analysis

In this section, we present experimental results of comparing the performance (accuracy loss and privacy leakage) of Algorithm 1 with that of exiting DP-SGD approaches. We first report the performance (training and test accuracy) of the baseline non-private model using the surrogate function. Then we analyse the experimental results (accuracy loss and privacy leakage) of the DP models (Algorithm 1 and existing DP-SGD) obtained on the real-world datasets.

5.2.1 Performance of the Surrogate Loss Function

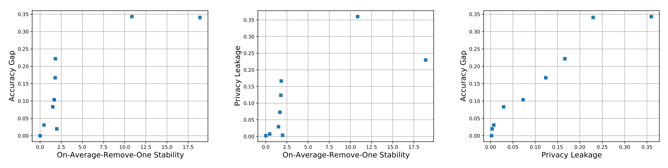
We compare the performance of models trained on the original non-convex loss function and on the convexified surrogate loss function with the same hyper-parameters and the same training and test datasets. Table 5 shows the average training accuracy and the average test accuracy achieved on the real-world datasets of 10 experiments, where α is the convexification constant in Equation (23).

As shown in Table 5, the surrogate loss function provides approximately the same model performance (training accuracy and test accuracy) as the original non-convex loss function. Such a result is also confirmed by the experimental analysis in [15]. Therefore, we can use the surrogate loss function to provide reliable baseline non-private models to further analyse the performance of the DP models.

5.2.2 Performance of the On-Average-Remove-One Stability on Measuring Overfitting

In this section, we examine the performance of the OARO stability on measuring overfitting via checking the relationship between the OARO stability and two empirical rules of detecting overfitting [26], i.e., accuracy gap between the training accuracy and the test accuracy and privacy leakage under the MI attacks.

Table 5 and Figure 2 show that the OARO stability is significantly correlated to both the accuracy gap and the privacy leakage, when the training accuracy of a baseline non-private model is acceptable. Specifically, in Figure 2a, the Pearson Correlation Coefficient is 0.8277 (p-value = 0.0059). In Figure 2b, the Pearson Correlation Coefficient is 0.7412 (p-value = 0.0223). In both Figures 2a and 2b, the Spearman's rank Correlation Coefficient is 0.7333 (p-value = 0.0246), the Kendall's rank Correlation Coefficient is 0.6667 (p-value = 0.0127). Therefore, we conclude that **the OARO stability can estimate the overfitting of a model when the model has a high training accuracy,** since we calculate the OARO stability during the training process. This could be a potential benefit when there are not enough data to prepare disjoint training and test datasets.



(a) OARO Stability vs. Acc. Gap (b) OARO Stability vs. Priv. Leak. (c) Priv. Leak. vs. Acc. Gap

Fig. 2: OARO Stability vs. Empirical Rules of Overfitting.

5.2.3 *Performance of the Differentially Private Models*

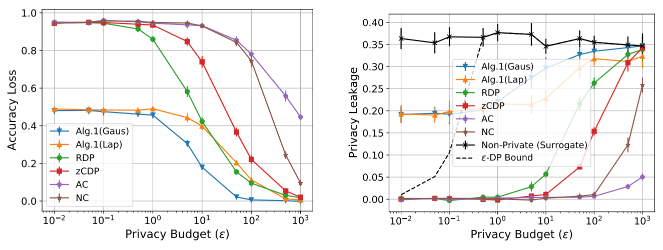
Figures 3 to 13 depict the accuracy loss and privacy leakage on the six real-world datasets, where *Alg. 1 (Gaus)* and *Alg. 1 (Lap)* represent Algorithm 1 implemented with Gaussian noise and Laplace noise, respectively. We give our key findings below.

Finding 1: Avoiding overfitting is still the rule of thumb to mitigate the effect of MI attacks (the shadow model approach). Based on the observed accuracy gap and privacy leakage of the non-private models on different datasets (Figure 2c, where the Pearson correlation coefficient is 0.9592, Spearman's rank and Kendall's rank correlation coefficients are 1, the p-value of all the three correlation coefficient is 0), we have the same conclusion as existing works [26], [30], [31]. That is, when a model is not overfitting, the privacy leakage of the non-private model would be rather marginal (almost zero privacy leakage) against MI attacks.

Finding 2: Algorithm 1 achieves a satisfying privacy-utility trade-off when the privacy leakage of the baseline model does not approximately equal to 0. Specifically, in most datasets (except the two most fitted models on MNIST and US Adults datasets, Figures 10 and 11, where the privacy leakage is less than 0.02), for a given privacy budget, Algorithm 1 (Gaussian noise and Laplace noise) provides the least accuracy loss and achieves the closest privacy leakage to the theoretical bound of ϵ-DP. Furthermore, when tuning the privacy budget to large values, say over 10 (Gaussian noise) or 100 (Laplace noise), Algorithm 1 could converge to the same privacy leakage as the non-private models as expected in DP theory in most of the datasets.

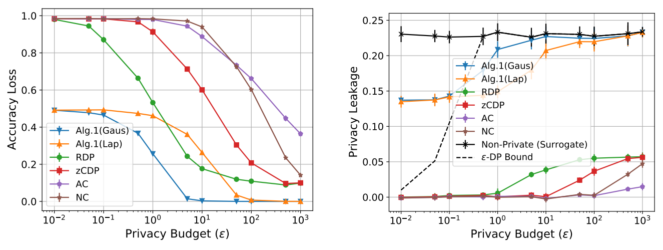
Finding 3: The accuracy loss of Algorithm 1 follows a stable monotonic decreasing curve when tuning the privacy budget. This property gives us a predictable accuracy loss when configuring a specific value of the privacy budget to generate DP prediction. Such a stability is in two-fold. First, the maximum accuracy loss of Algorithm 1 is about 0.5, which is much smaller than exiting DP-SGD approaches. Second, the accuracy loss of Algorithm 1 (Gaussian noise) significantly decreases from its maximum value to 0 when $\epsilon \in [0.1, 10]$, which is the commonly used range for tuning the privacy budget in practice (for Laplace noise, $\epsilon \in [1, 100]$). We observe exceptions on Location (Figure 3) and Purchases-2 (Figure 9) datasets, where the privacy budget are in $[1, 100]$ and $[0.01, 1]$, respectively. Whereas, we cannot observe such a stable curve in exiting DP-SGD approaches in any aspect, such as maximum accuracy loss (varying from 1.0 to 0.1 on different datasets), range of privacy budget for decreasing the accuracy loss (varying from $[0.05, 100]$ to $[0.01, 1]$ on different datasets).

Finding 4: Algorithm 1 (Laplace noise) achieves similar accuracy loss curve as Algorithm 1 (Gaussian noise) for the binary labels datasets. In the two binary labels datasets, Purchases-2 (Figure 9) and US Adults (Figure 11), the performance of injecting Gaussian noise and injecting Laplace noise is close to each other. Whereas, on the multi-label datasets, injecting Gaussian noise always outperforms Laplace noise injection on the privacy-utility trade-off.



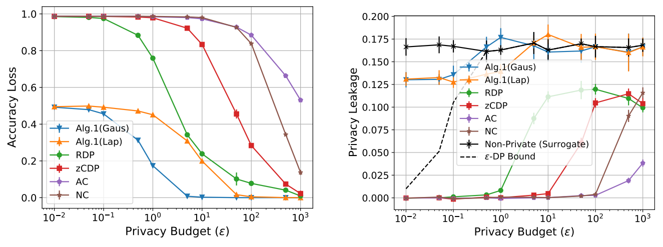
(a) Accuracy Loss (b) Privacy Leakage

Fig. 3: Performance Evaluations on Location Dataset.



(a) Accuracy Loss (b) Privacy Leakage

Fig. 4: Performance Evaluations on Texas Hospital Dataset.



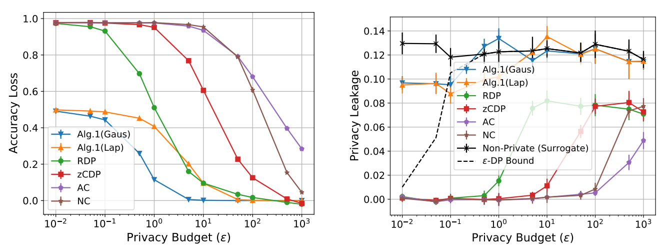
(a) Accuracy Loss (b) Privacy Leakage

Fig. 5: Performance Evaluations on Purchase-100 Dataset.

5.2.4 *Analysis of Experimental Results*

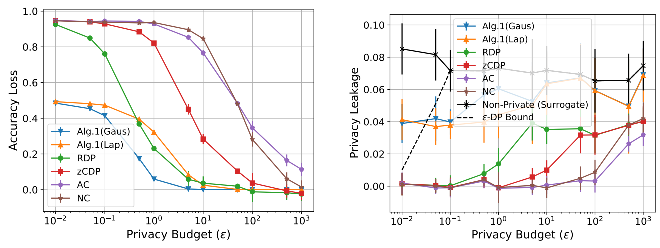
For the observations in Section 5.2.1 and Section 5.2.2, they reflect the expectation of the theoretical properties of the surrogate loss function shown in [14], [15] and Definition 3 for the OARO stability.

Findings 1, 2, 3 and 4 in Section 5.2.3 are supported by the noise injection scheme of Algorithm 1, i.e., applying the Exp-DP to sample an individual neuron for noise injection, and the tight upper bound of the global sensitivity. First,



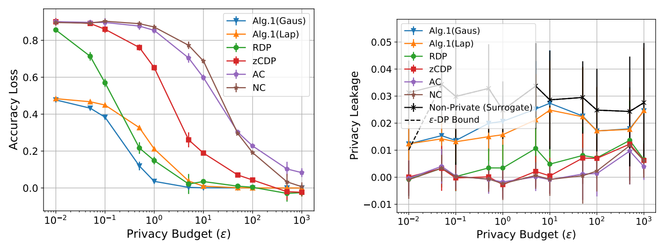
(a) Accuracy Loss (b) Privacy Leakage

Fig. 6: Performance Evaluations on Purchase-50 Dataset.



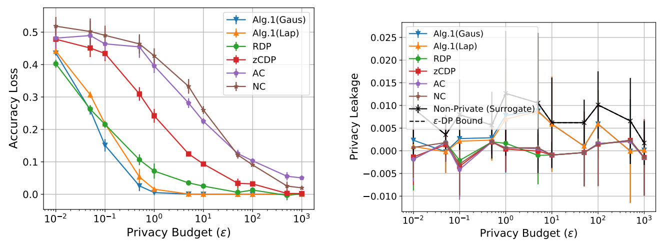
(a) Accuracy Loss (b) Privacy Leakage

Fig. 7: Performance Evaluations on Purchase-20 Dataset.



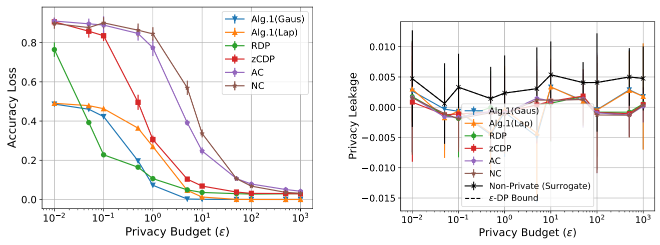
(a) Accuracy Loss (b) Privacy Leakage

Fig. 8: Performance Evaluations on Purchase-10 Dataset.



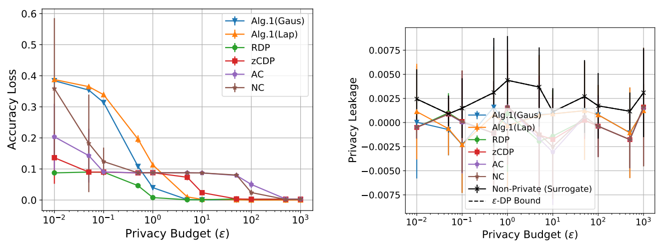
(a) Accuracy Loss (b) Privacy Leakage

Fig. 9: Performance Evaluations on Purchase-2 Dataset.



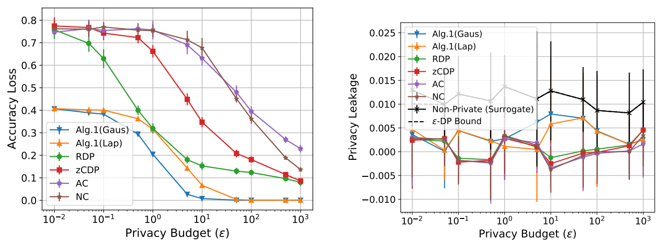
(a) Accuracy Loss (b) Privacy Leakage

Fig. 10: Performance Evaluations on MNIST Dataset.



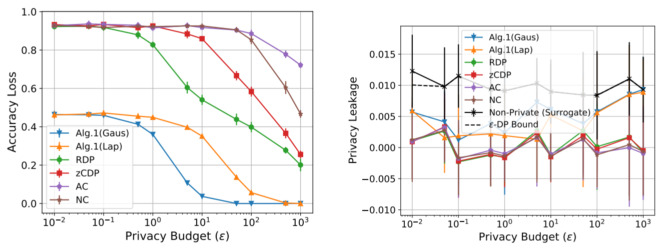
(a) Accuracy Loss (b) Privacy Leakage

Fig. 11: Performance Evaluations on US Adult Dataset.



(a) Accuracy Loss (b) Privacy Leakage

Fig. 12: Performance Evaluations on CIFAR-10 Dataset.



(a) Accuracy Loss (b) Privacy Leakage

Fig. 13: Performance Evaluations on CIFAR-100 Dataset.

when ϵ is small, we may inject very small negative noise, then the perturbation of the final prediction vector produced by Softmax (a marginal change in the denominator of Softmax) would not as significant as the noise amount. In this case, existing MIA implementations (not powerful enough) would simply ignore such marginal changes, then return high privacy leakage. This is the reason why Algorithm 1 has higher privacy leakage (against MI attacks using shadow models [2]) than the theoretical bound when using small ϵ. Second, based on the measurement for model's accuracy, injecting large positive noise into the neuron representing the correct prediction label (or injecting small negative noise into the neurons representing the incorrect prediction labels) would not impact the test accuracy of Algorithm 1. That is, Algorithm 1 gives the same prediction as the baseline model when using small privacy budgets in half times. Hence, Algorithm 1 has about 0.5 accuracy loss with small ϵ. Consider we are also using a tight upper bound of the global sensitivity, the accuracy loss of Algorithm 1 is much better than existing DP-SGD approaches. Then it further ensures an improved privacy-utility trade-off. In addition, in the binary labels datasets, since there are only two neurons at the output layer, with high probability, the Exp-DP would sample the neuron representing the accurate prediction label. In this case, the amount of noise would not impact the final prediction outcome. Hence, we could not observe significant difference between Gaussian noise and Laplace noise in binary-label datasets.

6 Conclusion

In this paper we provide a scheme for making differentially private prediction probability vector for general deep learning tasks. Our approach only injects DP noise into one neuron (sampled with Exponential mechanism of differential privacy) at the output layer of a given neural network. To implement our approach, we mathematically analyse the upper bound of L_1 global sensitivity of an individual neuron via the upper bound of L_2 global sensitivity of the model parameters. Our empirical studies show that, compared to existing DP stochastic gradient descent approaches, our approach achieves an improved trade-off between utility and privacy than existing DP-SGD approaches on six commonly used real-world datasets.

Acknowledgements

We thank Bargav Jayaraman for clarifying their codes of implementing DP-SGD. This work was conducted with funding received from the Optus Macquarie University Cyber Security Hub, in partnership with the Defence Science & Technology Group and Data61-CSIRO, through the Next Generation Technologies Fund. The experiments of this work was partially supported by the Australasian Leadership Computing Grants scheme, with computational resources provided by NCI Australia, an NCRIS enabled capability supported by the Australian Government.

References

[1] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1895–1912.

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[3] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[4] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography*. Springer, 2006, pp. 265–284.

[6] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, 2012.

[7] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[8] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization." *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[9] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1307–1322.

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

[11] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

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

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 609–616.

[14] J. T.-H. Lo, Y. Gui, and Y. Peng, "Overcoming the local-minimum problem in training multilayer perceptrons with the nrae training method," in *International Symposium on Neural Networks*. Springer, 2012, pp. 440–447.

[15] K. Dvijotham, M. Fazel, and E. Todorov, "Universal convexification via risk-aversion," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2014, pp. 162––171.

[16] F. Harder, M. Bauer, and M. Park, "Interpretable and differentially private predictions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4083–4090.

[17] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. 48th Annual IEEE Symposium on*. IEEE, 2007, pp. 94–103.

[18] G. Andrew, S. Chien, N. Papernot, and Anonymous_Contributors, "Tensorflow privacy," https://github.com/tensorflow/privacy, 2020.

[19] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[20] J. Dong, A. Roth, and W. J. Su, "Gaussian differential privacy," *arXiv preprint arXiv:1905.02383*, 2019.

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[22] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018, pp. 3835–3844.

[23] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, pp. 1–24, 2020.

[24] R. Yedida, "Finding a good learning rate," https://beginningwithml.wordpress.com/2019/01/07/2-2-finding-a-good-learning-rate/, 2019.

[25] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Advances in neural information processing systems*, 2006, pp. 123–130.

[26] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[27] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[28] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[29] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 486–503.

[30] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, "Effects of differential privacy and data skewness on membership inference vulnerability," *arXiv preprint arXiv:1911.09777*, 2019.

[31] S. M. Tonni, F. Farokhi, D. Vatsalan, D. Kaafar, Z. Lu, and G. Tangari, "Data and model dependencies of membership inference attack," *arXiv preprint arXiv:2002.06856*, 2020.