arXiv:1903.09822v5 [cs.CR] 15 Sep 2020

Scalable Differential Privacy with Certified Robustness in Adversarial Learning

NhatHai Phan [1] **My T. Thai** [2] **Han Hu** [1] **Ruoming Jin** [3] **Tong Sun** [4] **Dejing Dou** [5 6]

Abstract

In this paper, we aim to develop a scalable algorithm to preserve differential privacy (**DP**) in adversarial learning for deep neural networks (**DNNs**), with certified robustness to adversarial examples. By leveraging the sequential composition theory in DP, we randomize both input and latent spaces to strengthen our certified robustness bounds. To address the trade-off among model utility, privacy loss, and robustness, we design an original adversarial objective function, based on the post-processing property in DP, to tighten the sensitivity of our model. A new *stochastic batch training* is proposed to apply our mechanism on large DNNs and datasets, by bypassing the vanilla iterative batch-by-batch training in DP DNNs. An end-to-end theoretical analysis and evaluations show that our mechanism notably improves the robustness and scalability of DP DNNs.

1. Introduction

The pervasiveness of machine learning exposes new vulnerabilities in software systems, in which deployed machine learning models can be used (a) to reveal sensitive information in private training data (Fredrikson et al., 2015), and/or (b) to make the models misclassify, such as *adversarial examples* (Carlini & Wagner, 2017). Efforts to prevent such attacks typically seek one of three solutions: **(1)** Models which preserve differential privacy (**DP**) (Dwork et al., 2006), a rigorous formulation of privacy in probabilistic terms; **(2)** Adversarial training algorithms, which augment training data to consist of benign examples and adversarial examples crafted during the training process, thereby empirically increasing the classification accuracy given adversarial examples (Kardan & Stanley, 2017; Matyasko & Chau, 2017); and **(3)** Certified robustness, in which the model classification given adversarial examples is theoretically guaranteed to be consistent, i.e., a small perturbation in the input does not change the predicted label (Cisse et al., 2017; Kolter & Wong, 2017; Salman et al., 2019).

On the one hand, *private models*, trained with existing privacy-preserving mechanisms (Abadi et al., 2016; Shokri & Shmatikov, 2015; Phan et al., 2016; 2017a;b; Yu et al., 2019; Lee & Kifer, 2018), are unshielded under adversarial examples. On the other hand, *robust models*, trained with adversarial learning (with or without certified robustness to adversarial examples), do not offer privacy protections to the training data (Song et al., 2019). That one-sided approach poses serious risks to machine learning-based systems; since adversaries can attack a deployed model by using both privacy inference attacks and adversarial examples. To be safe, a model must be *i) private to protect the training data*, **and** *ii) robust to adversarial examples*. Unfortunately, there still lacks of study on how to develop such a model, which thus remains a largely open challenge (Phan et al., 2019).

Simply combining existing DP-preserving mechanisms and certified robustness conditions (Cisse et al., 2017; Kolter & Wong, 2017; Raghunathan et al., 2018) cannot solve the problem, for many reasons. **(a)** Existing sensitivity bounds (Phan et al., 2016; 2017a;b) and designs (Yu et al., 2019; Lee & Kifer, 2018; Phan et al., 2019; Wu et al., 2019; Xu et al., 2020) have not been developed to protect the training data in adversarial training. It is obvious that using adversarial examples crafted from the private training data to train our models introduces a previously unknown privacy risk, disclosing the participation of the benign examples (Song et al., 2019). **(b)** There is an unrevealed interplay among DP preservation, adversarial learning, and robustness bounds. **(c)** Existing algorithms cannot be readily applied to address the trade-off among model utility, privacy loss, and robustness. **(d)** It is challenging in applying existing algorithms to train large DNNs given large data (i.e., scalability); since, they employ the vanilla *iterative batch-by-batch training*, in which only a single batch of data instances can be used at each training step, such that the *privacy loss* can be es-

[1]Ying Wu College of Computing, New Jersey Institute of Technology, Newark, New Jersey, USA [2]Department of Computer & Information Sciences & Engineering, University of Florida, Gainesville, Florida, USA [3]Computer Science Department, Kent State University, Kent, Ohio, USA [4]Adobe Research, San Jose, California, USA [5]Computer and Information Science Department, University of Oregon, Eugene, Oregon, USA [6](Sabbatical leave from University of Oregon to) Baidu Research, Beijing, China. Correspondence to: NhatHai Phan <phan@njit.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

timated (Lee & Kifer, 2018; Phan et al., 2019; Yu et al., 2019; Wu et al., 2019; Xu et al., 2020). That prevents us from applying scalable methods, e.g., *distributed adversarial training* (Goyal et al., 2017), to achieve the same level of DP on large DNNs and datasets. Therefore, bounding the robustness of a model (which both protects the privacy and is robust against adversarial examples) *at scale* is nontrivial.

Contributions. Motivated by this open problem, we develop a novel *stochastic batch (**StoBatch**) mechanism* to: **1)** preserve DP of the training data, **2)** be provably and practically robust to adversarial examples, **3)** retain high model utility, and **4)** be scalable to large DNNs and datasets.

• In StoBatch, privacy-preserving noise is injected into inputs and hidden layers to achieve DP in learning private model parameters (**Theorem 1**). Then, we incorporate ensemble adversarial learning into our mechanism to improve the decision boundary under DP protections, by introducing a concept of *DP adversarial examples* crafted using benign examples in the private training data (**Eq. 8**). To address the trade-off between model utility and privacy loss, we propose a new DP adversarial objective function to tighten the model's global sensitivity (**Theorem 3**); thus, we reduce the amount of noise injected into our function, compared with existing works (Phan et al., 2016; 2017a;b). An end-to-end privacy analysis shows that, by slitting the private training data into *disjoint* and *fixed* batches across epochs, the privacy budget in our StoBatch is not accumulated across *gradient descent*-based training steps (**Theorems 3, 4**).

• After preserving DP in learning model parameters, we establish a new connection between DP preservation in adversarial learning and certified robustness. Noise injected into different layers is considered as a sequence of randomizing mechanisms, providing different levels of robustness. By leveraging the *sequential composition theory* in DP (Dwork & Roth, 2014), we derive a generalized robustness bound, which is a composition of these levels of robustness in both input and latent spaces (**Theorem 5** and **Corollary 1**), compared with only in the input space (Salman et al., 2019) or only in the latent space (Lecuyer et al., 2018).

• To bypass the iterative batch-by-batch training, we develop a *stochastic batch training*. In our algorithm, disjoint and fixed batches are distributed to local trainers, each of which learns DP parameters given its local data batches. A synchronous scheme can be leveraged to aggregate gradients observed from local trainers; thus enabling us to efficiently compute adversarial examples from multiple data batches at each iteration. This allows us to scale our mechanism to large DNNs and datasets, under the same DP guarantee. Rigorous experiments conducted on MNIST, CIFAR-10 (Lecun et al., 1998; Krizhevsky & Hinton, 2009), and (TinyImageNet) datasets show that our mechanism notably enhances the robustness and scalability of DP DNNs.

2. Background

In this section, we revisit DP, adversarial learning, and certified robustness. Let D be a database that contains N tuples, each of which contains data $x \in [-1,1]^d$ and a *ground-truth label* $y \in \mathbb{Z}_K$ (one-hot vector), with K possible categorical outcomes $y = \{y_1, \ldots, y_K\}$. A single *true class label* $y_x \in y$ given $x \in D$ is assigned to only one of the K categories. On input x and parameters θ, a model outputs class scores $f : \mathbb{R}^d \rightarrow \mathbb{R}^K$ that maps x to a vector of scores $f(x) = \{f_1(x), \ldots, f_K(x)\}$ s.t. $\forall k \in [1,K] : f_k(x) \in [0,1]$ and $\sum_{k=1}^{K} f_k(x) = 1$. The class with the highest score value is selected as the *predicted label* for x, denoted as $y(x) = \max_{k \in K} f_k(x)$. A loss function $L(f(x), y)$ presents the penalty for mismatching between the predicted values $f(x)$ and original values y. The notations and terminologies used in this paper are summarized in Table 1 (**Appendix A**). Let us briefly revisit DP DNNs, starting with the definition of DP.

Definition 1 *(ϵ, δ)-DP (Dwork et al., 2006). A randomized algorithm A fulfills (ϵ, δ)-DP, if for any two databases D and D' differing at most one tuple, and for all $O \subseteq Range(A)$, we have:*

$$Pr[A(D) = O] \leq e^{\epsilon} Pr[A(D') = O] + \delta \qquad (1)$$

ϵ controls the amount by which the distributions induced by D and D' may differ, δ is a broken probability.

DP also applies to general metrics $\rho(D, D') \leq 1$, where ρ can be l_p-norms (Chatzikokolakis et al., 2013). DP-preserving algorithms in DNNs can be categorized into three lines: 1) introducing noise into *parameter gradients* (Abadi et al., 2016; 2017; Shokri & Shmatikov, 2015; Yu et al., 2019; Lee & Kifer, 2018; Phan et al., 2019), 2) injecting noise into objective functions (Phan et al., 2016; 2017a;b), and 3) injecting noise into labels (Papernot et al., 2018).

Adversarial Learning. For some target model f and inputs (x, y_x), the adversary's goal is to find an *adversarial example* $x^{\text{adv}} = x + \alpha$, where α is the perturbation introduced by the attacker, such that: **(1)** x^{adv} and x are close, and **(2)** the model misclassifies x^{adv}, i.e., $y(x^{\text{adv}}) \neq y(x)$. In this paper, *we consider well-known $l_{p \in \{1,2,\infty\}}(\mu)$-norm bounded attacks (Goodfellow et al., 2014), where μ is the radius of the p-norm ball*. To improve the robustness of models, prior work focused on two directions: 1) Producing correct predictions on adversarial examples, while not compromising the accuracy on legitimate inputs (Kardan & Stanley, 2017; Matyasko & Chau, 2017; Wang et al., 2016; Papernot et al., 2016b;a; Gu & Rigazio, 2014; Papernot & McDaniel, 2017; Hosseini et al., 2017); and 2) Detecting adversarial examples (Metzen et al., 2017; Grosse et al., 2017; Xu et al., 2017; Abbasi & Gagné, 2017; Gao et al., 2017). Among existing solutions, adversarial training appears to hold the

greatest promise for learning robust models (Tramèr et al., 2017). A well-known algorithm was proposed in (Kurakin et al., 2016b). At every training step, new adversarial examples are generated and injected into batches containing both benign and adversarial examples (Alg. 2, **Appendix C**).

Certified Robustness and DP. Recently, some algorithms (Cisse et al., 2017; Kolter & Wong, 2017; Raghunathan et al., 2018; Cohen et al., 2019; Li et al., 2018; Salman et al., 2019) have been proposed to derive certified robustness, in which each prediction is guaranteed to be consistent under the perturbation α, if a robustness condition is held. Given a benign example x, we focus on achieving a robustness condition to $l_p(\mu)$-norm attacks, as follows:

$$\forall \alpha \in l_p(\mu) : f_k(x+\alpha) > \max_{i:i\neq k} f_i(x+\alpha) \quad (2)$$

where $k = y(x)$, indicating that a small perturbation α in the input does not change the predicted label $y(x)$. To achieve the robustness condition in Eq. 2, (Lecuyer et al., 2018) introduce an algorithm, called **PixelDP**. By considering an input x (e.g., images) as databases in DP parlance, and individual features (e.g., pixels) as tuples, PixelDP shows that randomizing the scoring function $f(x)$ to enforce DP on a small number of pixels in an image guarantees robustness of predictions. To randomize $f(x)$, *random noise* σ_r is injected into either input x or an arbitrary hidden layer, resulting in the following (ϵ_r, δ_r)-PixelDP condition:

Lemma 1 *(ϵ_r, δ_r)-PixelDP (Lecuyer et al., 2018). Given a randomized scoring function $f(x)$ satisfying (ϵ_r, δ_r)-PixelDP w.r.t. a l_p-norm metric, we have:*

$$\forall k, \forall \alpha \in l_p(1) : \mathbb{E} f_k(x) \leq e^{\epsilon_r} \mathbb{E} f_k(x+\alpha) + \delta_r \quad (3)$$

where $\mathbb{E} f_k(x)$ is the expected value of $f_k(x)$, ϵ_r is a predefined budget, δ_r is a broken probability.

At the prediction time, a certified robustness check is implemented for each prediction, as follows:

$$\hat{\mathbb{E}}_{lb} f_k(x) > e^{2\epsilon_r} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(x) + (1+e^{\epsilon_r})\delta_r \quad (4)$$

where $\hat{\mathbb{E}}_{lb}$ and $\hat{\mathbb{E}}_{ub}$ are the lower and upper bounds of the expected value $\hat{\mathbb{E}} f(x) = \frac{1}{n}\sum_n f(x)_n$, derived from the Monte Carlo estimation with an η-confidence, given n is the number of invocations of $f(x)$ with independent draws in the noise σ_r. Passing the check for a given input guarantees that no perturbation up to $l_p(1)$-norm can change the model's prediction. PixelDP *does not* preserve DP in learning private parameters θ to protect the training data.

3. Stochastic Batch (StoBatch) Mechanism

StoBatch is presented in Alg. 4 (**Appendix D**). Our DNN (Fig. 1a) is presented as: $f(x) = g(a(x,\theta_1),\theta_2)$, where $a(x,\theta_1)$ is a feature representation learning model with x as an input, and g will take the output of $a(x,\theta_1)$ and return the class scores $f(x)$. At a high level, there are four key components: **(1)** DP $a(x,\theta_1)$, which is to preserve DP in learning the feature representation model $a(x,\theta_1)$; **(2)** DP Adversarial Learning, which focuses on preserving DP in adversarial learning, given DP $a(x,\theta_1)$; **(3)** Certified Robustness and Verified Inferring, which are to compute robustness bounds given an input at the inference time; and **(4)** Stochastic batch training (Fig. 1b). To establish theoretical results in DP preservation and in deriving robustness bounds, let us first present our mechanism in the vanilla iterative batch-by-batch training (**Alg. 1**). The network f (Lines 2-3, Alg. 1) is trained over T training steps. In each step, a disjoint and fixed batch of m perturbed training examples and a disjoint and fixed batch of m DP adversarial examples, derived from D, are used to train our network (Lines 4-12, Alg. 1).

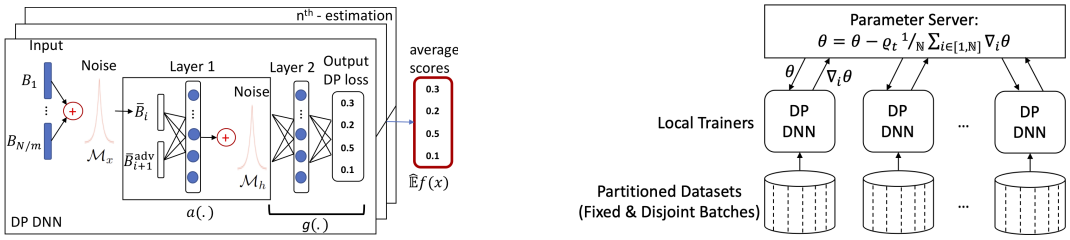
3.1. DP Feature Representation Learning

Our idea is to use auto-encoder to simultaneously learn DP parameters θ_1 and ensure that the output of $a(x,\theta_1)$ is DP, since: (1) It is easier to train, given its small size; and (2) It can be reused for different predictive models. A typical data reconstruction function (cross-entropy), given a batch B_t at the training step t of the input x_i, is as follows: $\mathcal{R}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\big[x_{ij}\log(1+e^{-\theta_{1j}h_i}) + (1-x_{ij})\log(1+e^{\theta_{1j}h_i})\big]$, where $h_i = \theta_1^T x_i$, the hidden layer $\mathbf{h}_1$ of $a(x,\theta_1)$ given the batch B_t is denoted as $\mathbf{h}_{1B_t} = \{\theta_1^T x_i\}_{x_i\in B_t}$, and $\widetilde{x}_i = \theta_1 h_i$ is the reconstruction of x_i.

To preserve ϵ_1-DP in learning θ_1 where ϵ_1 is a privacy budget, we first derive the 1st-order polynomial approximation of $\mathcal{R}_{B_t}(\theta_1)$ by applying Taylor Expansion (Arfken, 1985), denoted as $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$. Then, *Functional Mechanism* (Zhang et al., 2012) **(revisited in Appendix B)** is adapted to inject noise into coefficients of the approximated function $\widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\sum_{l=1}^{2}\sum_{r=0}^{1}\frac{\mathbf{F}_{lj}^{(r)}(0)}{r!}\big(\theta_{1j}h_i\big)^r$, where $\mathbf{F}_{1j}(z) = x_{ij}\log(1+e^{-z})$, $\mathbf{F}_{2j}(z) = (1-x_{ij})\log(1+e^{z})$, we have that: $\widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\Big[\log 2 + \theta_{1j}\big(\frac{1}{2}-x_{ij}\big)h_i\Big]$. In $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$, parameters θ_{1j} derived from the function optimization need to be ϵ_1-DP. To achieve that, Laplace noise $\frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ is injected into coefficients $\big(\frac{1}{2}-x_{ij}\big)h_i$, where $\Delta_{\mathcal{R}}$ is the sensitivity of $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$, as follows:

$$\widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\Big[\theta_{1j}\Big(\big(\frac{1}{2}-x_{ij}\big)h_i + \frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})\Big)\Big]$$
$$= \sum_{x_i\in B_t}\Big[\sum_{j=1}^{d}(\frac{1}{2}\theta_{1j}\overline{h}_i) - x_i\widetilde{x}_i\Big] \quad (5)$$

To ensure that the computation of $\widetilde{x}_i$ does not access the



(a) An instance of DP DNNs and verified inference

(b) An instance of stochastic batch training

Figure 1. Stochastic Batch mechanism.

original data, we further inject Laplace noise $\frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ into x_i. This can be done as a preprocessing step for all the benign examples in D to construct a set of *disjoint* batches $\overline{\mathbf{B}}$ of perturbed benign examples (Lines 2 and 5, Alg. 1). The perturbed function now becomes:

$$\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1) = \sum_{\overline{x}_i \in \overline{B}_t} \Big[\sum_{j=1}^{d} (\frac{1}{2}\theta_{1j}\overline{h}_i) - \overline{x}_i \widetilde{x}_i \Big] \tag{6}$$

where $\overline{x}_i = x_i + \frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1}), h_i = \theta_1^T \overline{x}_i, \overline{h}_i = h_i + \frac{2}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$, and $\widetilde{x}_i = \theta_1 \overline{h}_i$. Let us denote β as the number of neurons in $\mathbf{h}_1$, and h_i is bounded in $[-1, 1]$, the global sensitivity $\Delta_{\mathcal{R}}$ is as follows:

Lemma 2 *The global sensitivity of $\widetilde{\mathcal{R}}$ over any two neighboring batches, B_t and B'_t, is: $\Delta_{\mathcal{R}} \leq d(\beta + 2)$.*

All the proofs are in **Appendix**. By setting $\Delta_{\mathcal{R}} = d(\beta + 2)$, we show that the output of $a(\cdot)$, which is the perturbed affine transformation $\overline{\mathbf{h}}_{1\overline{B}_t} = \{\theta_1^T \overline{x}_i + \frac{2}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})\}_{\overline{x}_i \in \overline{B}_t}$, is (ϵ_1/γ)-DP, given $\gamma = \frac{2\Delta_{\mathcal{R}}}{m\|\overline{\theta}_1\|_{1,1}}$ and $\|\overline{\theta}_1\|_{1,1}$ is the maximum 1-norm of θ_1's columns (Operator norm, 2018). This is important to tighten the privacy budget consumption in computing the remaining hidden layers $g(a(x, \theta_1), \theta_2)$. In fact, without using additional information from the original data, the computation of $g(a(x, \theta_1), \theta_2)$ is also (ϵ_1/γ)-DP.

Similarly, the perturbation of each benign example x turns $\overline{B}_t = \{\overline{x}_i \leftarrow x_i + \frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})\}_{x_i \in B_t}$ into a $(\epsilon_1/\gamma_{\mathbf{x}})$-DP batch, with $\gamma_{\mathbf{x}} = \Delta_{\mathcal{R}}/m$. We do not use the post-processing property of DP to estimate the DP guarantee of $\overline{\mathbf{h}}_{1\overline{B}_t}$ based upon the DP guarantee of $\overline{B}_t$, since $\epsilon_1/\gamma < \epsilon_1/\gamma_{\mathbf{x}}$ in practice. So, the (ϵ_1/γ)-DP $\overline{\mathbf{h}}_{1\overline{B}_t}$ provides a more rigorous DP protection to the computation of $g(\cdot)$ and to the output layer.

Lemma 3 *The computation of the batch $\overline{B}_t$ as the input layer is $(\epsilon_1/\gamma_{\mathbf{x}})$-DP, and the computation of the affine transformation $\overline{\mathbf{h}}_{1\overline{B}_t}$ is (ϵ_1/γ)-DP.*

Departing from the vanilla Functional Mechanism, in which only *grid search*-based approaches can be applied to find DP-preserving θ_1 with a low loss $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$, our following Theorem 1 shows that *gradient descent*-based optimizing $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}}+\epsilon_1)$-DP in learning θ_1 given an $(\epsilon_1/\gamma_{\mathbf{x}})$-DP $\overline{B}_t$ batch. In fact, in addition to h_i, $\overline{h}_i$, $\widetilde{x}_i$, based on Lemma 3, we further show that the computation of gradients, i.e., $\forall j \in [1, d] : \frac{\delta \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)}{\delta \theta_{1j}} = \sum_{i=1}^{m} \overline{h}_i(\frac{1}{2} - \overline{x}_{ij})$, and descent operations given the $(\epsilon_1/\gamma_{\mathbf{x}})$-DP $\overline{B}_t$ batch are $(\epsilon_1/\gamma_{\mathbf{x}})$-DP, without incurring any additional information from the original data. As a result, gradient descent-based approaches can be applied to optimize $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ in Alg. 1, since all the computations on top of $\overline{B}_t$ are DP, without using any additional information from the original data.

Theorem 1 *The gradient descent-based optimization of $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ preserves $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP in learning θ_1.*

3.2. Adversarial Learning with Differential Privacy

To integrate adversarial learning, we first draft DP adversarial examples $\overline{x}_j^{\text{adv}}$ using perturbed benign examples $\overline{x}_j$, with an ensemble of attack algorithms A and a random perturbation budget $\mu_t \in (0, 1]$, at each step t (Lines 6-11, Alg. 1). This will significantly enhances the robustness of our models under different types of adversarial examples with an unknown adversarial attack size μ.

$$\overline{x}_j^{\text{adv}} = \overline{x}_j + \mu \cdot \text{sign}\Big(\nabla_{\overline{x}_j} \mathcal{L}\big(f(\overline{x}_j, \theta), y(\overline{x}_j)\big)\Big) \tag{7}$$

with $y(\overline{x}_j)$ is the class prediction result of $f(\overline{x}_j)$ to avoid label leaking of x_j during the adversarial example crafting. Given a set of DP adversarial examples $\overline{B}_t^{\text{adv}}$, training the auto-encoder with $\overline{B}_t^{\text{adv}}$ preserves $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP.

Theorem 2 *The gradient descent-based optimization of $\overline{\mathcal{R}}_{\overline{B}_t^{adv}}(\theta_1)$ preserves $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP in learning θ_1.*

The proof of Theorem 2 is in **Appendix J, Result 4**. It can be extended to iterative attacks as: $\overline{x}_{j,0}^{\text{adv}} = \overline{x}_j$,

$$\overline{x}_{j,\mathtt{t}+1}^{\text{adv}} = \overline{x}_{j,\mathtt{t}}^{\text{adv}} + \frac{\mu}{T_\mu} \cdot \text{sign}\Big(\nabla_{\overline{x}_{j,\mathtt{t}}^{\text{adv}}} \mathcal{L}\big(f(\overline{x}_{j,\mathtt{t}}^{\text{adv}}, \theta), y(\overline{x}_{j,\mathtt{t}}^{\text{adv}})\big)\Big) \tag{8}$$

where $y(\overline{x}_{j,\mathtt{t}}^{\text{adv}})$ is the prediction of $f(\overline{x}_{j,\mathtt{t}}^{\text{adv}}, \theta)$, $\mathtt{t} \in [0, T_\mu - 1]$.

Second, we propose a novel DP adversarial objective function $L_{B_t}(\theta_2)$, in which the loss function $\mathcal{L}$ for benign examples is combined with an additional loss function Υ for DP

Algorithm 1 Adversarial Learning with DP

Input: Database D, loss function L, parameters θ, batch size m, learning rate ϱ_t, privacy budgets: ϵ_1 and ϵ_2, robustness parameters: ϵ_r, Δ_r^x, and Δ_r^h, adversarial attack size μ_a, the number of invocations n, ensemble attacks A, parameters ψ and ξ, and the size $|\mathbf{h}_\pi|$ of $\mathbf{h}_\pi$

1: **Draw Noise** $\chi_1 \leftarrow [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^d$, $\chi_2 \leftarrow [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta$, $\chi_3 \leftarrow [Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})]^{|\mathbf{h}_\pi|}$
2: **Randomly Initialize** $\theta = \{\theta_1, \theta_2\}$, $\mathbf{B} = \{B_1, \ldots, B_{N/m}\}$ s.t. $\forall B \in \mathbf{B} : B$ is a batch with the size m, $B_1 \cap \ldots \cap B_{N/m} = \emptyset$, and $B_1 \cup \ldots \cup B_{N/m} = D$, $\overline{\mathbf{B}} = \{\overline{B}_1, \ldots, \overline{B}_{N/m}\}$ where $\forall i \in [1, N/m] : \overline{B}_i = \{\overline{x} \leftarrow x + \frac{\chi_1}{m}\}_{x \in B_i}$
3: **Construct** a deep network f with **hidden layers** $\{\mathbf{h}_1 + \frac{2\chi_2}{m}, \ldots, \mathbf{h}_\pi\}$, where $\mathbf{h}_\pi$ is the last hidden layer
4: **for** $t \in [T]$ **do**
5: **Take** a batch $\overline{B}_i \in \overline{\mathbf{B}}$ where $i = t\%(N/m)$, $\overline{B}_t \leftarrow \overline{B}_i$
6: **Ensemble DP Adversarial Examples:**
7: **Draw Random Perturbation Value** $\mu_t \in (0, 1]$
8: **Take** a batch $\overline{B}_{i+1} \in \overline{\mathbf{B}}$, **Assign** $\overline{B}_t^{\text{adv}} \leftarrow \emptyset$
9: **for** $l \in A$ **do**
10: **Take** the next batch $\overline{B}_a \subset \overline{B}_{i+1}$ with the size $m/|A|$
11: $\forall \overline{x}_j \in \overline{B}_a$: **Craft** $\overline{x}_j^{\text{adv}}$ by using attack algorithm $A[l]$ with $l_\infty(\mu_t)$, $\overline{B}_t^{\text{adv}} \leftarrow \overline{B}_t^{\text{adv}} \cup \overline{x}_j^{\text{adv}}$
12: **Descent:** $\theta_1 \leftarrow \theta_1 - \varrho_t \nabla_{\theta_1} \overline{\mathcal{R}}_{\overline{B}_t \cup \overline{B}_t^{\text{adv}}}(\theta_1)$; $\theta_2 \leftarrow \theta_2 - \varrho_t \nabla_{\theta_2} \overline{L}_{\overline{B}_t \cup \overline{B}_t^{\text{adv}}}(\theta_2)$ with the noise $\frac{\chi_3}{m}$

Output: $\epsilon = (\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$-DP parameters $\theta = \{\theta_1, \theta_2\}$, robust model with an ϵ_r budget

adversarial examples, to optimize the parameters θ_2. The objective function $L_{B_t}(\theta_2)$ is defined as follows:

$$L_{\overline{B}_t \cup \overline{B}_t^{\text{adv}}}(\theta_2) = \frac{1}{m(1+\xi)} \Big(\sum_{\overline{x}_i \in \overline{B}_t} \mathcal{L}\big(f(\overline{x}_i, \theta_2), y_i\big) + \xi \sum_{\overline{x}_j^{\text{adv}} \in \overline{B}_t^{\text{adv}}} \Upsilon\big(f(\overline{x}_j^{\text{adv}}, \theta_2), y_j\big) \Big) \quad (9)$$

where ξ is a hyper-parameter. For the sake of clarity, in Eq. 9, we denote y_i and y_j as the true class labels y_{x_i} and y_{x_j} of examples x_i and x_j. $\overline{x}_j^{\text{adv}}$ and x_j share the same label y_{x_j}.

Now we are ready to preserve DP in objective functions $\mathcal{L}\big(f(\overline{x}_i, \theta_2), y_i\big)$ and $\Upsilon\big(f(\overline{x}_j^{\text{adv}}, \theta_2), y_j\big)$ in order to achieve DP in learning θ_2. Since the objective functions use the true class labels y_i and y_j, we need to protect the labels at the output layer. Let us first present our approach to preserve DP in the objective function $\mathcal{L}$ for benign examples. Given $\mathbf{h}_{\pi i}$ computed from the $\overline{x}_i$ through the network with W_π is the parameter at the last hidden layer $\mathbf{h}_\pi$. Cross-entropy function is approximated as: $\mathcal{L}_{\overline{B}_t}(\theta_2) \approx \sum_{k=1}^K \sum_{\overline{x}_i} \big[\mathbf{h}_{\pi i} W_{\pi k} - (\mathbf{h}_{\pi i} W_{\pi k}) y_{ik} - \frac{1}{2}|\mathbf{h}_{\pi i} W_{\pi k}| + \frac{1}{8}(\mathbf{h}_{\pi i} W_{\pi k})^2\big] \approx \mathcal{L}_{1\overline{B}_t}(\theta_2) - \mathcal{L}_{2\overline{B}_t}(\theta_2)$, where $\mathcal{L}_{1\overline{B}_t}(\theta_2) = \sum_{k=1}^K \sum_{\overline{x}_i} \big[\mathbf{h}_{\pi i} W_{\pi k} - \frac{1}{2}|\mathbf{h}_{\pi i} W_{\pi k}| + \frac{1}{8}(\mathbf{h}_{\pi i} W_{\pi k})^2\big]$, and $\mathcal{L}_{2\overline{B}_t}(\theta_2) = \sum_{k=1}^K \sum_{\overline{x}_i} (\mathbf{h}_{\pi i} y_{ik}) W_{\pi k}$.

Based on the *post-processing property of DP* (Dwork & Roth, 2014), $\mathbf{h}_{\pi \overline{B}_t} = \{\mathbf{h}_{\pi i}\}_{\overline{x}_i \in \overline{B}_t}$ is (ϵ_1/γ)-DP, since the computation of $\overline{\mathbf{h}}_{1\overline{B}_t}$ is (ϵ_1/γ)-DP (Lemma 3). Hence, the optimization of $\mathcal{L}_{1\overline{B}_t}(\theta_2)$ does not disclose any information from the training data, and $\frac{Pr(\mathcal{L}_{1\overline{B}_t}(\theta_2))}{Pr(\mathcal{L}_{1\overline{B}'_t}(\theta_2))} = \frac{Pr(\mathbf{h}_{\pi \overline{B}_t})}{Pr(\mathbf{h}_{\pi \overline{B}'_t})} \leq e^{\epsilon_1/\gamma}$, given neighboring batches $\overline{B}_t$ and $\overline{B}'_t$. Thus, we only need to preserve ϵ_2-DP in the function $\mathcal{L}_{2\overline{B}_t}(\theta_2)$, which accesses the ground-truth label y_{ik}. Given coefficients $\mathbf{h}_{\pi i} y_{ik}$, the sensitivity $\Delta_{\mathcal{L}2}$ of $\mathcal{L}_{2\overline{B}_t}(\theta_2)$ is computed as:

Lemma 4 *Let $\overline{B}_t$ and $\overline{B}'_t$ be neighboring batches of benign examples, we have the following inequality: $\Delta_{\mathcal{L}2} \leq 2|\mathbf{h}_\pi|$, where $|\mathbf{h}_\pi|$ is the number of hidden neurons in $\mathbf{h}_\pi$.*

The sensitivity of our objective function is notably smaller than the state-of-the-art bound (Phan et al., 2017b), which is crucial to improve our model utility. The perturbed functions become: $\overline{\mathcal{L}}_{\overline{B}_t}(\theta_2) = \mathcal{L}_{1\overline{B}_t}(\theta_2) - \overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)$, where $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2) = \sum_{k=1}^K \sum_{\overline{x}_i} \big(\mathbf{h}_{\pi i} y_{ik} + \frac{1}{m} Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})\big) W_{\pi k}$.

Theorem 3 *Algorithm 1 preserves $(\epsilon_1/\gamma + \epsilon_2)$-DP in the gradient descent-based optimization of $\overline{\mathcal{L}}_{\overline{B}}(\theta_2)$.*

We apply the same technique to preserve $(\epsilon_1/\gamma + \epsilon_2)$-DP in the optimization of the function $\Upsilon\big(f(\overline{x}_j^{\text{adv}}, \theta_2), y_j\big)$ over the DP adversarial examples $\overline{x}_j^{\text{adv}} \in \overline{B}_t^{\text{adv}}$. As the perturbed functions $\overline{\mathcal{L}}$ and $\overline{\Upsilon}$ are always optimized given two disjoint batches $\overline{B}_t$ and $\overline{B}_t^{\text{adv}}$, the privacy budget used to preserve DP in the adversarial objective function $L_{B_t}(\theta_2)$ is $(\epsilon_1/\gamma + \epsilon_2)$, following the *parallel composition* property (Dwork & Roth, 2014). The total budget to learn private parameters $\overline{\theta} = \{\overline{\theta}_1, \overline{\theta}_2\} = \arg\min_{\{\theta_1, \theta_2\}} (\overline{\mathcal{R}}_{\overline{B}_t \cup \overline{B}_t^{\text{adv}}}(\theta_1) + \overline{L}_{\overline{B}_t \cup \overline{B}_t^{\text{adv}}}(\theta_2))$ is $\epsilon = (\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$ (Line 12, Alg. 1).

DP at the Dataset Level. Our mechanism achieves DP at the batch level $\overline{B}_t \cup \overline{B}_t^{\text{adv}}$ given a specific training step t. By constructing *disjoint* and *fixed* batches from D, we leverage both parallel composition and post-processing properties of DP to extend the result to ϵ-DP in learning $\{\overline{\theta}_1, \overline{\theta}_2\}$ on D across T training steps. There are three key properties in our model: **(1)** It only reads perturbed inputs $\overline{B}_t$ and perturbed coefficients $\overline{\mathbf{h}}_1$, which are DP across T training steps with *a single draw of Laplace noise* (i.e., no further privacy leakage); **(2)** Given N/m disjoint batches in each epoch, $\forall \overline{x}$, $\overline{x}$ is included in *one and only one* batch, denoted $B_x \in \overline{\mathbf{B}}$. As a result, the DP guarantee to $\overline{x}$ in D is equivalent to the DP guarantee to $\overline{x}$ in B_x; since the optimization using any

other batches does not affect the DP guarantee of $\overline{x}$, even the objective function given B_x can be slightly different from the objective function given any other batches in $\overline{\mathbf{B}}$; and **(3)** All the batches are fixed across T training steps to prevent additional privacy leakage, caused by generating new and overlapping batches (which are considered overlapping datasets in the parlance of DP) in the typical training.

Theorem 4 *Algorithm 1 achieves* $(\epsilon_1+\epsilon_1/\gamma_{\mathbf{x}}+\epsilon_1/\gamma+\epsilon_2)$*-DP parameters* $\overline{\theta}=\{\overline{\theta}_1,\overline{\theta}_2\}$ *on the private training data* D *across* T *gradient descent-based training steps.*

3.3. Certified Robustness

Now, we establish the correlation between our mechanism and certified robustness. In the *inference time*, to derive the certified robustness condition against adversarial examples $x+\alpha$, i.e., $\forall\alpha\in l_p(1)$, PixelDP randomizes the function $f(x)$ by injecting *robustness noise* σ_r into either input x or a hidden layer, i.e., $x'=x+Lap(\frac{\Delta_r^x}{\epsilon_r})$ or $h'=h+Lap(\frac{\Delta_r^h}{\epsilon_r})$, where Δ_r^x and Δ_r^h are the sensitivities of x and h, measuring how much x and h can be changed given the perturbation $\alpha\in l_p(1)$ in the input x. Monte Carlo estimation of the expected values $\hat{\mathbb{E}}f(x)$, $\hat{\mathbb{E}}_{lb}f_k(x)$, and $\hat{\mathbb{E}}_{ub}f_k(x)$ are used to derive the robustness condition in Eq. 4.

On the other hand, in our mechanism, the privacy noise σ_p includes Laplace noise injected into both input x, i.e., $\frac{1}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$, and its affine transformation h, i.e., $\frac{2}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$. Note that the perturbation of $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)$ is equivalent to $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)=\sum_{k=1}^K\sum_{\overline{x}_i}(\mathbf{h}_{\pi i}y_{ik}W_{\pi k}+\frac{1}{m}Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})W_{\pi k})$. This helps us to avoid injecting the noise directly into the coefficients $\mathbf{h}_{\pi i}y_{ik}$. The correlation between our DP preservation and certified robustness lies in the correlation between the privacy noise σ_p and the robustness noise σ_r.

We can derive a robustness bound by projecting the privacy noise σ_p *on the scale of the robustness noise* σ_r. Given the input x, let $\kappa=\frac{\Delta_{\mathcal{R}}}{m\epsilon_1}/\frac{\Delta_r^x}{\epsilon_r}$, in our mechanism we have that: $\overline{x}=x+Lap(\kappa\Delta_r^x/\epsilon_r)$. By applying a group privacy size κ (Dwork & Roth, 2014; Lecuyer et al., 2018), the scoring function $f(x)$ satisfies ϵ_r-PixelDP given $\alpha\in l_p(\kappa)$, or equivalently is ϵ_r/κ-PixelDP given $\alpha\in l_p(1)$, $\delta_r=0$. By applying Lemma 1, we have

$$\forall k,\forall\alpha\in l_p(\kappa):\mathbb{E}f_k(x)\leq e^{\epsilon_r}\mathbb{E}f_k(x+\alpha),$$
$$or\quad\forall k,\forall\alpha\in l_p(1):\mathbb{E}f_k(x)\leq e^{\frac{\epsilon_r}{\kappa}}\mathbb{E}f_k(x+\alpha)$$

With that, we can achieve a robustness condition against $l_p(\kappa)$-norm attacks, as follows:

$$\hat{\mathbb{E}}_{lb}f_k(x)>e^{2\epsilon_r}\max_{i:i\neq k}\hat{\mathbb{E}}_{ub}f_i(x)\tag{10}$$

with the probability $\geq\eta_x$-confidence, derived from the Monte Carlo estimation of $\hat{\mathbb{E}}f(x)$. Our mechanism also perturbs h (Eq. 6). Given $\varphi=\frac{2\Delta_{\mathcal{R}}}{m\epsilon_1}/\frac{\Delta_r^h}{\epsilon_r}$, we further have $\overline{h}=h+Lap(\frac{\varphi\Delta_r^h}{\epsilon_r})$. Therefore, the scoring function $f(x)$ also satisfies ϵ_r-PixelDP given the perturbation $\alpha\in l_p(\varphi)$. In addition to the robustness to the $l_p(\kappa)$-norm attacks, we achieve an additional robustness bound in Eq. 10 against $l_p(\varphi)$-norm attacks. Similar to PixelDP, these robustness conditions can be achieved as randomization processes in the inference time. They can be considered as *two independent and certified defensive mechanisms* applied against two l_p-norm attacks, i.e., $l_p(\kappa)$ and $l_p(\varphi)$.

One challenging question here is: *"What is the general robustness bound, given* κ *and* φ*?"* Intuitively, our model is robust to attacks with $\alpha\in l_p(\frac{\kappa\varphi}{\kappa+\varphi})$. We leverage the theory of *sequential composition* in DP (Dwork & Roth, 2014) to theoretically answer this question. Given S independent mechanisms $\mathcal{M}_1,\ldots,\mathcal{M}_S$, whose privacy guarantees are $\epsilon_1,\ldots,\epsilon_S$-DP with $\alpha\in l_p(1)$. Each mechanism $\mathcal{M}_s$, which takes the input x and outputs the value of $f(x)$ with the Laplace noise only injected to randomize the layer s (i.e., no randomization at any other layers), denoted as $f^s(x)$, is defined as: $\forall s\in[1,S],\mathcal{M}_sf(x):\mathbb{R}^d\rightarrow f^s(x)\in\mathbb{R}^K$. We aim to derive a generalized robustness of any composition scoring function $f(\mathcal{M}_1,\ldots,\mathcal{M}_s|x):\prod_{s=1}^S\mathcal{M}_sf(x)$ bounded in $[0,1]$, defined as follows:

$$f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x):\mathbb{R}^d\rightarrow\prod_{s\in[1,S]}f^s(x)\in\mathbb{R}^K\tag{11}$$

Our setting follows the sequential composition in DP (Dwork & Roth, 2014). Thus, we can prove that the expected value $\mathbb{E}f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)$ is insensitive to small perturbations $\alpha\in l_p(1)$ in Lemma 5, and we derive our composition of robustness in Theorem 5, as follows:

Lemma 5 *Given* S *independent mechanisms* $\mathcal{M}_1,\ldots,\mathcal{M}_S$*, which are* $\epsilon_1,\ldots,\epsilon_S$*-DP w.r.t a* l_p*-norm metric, then the expected output value of any sequential function* f *of them, i.e.,* $f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)\in[0,1]$*, satisfies:*

$$\forall\alpha\in l_p(1):\mathbb{E}f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)\leq e^{(\sum_{s=1}^S\epsilon_s)}\mathbb{E}f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x+\alpha)$$

Theorem 5 *(Composition of Robustness) Given* S *independent mechanisms* $\mathcal{M}_1,\ldots,\mathcal{M}_S$*. Given any sequential function* $f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)$*, and let* $\hat{\mathbb{E}}_{lb}$ *and* $\hat{\mathbb{E}}_{ub}$ *are lower and upper bounds with an* η*-confidence, for the Monte Carlo estimation of* $\hat{\mathbb{E}}f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)=\frac{1}{n}\sum_n f(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)_n=\frac{1}{n}\sum_n(\prod_{s=1}^S f^s(x)_n)$.

$$\forall x,if\ \exists k\in K:\hat{\mathbb{E}}_{lb}f_k(\mathcal{M}_1,\ldots,\mathcal{M}_S|x)>e^{2(\sum_{s=1}^S\epsilon_s)}\max_{i:i\neq k}\hat{\mathbb{E}}_{ub}f_i(\mathcal{M}_1,\ldots,\mathcal{M}_S|x),\tag{12}$$

then the predicted label $k = \arg\max_k \hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)$, *is robust to adversarial examples* $x + \alpha$, $\forall \alpha \in l_p(1)$, *with probability* $\geq \eta$, *by satisfying:* $\hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha) > \max_{i:i\neq k} \hat{\mathbb{E}} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha)$, *which is the targeted robustness condition in Eq. 2.*

There is no η_s-confidence for each mechanism s, since we do not estimate the expected value $\hat{\mathbb{E}} f^s(x)$ independently. To apply the composition of robustness in our mechanism, the noise injections into the input x and its affine transformation h can be considered as two mechanisms $\mathcal{M}_x$ and $\mathcal{M}_h$, sequentially applied as $(\mathcal{M}_h(x), \mathcal{M}_x(x))$. When $\mathcal{M}_h(x)$ is applied by invoking $f(x)$ with independent draws in the noise χ_2, the noise χ_1 injected into x is fixed; and vice-versa. By applying group privacy (Dwork & Roth, 2014) with sizes κ and φ, the scoring functions $f^x(x)$ and $f^h(x)$, given $\mathcal{M}_x$ and $\mathcal{M}_h$, are ϵ_r/κ-DP and ϵ_r/φ-DP with $\alpha \in l_p(1)$. With Theorem 5, we have a generalized bound as follows:

Corollary 1 *(StoBatch Robustness).* $\forall x$, *if* $\exists k \in K$: $\hat{\mathbb{E}}_{lb} f_k(\mathcal{M}_h, \mathcal{M}_x|x) > e^{2\epsilon_r} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_h, \mathcal{M}_x|x)$ *(i.e., Eq. 12), then the predicted label* k *of our function* $f(\mathcal{M}_h, \mathcal{M}_x|x)$ *is robust to perturbations* $\alpha \in l_p(\frac{\kappa\varphi}{\kappa+\varphi})$ *with the probability* $\geq \eta$, *by satisfying*

$$\forall \alpha \in l_p(\frac{\kappa\varphi}{\kappa+\varphi}) : \hat{\mathbb{E}} f_k(\mathcal{M}_h, \mathcal{M}_x|x+\alpha) > \max_{i:i\neq k} \hat{\mathbb{E}} f_i(\mathcal{M}_h, \mathcal{M}_x|x+\alpha)$$

Compared with state-of-the-art robustness analysis (Salman et al., 2019; Lecuyer et al., 2018), in which either the input space or the latent space are randomized, the advantage of our robustness bound is the composition of different levels of robustness in both input and latent spaces.

3.4. Verified Inference

At the inference time, we implement a *verified inference* (Alg. 3, **Appendix D**) to return a *robustness size guarantee* for each example x, i.e., the maximal value of $\frac{\kappa\varphi}{\kappa+\varphi}$, for which the robustness condition in Corollary 1 holds. Maximizing $\frac{\kappa\varphi}{\kappa+\varphi}$ is equivalent to maximizing the robustness epsilon ϵ_r, which is the only parameter controlling the size of $\frac{\kappa\varphi}{\kappa+\varphi}$; since, all the other hyper-parameters, i.e., $\Delta_{\mathcal{R}}$, m, ϵ_1, ϵ_2, θ_1, θ_2, Δ_r^x, and Δ_r^h are fixed given a well-trained model $f(x)$:

$$(\frac{\kappa\varphi}{\kappa+\varphi})_{max} = \max_{\epsilon_r} \frac{\Delta_{\mathcal{R}}\epsilon_r}{m\epsilon_1(\Delta_r^x + \Delta_r^h/2)}$$
$$\text{s.t. } \hat{\mathbb{E}}_{lb} f_k(x) > e^{2\epsilon_r} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(x) \text{ (i.e., Eq. 12)} \quad (13)$$

The prediction on an example x is robust to attacks up to $(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$. The failure probability 1-η can be made arbitrarily small by increasing the number of invocations of $f(x)$, with independent draws in the noise. Similar to (Lecuyer et al., 2018), Hoeffding's inequality is applied to *bound* the approximation error in $\hat{\mathbb{E}} f_k(x)$ and to *search* for the robustness bound $(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$. We use the following sensitivity bounds $\Delta_r^h = \beta\|\theta_1\|_\infty$ where $\|\theta_1\|_\infty$ is the maximum 1-norm of θ_1's rows, and $\Delta_r^x = \mu d$ for l_∞ attacks. In the Monte Carlo Estimation of $\hat{\mathbb{E}} f(x)$, we also propose a new method to draw independent noise to control the *distribution shifts* between training and inferring, in order to improve the verified inference effectiveness, without affecting the DP protection and the robustness bounds (**Appendix N**).

3.5. Distributed Training

In the vanilla iterative batch-by-batch training for DP DNNs, at each step, only one batch of examples can be used to train our model, so that the privacy loss can be computed (Lee & Kifer, 2018; Yu et al., 2019; Wu et al., 2019; Xu et al., 2020). Parameters θ_1 and θ_2 are independently updated (Lines 4-12, Alg. 1). This prevents us from applying *practical adversarial training* (Xie et al., 2019; Goyal et al., 2017), in which *distributed training using synchronized SGD* on many GPUs (e.g., 128 GPUs) is used to scale adversarial training to large DNNs. Each GPU processes a mini-batch of 32 images (i.e., the total batch size is $128 \times 32 = 4,096$).

To overcome this, a well-applied technique (Yu et al., 2019) is to fine-tune a limited number of layers, such as a fully connected layer and the output layer, under DP of a pre-trained model, i.e., VGG16, trained over a public and large dataset, e.g., ImageNet, in order to handle simpler tasks on smaller private datasets, e.g., CIFAR-10. Although this approach works well, there are several utility and security concerns: **(1)** Suitable public data may not always be available, especially for highly sensitive data; **(2)** Trojans can be implanted in the pre-trained model for backdoor attacks (Liu et al., 2018); and **(3)** Public data can be poisoned (Shafahi et al., 2018). Fine-tuning a limited number of layers may not be secure; while fine-tuning an entire of a large pre-trained model iteratively batch-by-batch is still inefficient.

To address this bottleneck, we leverage the training recipe of (Xie et al., 2019; Goyal et al., 2017) to propose a distributed training algorithm, called **StoBatch** (Fig. 1b), in order to efficiently train large DP DNNs in adversarial learning, without affecting the DP protection (Alg. 4, **Appendix D**). In StoBatch, fixed and disjoint batches $\overline{\mathbf{B}}$ are distributed to $N/(2m)$ local trainers, each of which have two batches $\{\overline{B}_{i1}, \overline{B}_{i2}\}$ randomly picked from $\overline{\mathbf{B}}$ with $i \in [1, N/(2m)]$ (Line 4, Alg. 4). At each training step t, we randomly pick $\mathbb{N}$ local trainers, each of which gets the latest global parameters θ from the parameter server. A local trainer i will compute the gradients $\nabla_i\theta_1$ and $\nabla_i\theta_2$ to optimize the DP objective functions $\overline{\mathcal{R}}$ and $\overline{L}$ using its local batch $\overline{B}_{i1}$ and ensemble DP adversarial examples crafted from $\overline{B}_{i2}$ (Lines 5-14, Alg.

4). The gradients will be sent back to the parameter server for a synchronized SGD (Lines 15-16, Alg. 4), as follows: $\theta_1 \leftarrow \theta_1 - \frac{\varrho_t}{\mathbb{N}} \sum_{i\in[1,\mathbb{N}]} \nabla_i \theta_1, \quad \theta_2 \leftarrow \theta_2 - \frac{\varrho_t}{\mathbb{N}} \sum_{i\in[1,\mathbb{N}]} \nabla_i \theta_2$. This enables us to train large DNNs with our DP adversarial learning, by training from multiple batches simultaneously with more adversarial examples, without affecting the DP guarantee in Theorem 4; since the optimization of one batch does not affect the DP protection at any other batch and at the dataset level D across T training steps (**Theorem 4**).

In addition, the *average errors of our approximation functions* are always *bounded*, and are *independent* of the number of data instances N in D (**Appendix O**). This further ensures that our functions can be applied in large datasets.

Our approach can be extended into two different complementary scenarios: **(1)** Distributed training for each local trainer i, in which the batches $\{\overline{B}_{i1}, \overline{B}_{i2}\}$ can be located across $\mathbb{M}$ GPUs to efficiently compute the gradients $\nabla_i \theta_1 = \frac{1}{\mathbb{M}} \sum_{j\in[1,\mathbb{M}]} \nabla_{i,j} \theta_1$ and $\nabla_i \theta_2 = \frac{1}{\mathbb{M}} \sum_{j\in[1,\mathbb{M}]} \nabla_{i,j} \theta_2$; and **(2)** Federated training, given each local trainer can be considered as an independent party. In this setting, an independent party can further have different sizes of batches. As long as the global sensitivities $\Delta_{\mathcal{R}}$ and $\Delta_{\mathcal{L}2}$ are the same for all the parties, the DP guarantee in Theorem 4 does hold given D be the union of all local datasets from all the parties. This can be achieved by nomalizing all the inputs x to be in $[-1, 1]^d$. This is a step forward compared with the classical federated learning (McMahan et al., 2016). We focus on the distributed training setting in this paper, and reserve the federated learning scenarios for future exploration.

4. Experimental Results

We have conducted an extensive experiment on the MNIST, CIFAR-10, and Tiny ImageNet datasets. Our validation focuses on shedding light into the interplay among model utility, privacy loss, and robustness bounds, by learning **1)** the impact of the privacy budget $\epsilon = (\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$, **2)** the impact of attack sizes μ_a, and **3)** the scalability of our mechanism. We consider the class of l_∞-bounded adversaries. *All statistical tests are 2-tail t-tests.* Please refer to the **Appendix Q** for a complete analysis of our experimental results, including **Figures 2 - 9**. The implementation of our mechanism is available in TensorFlow[1].

Baseline Approaches. Our **StoBatch** mechanism is evaluated in comparison with state-of-the-art mechanisms in: (1) DP-preserving algorithms in deep learning, i.e., **DP-SGD** (Abadi et al., 2016), **AdLM** (Phan et al., 2017b); in (2) Certified robustness, i.e., **PixelDP** (Lecuyer et al., 2018); and in (3) DP-preserving algorithms with certified robustness, i.e., **SecureSGD** given heterogeneous noise (Phan et al., 2019), and **SecureSGD-AGM** (Phan et al., 2019) given the Analytic Gaussian Mechanism (AGM) (Balle & Wang, 2018). To preserve DP, DP-SGD injects random noise into gradients of parameters, while AdLM is a Functional Mechanism-based approach. PixelDP is one of the state-of-the-art mechanisms providing certified robustness using DP bounds. SecureSGD is a combination of PixelDP and DP-SGD with an advanced heterogeneous noise distribution; i.e., "more noise" is injected into "more vulnerable" latent features, to improve the robustness. The baseline models share the same design in our experiment. Four white-box attacks were used, including **FGSM**, **I-FGSM**, Momentum Iterative Method (**MIM**) (Dong et al., 2017), and **MadryEtAl** (Madry et al., 2018). Pure robust training and analysis can incur privacy leakage (Song et al., 2019); thus, in this study, similar algorithms to (Salman et al., 2019) do not fit as comparable baselines, since they may not be directly applicable to DP DNNs.

Model Configuration (Appendix P). It is important to note that $x \in [-1, 1]^d$ in our setting, which is different from a common setting, $x \in [0, 1]^d$. Thus, a given attack size $\mu_a = 0.3$ in the setting of $x \in [0, 1]^d$ is equivalent to an attack size $2\mu_a = 0.6$ in our setting. The reason for using $x \in [-1, 1]^d$ is to achieve better model utility, while retaining the same global sensitivities to preserve DP, compared with $x \in [0, 1]^d$. As in (Lecuyer et al., 2018), we apply two accuracy metrics:

$$\textit{conventional acc} = \sum_{i=1}^{|test|} \frac{isCorrect(x_i)}{|test|}$$

$$\textit{certified acc} = \sum_{i=1}^{|test|} \frac{isCorrect(x_i)\ \&\ isRobust(x_i)}{|test|}$$

where $|test|$ is the number of test cases, $isCorrect(\cdot)$ returns 1 if the model makes a correct prediction (else, returns 0), and $isRobust(\cdot)$ returns 1 if the robustness size is larger than a given attack size μ_a (else, returns 0).

Results on the MNIST Dataset. Figure 2 illustrates the conventional accuracy of each model as a function of the privacy budget ϵ on the MNIST dataset under $l_\infty(\mu_a)$-norm attacks, with $\mu_a = 0.2$. Our StoBatch outperforms AdLM, DP-SGD, SecureSGD, and SecureSGD-AGM, in all cases, with $p < 1.32e-4$. When the privacy budget $\epsilon = 0.2$ (a tight DP protection), there are significant drops, in terms of conventional accuracy, given the baseline approaches. By contrast, our StoBatch only shows a small degradation in the conventional accuracy. At $\epsilon = 0.2$, our StoBatch achieves 82.7%, compared with 11.2% and 41.64% correspondingly for SecureSGD-AGM and SecureSGD. This shows the ability to offer tight DP protections under adversarial example attacks in our model, compared with existing algorithms.

• Figure 4 presents the conventional accuracy as a function of μ_a, under a strong DP guarantee, $\epsilon = 0.2$. It is

[1] https://github.com/haiphanNJIT/StoBatch

clear that our StoBatch mechanism outperforms the baseline approaches in all cases. On average, our StoBatch model improves 44.91% over SecureSGD, a 61.13% over SecureSGD-AGM, a 52.21% over AdLM, and a 62.20% over DP-SGD. More importantly, thanks to the composition robustness bounds in both input and latent spaces, and the random perturbation size $\mu_t \in (0, 1]$, our StoBatch model is resistant to different attack algorithms with different attack sizes μ_a, compared with baseline approaches.

• Figure 6 demonstrates the certified accuracy as a function of μ_a. The privacy budget is set to 1.0, offering a reasonable privacy protection. In PixelDP, the construction attack bound ϵ_r is set to 0.1, which is a pretty reasonable defense. With (small perturbation) $\mu_a \leq 0.2$, PixelDP achieves better certified accuracies under all attacks; since PixelDP does not preserve DP to protect the training data, compared with other models. Meanwhile, our StoBatch model outperforms all the other models when $\mu_a \geq 0.3$, indicating a stronger defense to more aggressive attacks.

Results on the CIFAR-10 Dataset further strengthen our observations. In Figure 3, our StoBatch outperforms baseline models in all cases ($p < 6.17e-9$), especially with small privacy budget ($\epsilon < 4$), yielding strong DP protections. On average conventional accuracy, our StoBatch mechanism has an improvement of 10.42% over SecureSGD, 14.08% over SecureSGD-AGM, 29.22% over AdLM, and 14.62% over DP-SGD. Furthermore, the accuracy of our model is consistent given different attacks with different adversarial perturbations μ_a under a rigorous DP protection ($\epsilon_t = 2.0$), compared with baseline approaches (Figure 5). In fact, when the attack size μ_a increases from 0.05 to 0.5, the conventional accuracies of the baseline approaches are remarkably reduced, i.e., a drop of 25.26% on average given the most effective baseline approach, SecureSGD. Meanwhile, there is a much smaller degradation (4.79% on average) in terms of the conventional accuracy observed in our StoBatch model. Figure 7 further shows that our StoBatch model is more accurate than baseline approaches (i.e., ϵ_r is set to 0.1 in PixelDP) in terms of certified accuracy in all cases, with a tight privacy budget set to 2.0 ($p < 2.04e-18$).

Scalability and Strong Iterative Attacks. First, we scale our model in terms of *adversarial training* in the CIFAR-10 data, i.e., the number of iterative attack steps is increased to T_μ=*200 in training*, and to T_a=*2,000 in testing*. The iterative batch-by-batch DP adversarial training (Alg. 1) is infeasible in this setting, taking over 30 days for one training with 600 epochs. Thanks to the distributed training, our StoBatch takes $\approxeq$ 3 days to finish the training ($\mathbb{N} = 1$, $\mathbb{M} = 4$). More importantly, our StoBatch achieves consistent accuracies under strong iterative attacks with T_a=$\{1,000; 2,000\}$, compared with the best baseline, i.e., SecureSGD (Figure 8). On average, across attack sizes $\mu_a \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and steps $T_a \in \{100, 500, 1000, 2000\}$, our StoBatch achieves 45.25±1.6% and 42.59±1.58% in conventional and certified accuracies, compared with 29.08±11.95% and 19.58±5.0% of SecureSGD ($p < 2.75e-20$).

• We achieve a similar improvement over the **Tiny ImageNet** with a ResNet18 model, i.e., *a larger dataset on a larger network*, ($\mathbb{N} = 1$, $\mathbb{M} = 20$) (Figure 9). On average, across attack sizes $\mu_a \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and steps $T_a \in \{100, 500, 1000, 2000\}$, our StoBatch achieves 29.78±4.8% and 28.31±1.58% in conventional and certified accuracies, compared with 8.99±5.95% and 8.72±5.5% of SecureSGD ($p < 1.55e-42$).

Key observations: (1) Incorporating ensemble adversarial learning into DP preservation, tightened sensitivity bounds, a random perturbation size μ_t at each training step, and composition robustness bounds in both input and latent spaces does enhance the consistency, robustness, and accuracy of DP model against different attacks with different levels of perturbations. These are key advantages of our mechanism; **(2)** As a result, our StoBatch model outperforms baseline algorithms, in terms of conventional and certified accuracies in most of the cases. It is clear that existing DP-preserving approaches have not been designed to withstand against adversarial examples; and **(3)** Our StoBatch training can help us to scale our mechanism to larger DP DNNs and datasets with distributed adversarial learning, without affecting the model accuracies and DP protections.

5. Conclusion

In this paper, we established a connection among DP preservation to protect the training data, adversarial learning, and certified robustness. A sequential composition robustness was introduced to generalize robustness given any sequential and bounded function of independent defensive mechanisms in both input and latent spaces. We addressed the trade-off among model utility, privacy loss, and robustness by tightening the global sensitivity bounds. We further developed a stochastic batch training mechanism to bypass the vanilla iterative batch-by-batch training in DP DNNs. The average errors of our approximation functions are always bounded by constant values. Last but not least, a new Monte Carlo Estimation was proposed to stabilize the estimation of the robustness bounds. Rigorous experiments conducted on benchmark datasets shown that our mechanism significantly enhances the robustness and scalability of DP DNNs. In future work, we will test our algorithms and models in the Baidu Fedcube platform (Baidu, 2020). In addition, we will evaluate our robustness bounds against synergistic attacks, in which adversarial examples can be combined with other attacks, such as Trojans (Gu et al., 2017; Liu et al., 2018), to create more lethal and stealthier threats (Pang et al., 2020).

Acknowledgements

The authors gratefully acknowledge the support from the National Science Foundation (NSF) grants CNS-1850094, CNS-1747798, CNS-1935928 / 1935923, and Adobe Unrestricted Research Gift.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. *arXiv:1607.00133*, 2016.

Abadi, M., Erlingsson, U., Goodfellow, I., McMahan, H. B., Mironov, I., Papernot, N., Talwar, K., and Zhang, L. On the protection of private information in machine learning systems: Two recent approches. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 1–6. IEEE, 2017.

Abbasi, M. and Gagné, C. Robustness to adversarial examples through an ensemble of specialists. *CoRR*, abs/1702.06856, 2017. URL http://arxiv.org/abs/1702.06856.

Apostol, T. *Calculus*. John Wiley & Sons, 1967.

Arfken, G. In *Mathematical Methods for Physicists (Third Edition)*. Academic Press, 1985.

Baidu. Fedcube, 2020. URL http://fedcube.baidu.com/.

Balle, B. and Wang, Y.-X. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 394–403, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/balle18a.html.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, May 2017. doi: 10.1109/SP.2017.49.

Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E., and Palamidessi, C. Broadening the scope of differential privacy using metrics. In De Cristofaro, E. and Wright, M. (eds.), *Privacy Enhancing Technologies*, pp. 82–102, 2013.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 854–863, International Convention Centre, Sydney, Australia, 06–11 Aug 2017.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1310–1320, Long Beach, California, USA, 09–15 Jun 2019.

Dong, Y., Liao, F., Pang, T., Hu, X., and Zhu, J. Discovering adversarial examples with momentum. *CoRR*, abs/1710.06081, 2017.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL http://dx.doi.org/10.1561/0400000042.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography*, pp. 265–284, 2006.

Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 1322–1333, 2015. doi: 10.1145/2810103.2813677.

Gao, J., Wang, B., and Qi, Y. Deepmask: Masking DNN models for robustness against adversarial samples. *CoRR*, abs/1702.06763, 2017. URL http://arxiv.org/abs/1702.06763.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. D. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017. URL http://arxiv.org/abs/1702.06280.

Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014. URL http://arxiv.org/abs/1412.5068.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017. URL http://arxiv.org/abs/1708.06733.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Hosseini, H., Chen, Y., Kannan, S., Zhang, B., and Poovendran, R. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

Kardan, N. and Stanley, K. O. Mitigating fooling with competitive overcomplete output layer neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 518–525, 2017.

Kolter, J. Z. and Wong, E. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017. URL http://arxiv.org/abs/1711.00851.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016a.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016b.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *arXiv:1802.03471*, 2018. URL https://arxiv.org/abs/1802.03471.

Lee, J. and Kifer, D. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1656–1665, 2018.

Li, B., Chen, C., Wang, W., and Carin, L. Second-order adversarial attack and certifiable robustness. *CoRR*, abs/1809.03113, 2018. URL http://arxiv.org/abs/1809.03113.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Matyasko, A. and Chau, L. P. Margin maximization for robust classification using deep learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 300–307, 2017.

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1702.04267.

Pang, R., Shen, H., Zhang, X., Ji, S., Vorobeychik, Y., Luo, X., Liu, A., and Wang, T. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2020.

Papernot, N. and McDaniel, P. Extending defensive distillation. *arXiv preprint arXiv:1705.05264*, 2017.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy*, pp. 372–387, March 2016a. doi: 10.1109/EuroSP.2016.36.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, May 2016b. doi: 10.1109/SP.2016.41.

Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., and Erlingsson, Ú. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

Phan, N., Wang, Y., Wu, X., and Dou, D. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *AAAI'16*, pp. 1309–1316, 2016.

Phan, N., Wu, X., and Dou, D. Preserving differential privacy in convolutional deep belief networks. *Machine Learning*, 2017a. doi: 10.1007/s10994-017-5656-2.

Phan, N., Wu, X., Hu, H., and Dou, D. Adaptive laplace mechanism: Differential privacy preservation in deep learning. In *IEEE ICDM'17*, 2017b.

Phan, N., Vu, M. N., Liu, Y., Jin, R., Dou, D., Wu, X., and Thai, M. T. Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pp. 4753–4759, 10–16 August 2019.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018. URL http://arxiv.org/abs/1801.09344.

Rudin, W. *Principles of Mathematical Analysis*. McGraw-Hill, 1976.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I. P., and Bubeck, S. Provably robust deep learning via adversarially trained smoothed classifiers. *CoRR*, abs/1906.04584, 2019.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems 31*, pp. 6103–6113. 2018.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *CCS'15*, pp. 1310–1321, 2015.

Song, L., Shokri, R., and Mittal, P. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. *arXiv e-prints*, art. arXiv:1905.10291, May 2019.

TensorFlow. URL https://github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/python/ops/nn_impl.py.

Operator norm. Operator norm, 2018. URL https://en.wikipedia.org/wiki/Operator_norm.

TinyImageNet. URL https://tiny-imagenet.herokuapp.com.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Wang, Q., Guo, W., Zhang, K., II, A. G. O., Xing, X., Giles, C. L., and Liu, X. Learning adversary-resistant deep neural networks. *CoRR*, abs/1612.01401, 2016.

Wu, B., Zhao, S., Sun, G., Zhang, X., Su, Z., Zeng, C., and Liu, Z. P3SGD: patient privacy preserving SGD for regularizing deep cnns in pathological image classification. In *CVPR*, 2019.

Xie, C., Wu, Y., van der Maaten, L., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *CoRR*, abs/1704.01155, 2017. URL http://arxiv.org/abs/1704.01155.

Xu, Z., Shi, S., Liu, X. A., Zhao, J., and Chen, L. An adaptive and fast convergent approach to differentially private deep learning. In *INFOCOM*, 2020.

Yu, L., Liu, L., Pu, C., Gursoy, M., and Truex, S. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 326–343, 2019.

Zhang, J., Zhang, Z., Xiao, X., Yang, Y., and Winslett, M. Functional mechanism: regression analysis under differential privacy. *PVLDB*, 5(11):1364–1375, 2012.

A. Notations and Terminologies

Table 1. Notations and Terminologies.

D and x	Training data with benign examples $x \in [-1, 1]^d$
$y = \{y_1, \ldots, y_K\}$	One-hot label vector of K categories
$f : \mathbb{R}^d \to \mathbb{R}^K$	Function/model f that maps inputs x to a vector of scores $f(x) = \{f_1(x), \ldots, f_K(x)\}$
$y_x \in y$	A single true class label of example x
$y(x) = \max_{k \in K} f_k(x)$	Predicted label for the example x given the function f
$x^{\text{adv}} = x + \alpha$	Adversarial example where α is the perturbation
$l_p(\mu) = \{\alpha \in \mathbb{R}^d : \|\alpha\|_p \leq \mu\}$	The l_p-norm ball of attack radius μ
(ϵ_r, δ_r)	Robustness budget ϵ_r and broken probability δ_r
$\mathbb{E}f_k(x)$	The expected value of $f_k(x)$
$\hat{\mathbb{E}}_{lb}$ and $\hat{\mathbb{E}}_{ub}$	Lower and upper bounds of the expected value $\hat{\mathbb{E}}f(x) = \frac{1}{n}\sum_n f(x)_n$
$a(x, \theta_1)$	Feature representation learning model with x and parameters θ_1
B_t	A batch of benign examples x_i
$\mathcal{R}_{B_t}(\theta_1)$	Data reconstruction function given B_t in $a(x, \theta_1)$
$\mathbf{h}_{1B_t} = \{\theta_1^T x_i\}_{x_i \in B_t}$	The values of all hidden neurons in the hidden layer $\mathbf{h}_1$ of $a(x, \theta_1)$ given the batch B_t
$\widetilde{\mathcal{R}}_{B_t}(\theta_1)$ and $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$	Approximated and perturbed functions of $\mathcal{R}_{B_t}(\theta_1)$
$\overline{x}_i$ and $\widetilde{x}_i$	Perturbed and reconstructed inputs x_i
$\Delta_{\mathcal{R}} = d(\beta + 2)$	Sensitivity of the approximated function $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$
$\overline{\mathbf{h}}_{1B_t}$	Perturbed affine transformation $\mathbf{h}_{1B_t}$
$\overline{x}_j^{\text{adv}} = x_j^{\text{adv}} + \frac{1}{m} Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$	DP adversarial examples crafting from benign example x_j
$\overline{B}_t$ and $\overline{B}_t^{\text{adv}}$	Sets of perturbed inputs $\overline{x}_i$ and DP adversarial examples $\overline{x}_j^{\text{adv}}$
$\mathcal{L}_{\overline{B}_t}(\theta_2)$	Loss function of perturbed benign examples in $\overline{B}_t$, given θ_2
$\Upsilon(f(\overline{x}_j^{\text{adv}}, \theta_2), y_j)$	Loss function of DP adversarial examples $\overline{x}_j^{\text{adv}}$, given θ_2
$\overline{\mathcal{L}}_{\overline{B}_t}(\theta_2)$	DP loss function for perturbed benign examples $\overline{B}_t$
$\mathcal{L}_{2\overline{B}_t}(\theta_2)$	A part of the loss function $\mathcal{L}_{\overline{B}_t}(\theta_2)$ that needs to be DP
$f(\mathcal{M}_1, \ldots, \mathcal{M}_s \| x)$	Composition scoring function given independent randomizing mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_s$
Δ_r^x and Δ_r^h	Sensitivities of x and h, given the perturbation $\alpha \in l_p(1)$
$(\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$	Privacy budget to protect the training data D
$(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$	Robustness size guarantee given an input x at the inference time

B. Functional Mechanism (Zhang et al., 2012)

Functional mechanism (Zhang et al., 2012) achieves ϵ-DP by perturbing the objective function $L_D(\theta)$ and then releasing the model parameter $\overline{\theta}$ minimizing the perturbed objective function $\overline{L}_D(\theta)$ instead of the original θ, given a private training dataset D. The mechanism exploits the polynomial representation of $L_D(\theta)$. The model parameter θ is a vector that contains $\mathbf{d}$ values $\theta_1, \ldots, \theta_{\mathbf{d}}$. Let $\phi(\theta)$ denote a product of $\theta_1, \ldots, \theta_{\mathbf{d}}$, namely, $\phi(\theta) = \theta_1^{c_1} \cdot \theta_2^{c_2} \cdots \theta_{\mathbf{d}}^{c_{\mathbf{d}}}$ for some $c_1, \ldots, c_{\mathbf{d}} \in \mathbb{N}$. Let $\Phi_j (j \in \mathbb{N})$ denote the set of all products of $\theta_1, \ldots, \theta_{\mathbf{d}}$ with degree j, i.e., $\Phi_j = \left\{\theta_1^{c_1} \cdot \theta_2^{c_2} \cdots \theta_{\mathbf{d}}^{c_{\mathbf{d}}} \middle| \sum_{a=1}^{\mathbf{d}} c_a = j\right\}$. By the Stone-Weierstrass Theorem (Rudin, 1976), any continuous and differentiable $L(x_i, \theta)$ can always be written as a polynomial of $\theta_1, \ldots, \theta_{\mathbf{d}}$, for some $J \in [0, \infty]$, i.e., $L(x_i, \theta) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \lambda_{\phi x_i} \phi(\theta)$ where $\lambda_{\phi x_i} \in \mathbb{R}$ denotes the coefficient of $\phi(\theta)$ in the polynomial.

For instance, the polynomial expression of the loss function in the linear regression is as follows: $L(x_i, \theta) = (y_i - x_i^\top \theta)^2 = y_i^2 - \sum_{j=1}^{d}(2y_i x_{ij})\theta_j + \sum_{1 \leq j,a \leq d}(x_{ij}x_{ia})\theta_j\theta_a$, where $d\ (= \mathbf{d})$ is the number of features in x_i. In fact, $L(x_i, \theta)$ only involves monomials in $\Phi_0 = \{1\}, \Phi_1 = \{\theta_1, \ldots, \theta_d\}$, and $\Phi_2 = \{\theta_i\theta_a | i, a \in [1, d]\}$. Each $\phi(\theta)$ has its own coefficient,

e.g., for θ_j, its polynomial coefficient $\lambda_{\phi_{x_i}} = -2y_i x_{ij}$. Similarly, $L_D(\theta)$ can be expressed as a polynomial of $\theta_1, \ldots, \theta_d$, as

$$L_D(\theta) = \sum_{x_i \in D} L(x_i, \theta) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \sum_{x_i \in D} \lambda_{\phi x_i} \phi(\theta) \tag{14}$$

To achieve ϵ-DP, $L_D(\theta)$ is perturbed by injecting Laplace noise $Lap(\frac{\Delta}{\epsilon})$ into its polynomial coefficients λ_ϕ, and then the model parameter $\overline{\theta}$ is derived to minimize the perturbed function $\overline{L}_D(\theta)$, where the global sensitivity $\Delta = 2\max_x \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \|\lambda_{\phi x}\|_1$ is derived given any two neighboring datasets. To guarantee that the optimization of $\overline{\theta} = \arg\min_\theta \overline{L}_D(\theta)$ achieves ϵ-DP without accessing the original data, i.e., that may potentially incur additional privacy leakage, grid search-based approaches are applied to learn the ϵ-DP parameters $\overline{\theta}$ with low loss $\overline{L}_D(\theta)$. Although this approach works well in simple tasks, i.e., logistic regression, it may not be optimal in large models, such as DNNs.

C. Pseudo-code of Adversarial Training (Kurakin et al., 2016b)

Let $l_p(\mu) = \{\alpha \in \mathbb{R}^d : \|\alpha\|_p \leq \mu\}$ be the l_p-norm ball of radius μ. One of the goals in adversarial learning is to minimize the risk over adversarial examples: $\theta^* = \arg\min_\theta \mathbb{E}_{(x, y_{\text{true}}) \sim \mathcal{D}} \big[\max_{\|\alpha\|_p \leq \mu} L\big(f(x+\alpha, \theta), y_x\big)\big]$, where an attack is used to approximate solutions to the inner maximization problem, and the outer minimization problem corresponds to training the model f with parameters θ over these adversarial examples $x^{\text{adv}} = x + \alpha$. There are two basic adversarial example attacks. The first one is a *single-step* algorithm, e.g., **FGSM** algorithm (Goodfellow et al., 2014), in which only a single gradient computation is required to find adversarial examples by solving the inner maximization $\max_{\|\alpha\|_p \leq \mu} L\big(f(x+\alpha, \theta), y_x\big)$. The second one is an *iterative* algorithm, e.g., **Iterative-FGSM** algorithm (Kurakin et al., 2016a), in which multiple gradients are computed and updated in T_μ small steps, each of which has a size of μ / T_μ.

Given a loss function:

$$L(\theta) = \frac{1}{m_1 + \xi m_2} \Big(\sum_{x_i \in B_t} \mathcal{L}\big(f(x_i, \theta), y_i\big) + \xi \sum_{x_j^{\text{adv}} \in B_t^{\text{adv}}} \Upsilon\big(f(x_j^{\text{adv}}, \theta), y_j\big) \Big) \tag{15}$$

where m_1 and m_2 correspondingly are the numbers of examples in B_t and B_t^{adv} at each training step. Algorithm 2 presents the vanilla adversarial training.

Algorithm 2 Adversarial Training (Kurakin et al., 2016b)

Input: Database D, loss function L, parameters θ, batch sizes m_1 and m_2, learning rate ϱ_t, parameter ξ

1: **Initialize** θ randomly
2: **for** $t \in [T]$ **do**
3: **Take** a random batch B_t with the size m_1, and a random batch B_a with the size m_2
4: Craft adversarial examples $B_t^{\text{adv}} = \{x_j^{\text{adv}}\}_{j \in [1, m_2]}$ from corresponding benign examples $x_j \in B_a$
5: **Descent:** $\theta \leftarrow \theta - \varrho_t \nabla_\theta L(\theta)$

D. Pseudo-code of Verified Inferring and StoBatch Training

Algorithm 3 Verified Inferring

Input: (an input x, attack size μ_a)

1: **Compute** robustness size $(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$ in Eq. 13 of x
2: **if** $(\frac{\kappa\varphi}{\kappa+\varphi})_{max} \geq \mu_a$ **then**
3: **Return** $isRobust(x) = True$, label k, $(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$
4: **else**
5: **Return** $isRobust(x) = False$, label k, $(\frac{\kappa\varphi}{\kappa+\varphi})_{max}$

Algorithm 4 StoBatch Training

Input: Database D, loss function L, parameters θ, batch size m, learning rate ϱ_t, privacy budgets: ϵ_1 and ϵ_2, robustness parameters: ϵ_r, Δ_r^x, and Δ_r^h, adversarial attack size μ_a, the number of invocations n, ensemble attacks A, parameters ψ and ξ, the size $|\mathbf{h}_\pi|$ of $\mathbf{h}_\pi$, a number of $\mathbb{N}$ random local trainers ($\mathbb{N} \leq N/(2m)$)

Draw Noise $\chi_1 \leftarrow [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^d$, $\chi_2 \leftarrow [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta$, $\chi_3 \leftarrow [Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})]^{|\mathbf{h}_\pi|}$
Randomly Initialize $\theta = \{\theta_1, \theta_2\}$, $\mathbf{B} = \{B_1, \ldots, B_{N/m}\}$ s.t. $\forall B \in \mathbf{B}$: B is a batch with the size m, $B_1 \cap \ldots \cap B_{N/m} = \emptyset$, and $B_1 \cup \ldots \cup B_{N/m} = D$, $\overline{\mathbf{B}} = \{\overline{B}_1, \ldots, \overline{B}_{N/m}\}$ where $\forall i \in [1, N/m] : \overline{B}_i = \{\overline{x} \leftarrow x + \frac{\chi_1}{m}\}_{x \in B_i}$
Construct a deep network f with **hidden layers** $\{\mathbf{h}_1 + \frac{2\chi_2}{m}, \ldots, \mathbf{h}_\pi\}$, where $\mathbf{h}_\pi$ is the last hidden layer
Distribute fixed and disjoint batches $\overline{\mathbf{B}}$ to $N/(2m)$ local trainers, each of which have two batches $\{\overline{B}_{i1}, \overline{B}_{i2}\}$ randomly picked from $\overline{\mathbf{B}}$ with $i \in [1, N/(2m)]$
for $t \in [T]$ **do**
  Randomly Pick $\mathbb{N}$ local trainers, each of which **Gets** the latest global parameters θ from the parameter server
  for $i \in [1, \mathbb{N}]$ **do**
    Assign $\overline{B}_{t,i} \leftarrow \overline{B}_{i1}$
    Ensemble DP Adversarial Examples:
    Draw Random Perturbation Value $\mu_t \in (0, 1]$, **Assign** $\overline{B}_{t,i}^{\text{adv}} \leftarrow \emptyset$
    for $l \in A$ **do**
      Take the next batch $\overline{B}_a \subset \overline{B}_{i2}$ with the size $m/|A|$
      $\forall \overline{x}_j \in \overline{B}_a$: **Craft** $\overline{x}_j^{\text{adv}}$ by using attack algorithm $A[l]$ with $l_\infty(\mu_t)$, $\overline{B}_{t,i}^{\text{adv}} \leftarrow \overline{B}_{t,i}^{\text{adv}} \cup \overline{x}_j^{\text{adv}}$
    Compute $\nabla_i\theta_1 \leftarrow \nabla_{\theta_1}\overline{\mathcal{R}}_{\overline{B}_{t,i} \cup \overline{B}_{t,i}^{\text{adv}}}(\theta_1)$, $\nabla_i\theta_2 \leftarrow \nabla_{\theta_2}\overline{L}_{\overline{B}_{t,i} \cup \overline{B}_{t,i}^{\text{adv}}}(\theta_2)$ with the noise $\frac{\chi_3}{m}$
    Send $\nabla_i\theta_1$ and $\nabla_i\theta_2$ to the parameter server
  Descent: $\theta_1 \leftarrow \theta_1 - \varrho_t \frac{1}{\mathbb{N}} \sum_{i \in [1,\mathbb{N}]} \nabla_i\theta_1$; $\theta_2 \leftarrow \theta_2 - \varrho_t \frac{1}{\mathbb{N}} \sum_{i \in [1,\mathbb{N}]} \nabla_i\theta_2$, on the parameter server
Output: $\epsilon = (\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$-DP parameters $\theta = \{\theta_1, \theta_2\}$, robust model with an ϵ_r budget

E. Proof of Lemma 2

Proof 1 *Assume that B_t and B_t' differ in the last tuple, x_m (x_m'). Then,*

$$\Delta_{\mathcal{R}} = \sum_{j=1}^{d} \Big[\| \sum_{x_i \in B_t} \frac{1}{2} h_i - \sum_{x_i' \in B_t'} \frac{1}{2} h_i' \|_1 + \| \sum_{x_i \in B_t} x_{ij} - \sum_{x_i' \in B_t'} x_{ij}' \|_1 \Big]$$
$$\leq 2 \max_{x_i} \sum_{j=1}^{d} (\| \frac{1}{2} h_i \|_1 + \| x_{ij} \|_1) \leq d(\beta + 2)$$

F. Proof of Lemma 3

Proof 2 *Regarding the computation of $\mathbf{h}_{1\overline{B}_t} = \{\overline{\theta}_1^T \overline{x}_i\}_{\overline{x}_i \in \overline{B}_t}$, we can see that $h_i = \overline{\theta}_1^T \overline{x}_i$ is a linear function of x. The sensitivity of a function h is defined as the maximum change in output, that can be generated by a change in the input (Lecuyer et al., 2018). Therefore, the global sensitivity of $\mathbf{h}_1$ can be computed as follows:*

$$\Delta_{\mathbf{h}_1} = \frac{\| \sum_{\overline{x}_i \in \overline{B}_t} \overline{\theta}_1^T \overline{x}_i - \sum_{\overline{x}_i' \in \overline{B}_t'} \overline{\theta}_1^T \overline{x}_i' \|_1}{\| \sum_{\overline{x}_i \in \overline{B}_t} \overline{x}_i - \sum_{\overline{x}_i' \in \overline{B}_t'} \overline{x}_i' \|_1} \leq \max_{x_i \in B_t} \frac{\| \overline{\theta}_1^T \overline{x}_i \|_1}{\| \overline{x}_i \|_1} \leq \| \overline{\theta}_1^T \|_{1,1}$$

following matrix norms (Operator norm, 2018): $\| \overline{\theta}_1^T \|_{1,1}$ is the maximum 1-norm of θ_1's columns. By injecting Laplace noise $Lap(\frac{\Delta_{\mathbf{h}_1}}{\epsilon_1})$ into $\mathbf{h}_{1B_t}$, i.e., $\overline{\mathbf{h}}_{1\overline{B}_t} = \{\overline{\theta}_1^T \overline{x}_i + Lap(\frac{\Delta_{\mathbf{h}_1}}{\epsilon_1})\}_{\overline{x}_i \in \overline{B}_t}$, we can preserve ϵ_1-DP in the computation of $\overline{\mathbf{h}}_{1\overline{B}_t}$. Let us set $\Delta_{\mathbf{h}_1} = \| \overline{\theta}_1^T \|_{1,1}$, $\gamma = \frac{2\Delta_{\mathcal{R}}}{m\Delta_{\mathbf{h}_1}}$, and χ_2 drawn as a Laplace noise $[Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta$, in our mechanism, the perturbed

affine transformation $\overline{\mathbf{h}}_{1\overline{B}_t}$ *is presented as:*

$$\begin{aligned}\overline{\mathbf{h}}_{1\overline{B}_t} &= \{\overline{\theta}_1^T\overline{x}_i + \frac{2\chi_2}{m}\}_{\overline{x}_i\in\overline{B}_t} = \{\overline{\theta}_1^T\overline{x}_i + \frac{2}{m}[Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta\}_{\overline{x}_i\in\overline{B}_t}\\ &= \{\overline{\theta}_1^T\overline{x}_i + [Lap(\frac{\gamma\Delta_{\mathbf{h}_1}}{\epsilon_1})]^\beta\}_{\overline{x}_i\in\overline{B}_t} = \{\overline{\theta}_1^T\overline{x}_i + [Lap(\frac{\Delta_{\mathbf{h}_1}}{\epsilon_1/\gamma})]^\beta\}_{\overline{x}_i\in\overline{B}_t}\end{aligned}$$

This results in an (ϵ_1/γ)*-DP affine transformation* $\overline{\mathbf{h}}_{1B_t} = \{\overline{\theta}_1^T\overline{x}_i + [Lap(\frac{\Delta_{\mathbf{h}_1}}{\epsilon_1/\gamma})]^\beta\}_{\overline{x}_i\in\overline{B}_t}$.

Similarly, the perturbed inputs $\overline{B}_t = \{\overline{x}_i\}_{\overline{x}_i\in\overline{B}_t} = \{x_i + \frac{\chi_1}{m}\}_{x_i\in B_t} = \{x_i + [Lap(\frac{\Delta_{\mathbf{x}}}{\epsilon_1/\gamma_{\mathbf{x}}})]^d\}_{x_i\in B_t}$*, where* $\Delta_{\mathbf{x}}$ *is the sensitivity measuring the maximum change in the input layer that can be generated by a change in the batch* B_t *and* $\gamma_{\mathbf{x}} = \frac{\Delta_{\mathcal{R}}}{m\Delta_{\mathbf{x}}}$*. Following (Lecuyer et al., 2018),* $\Delta_{\mathbf{x}}$ *can be computed as follows:* $\Delta_{\mathbf{x}} = \frac{\|\sum_{x_i\in B_t}\overline{x}_i - \sum_{x'_i\in B'_t}\overline{x}'_i\|_1}{\|\sum_{x_i\in B_t}\overline{x}_i - \sum_{x'_i\in B'_t}\overline{x}'_i\|_1} = 1$*. As a result, the computation of* $\overline{B}_t$ *is* $(\epsilon_1/\gamma_{\mathbf{x}})$*-DP. Consequently, Lemma 3 does hold.*

G. Proof of Theorem 1

Proof 3 *Given* χ_1 *drawn as a Laplace noise* $[Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^d$ *and* χ_2 *drawn as a Laplace noise* $[Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta$*, the perturbation of the coefficient* $\phi \in \Phi = \{\frac{1}{2}h_i, x_i\}$*, denoted as* $\overline{\phi}$*, can be rewritten as follows:*

$$\text{for } \phi\in\{x_i\}: \overline{\phi} = \sum_{x_i\in B}(\phi_{x_i} + \frac{\chi_1}{m}) = \sum_{x_i\in B}\phi_{x_i} + \chi_1 = \sum_{x_i\in B}\phi_{x_i} + [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^d \tag{16}$$

$$\text{for } \phi\in\{\frac{1}{2}h_i\}: \overline{\phi} = \sum_{x_i\in B}\frac{1}{2}(h_i + \frac{2\chi_2}{m}) = \sum_{x_i\in B}(\phi_{x_i} + \frac{\chi_2}{m}) = \sum_{x_i\in B}\phi_{x_i} + \chi_2 = \sum_{x_i\in B}\phi_{x_i} + [Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^\beta \tag{17}$$

we have

$$Pr\big(\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)\big) = \prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\big(-\frac{\epsilon_1\|\sum_{x_i\in B_t}\phi_{x_i} - \overline{\phi}\|_1}{\Delta_{\mathcal{R}}}\big)$$

$\Delta_{\mathcal{R}}$ *is set to* $d(\beta+2)$*, we have that:*

$$\begin{aligned}\frac{Pr\big(\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)\big)}{Pr\big(\overline{\mathcal{R}}_{\overline{B}'_t}(\theta_1)\big)} &= \frac{\prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\big(-\frac{\epsilon_1\|\sum_{x_i\in B_t}\phi_{x_i} - \overline{\phi}\|_1}{\Delta_{\mathcal{R}}}\big)}{\prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\big(-\frac{\epsilon_1\|\sum_{x'_i\in B'_t}\phi_{x'_i} - \overline{\phi}\|_1}{\Delta_{\mathcal{R}}}\big)}\\ &\le \prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\Big(\frac{\epsilon_1}{\Delta_{\mathcal{R}}}\Big\|\sum_{x_i\in B_t}\phi_{x_i} - \sum_{x'_i\in B'_t}\phi_{x'_i}\Big\|_1\Big)\\ &\le \prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\Big(\frac{\epsilon_1}{\Delta_{\mathcal{R}}}2\max_{x_i\in B_t}\big\|\phi_{x_i}\big\|_1\Big) \le \exp\Big(\frac{\epsilon_1 d(\beta+2)}{\Delta_{\mathcal{R}}}\Big) = \exp(\epsilon_1)\end{aligned} \tag{18}$$

Consequently, the computation of $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ *preserves* ϵ_1*-DP in Alg. 1 (**Result 1**). To show that gradient descent-based optimizers can be used to optimize the objective function* $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ *in learning private parameters* θ_1*, we prove that all the computations on top of the perturbed data* $\overline{B}_t$*, including* h_i*,* $\overline{h}_i$*,* $\widetilde{x}_i$*, gradients and descent, are DP without incurring any additional information from the original data, as follows.*

First, by following the post-processing property in DP (Dwork & Roth, 2014), it is clear that the computations of $\mathbf{h}_{1\overline{B}_t} = \{h_i\}_{\overline{x}_i\in\overline{B}_t} = \theta_1^T\{\overline{x}_i\}_{\overline{x}_i\in\overline{B}_t}$ *is* $(\epsilon_1/\gamma_{\mathbf{x}})$*-DP. As in Lemma 3, we also have that* $\overline{\mathbf{h}}_{1\overline{B}_t} = \{h_i + \frac{2\chi_2}{m}\}_{\overline{x}_i\in\overline{B}_t}$ *is* (ϵ_1/γ)*-DP. Given this, it is obvious that* $\widetilde{\mathbf{x}}_i = \{\widetilde{x}_i\}_{\overline{x}_i\in\overline{B}_t} = \theta_1\{\overline{h}_i\}_{\overline{x}_i\in\overline{B}_t}$ *is* (ϵ_1/γ)*-DP, i.e., the post-processing property in DP. In addition, the computations of* $\mathbf{h}_{1\overline{B}_t}$*,* $\overline{\mathbf{h}}_{1\overline{B}_t}$*, and* $\widetilde{\mathbf{x}}_i$ *do not access the original data* B_t*. Therefore, they do not incur any additional information from the private data, except the privacy loss measured by* $(\epsilon_1/\gamma_{\mathbf{x}})$*-DP, since the computations of* $\overline{\mathbf{h}}_{1\overline{B}_t}$ *and* $\widetilde{\mathbf{x}}_i$ *are based on the* $(\epsilon_1/\gamma_{\mathbf{x}})$*-DP* $\mathbf{h}_{1\overline{B}_t}$*. (**Result 2**)*

Second, the gradient of a particular parameter θ_{1j}, with $\forall j \in [1, d]$, can be computed as follows:

$$\forall j \in [1,d] : \nabla_{\theta_{1j}} \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1) = \frac{\delta \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)}{\delta \theta_{1j}} = \sum_{i=1}^{m} \overline{h}_i (\frac{1}{2} - \overline{x}_{ij}) \tag{19}$$

$$= \sum_{i=1}^{m} (h_i + \frac{2\chi_2}{m})(\frac{1}{2} - \overline{x}_{ij}) \tag{20}$$

$$= [\sum_{i=1}^{m} h_i (\frac{1}{2} - \overline{x}_{ij})] + \chi_2 - [\frac{2\chi_2}{m} \sum_{i=1}^{m} \overline{x}_{ij}] \tag{21}$$

*In Eq. 21, we have that $\sum_{i=1}^{m} \overline{x}_{ij} = (\sum_{i=1}^{m} x_{ij}) + Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ (Eq. 16), which is $(\epsilon_1/\gamma_{\mathbf{x}})$-DP. Therefore, the term $\frac{2\chi_2}{m} \sum_{i=1}^{m} \overline{x}_{ij}$ also is $(\epsilon_1/\gamma_{\mathbf{x}})$-DP (the post-processing property in DP). (**Result 3**)*

Regarding the term $\sum_{i=1}^{m} h_i(\frac{1}{2} - \overline{x}_{ij})$ in Eq. 21, its global sensitivity given two arbitrary neighboring batches, denoted as Δ_g, can be bounded as follows: $\Delta_g \leq 2 \max_{\overline{x}_i} \|h_i(\frac{1}{2} - \overline{x}_{ij})\|_1 = 3\beta$. As a result, we have that:

$$[\sum_{i=1}^{m} h_i (\frac{1}{2} - \overline{x}_{ij})] + \chi_2 = [\sum_{i=1}^{m} h_i (\frac{1}{2} - \overline{x}_{ij})] + [Lap(\frac{\Delta_g}{\epsilon_1 / \frac{\Delta_{\mathcal{R}}}{\Delta_g}})]^{\beta} \tag{22}$$

*which is $(\epsilon_1 / \frac{\Delta_{\mathcal{R}}}{\Delta_g})$-DP. (**Result 4**)*

*From Results 3 and 4, the computation of gradients $\nabla_{\theta_{1j}} \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ is $(\epsilon_1 / \frac{\Delta_{\mathcal{R}}}{\Delta_g} + \epsilon_1/\gamma_{\mathbf{x}})$-DP, since: **(1)** The computations of the two terms in Eq. 21 can be treated as two independent DP-preserving mechanisms applied on the perturbed batch $\overline{B}_t$; and **(2)** This is true for every dimension $j \in [1, d]$, each of which $\nabla_{\theta_{1j}}$ is independently computed and bounded. It is important to note that this result is different from the traditional DPSGD (Abadi et al., 2016), in which the parameter gradients are jointly clipped by a l_2-norm constant bound, such that Gaussian noise can be injected to achieve DP. In addition, as in Eq. 19, the computation of $\nabla_{\theta_{1j}} \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ only uses $(\epsilon_1/\gamma_{\mathbf{x}})$-DP $\overline{B}_t = \{\overline{x}_i\}_{\overline{x}_i \in \overline{B}_t}$ and (ϵ_1/γ)-DP $\overline{\mathbf{h}}_{1\overline{B}_t}$, without accessing the original data. Basically, $\overline{\mathbf{h}}_{1\overline{B}_t}$ is computed on top of $\overline{B}_t$, without touching any benign example. Therefore, it does not incur any additional information from the private data, except the privacy loss $(\epsilon_1 / \frac{\Delta_{\mathcal{R}}}{\Delta_g} + \epsilon_1/\gamma_{\mathbf{x}})$-DP. In practice, we observed that $\epsilon_1/\gamma_{\mathbf{x}} \gg \epsilon_1 / \frac{\Delta_{\mathcal{R}}}{\Delta_g} \cong \epsilon_1 \times 1e-3$, which is tiny. We can simply consider that the computation of gradients $\nabla_{\theta_{1j}} \overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}})$-DP without affecting the general DP protection. In addition to the gradient computation, the descent operations are simply post-processing steps without consuming any further privacy budget. (**Result 5**)*

From Results 1, 2, and 5, we have shown that all the computations on top of $(\epsilon_1/\gamma_{\mathbf{x}})$-DP $\overline{B}_t$, including parameter gradients and gradient descents, clearly are DP without accessing the original data; therefore, they do not incur any additional information from the private data (the post-processing property in DP). As a result, gradient descent-based approaches can be applied to optimize $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ in Alg. 1. The total privacy budget to learn the perturbed optimal parameters $\overline{\theta}_1$ in Alg. 1 is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP, where the $\epsilon_1/\gamma_{\mathbf{x}}$ is counted for the perturbation on the batch of benign examples B_t.

Consequently, Theorem 1 does hold.

H. Proof of Lemma 4

Proof 4 *Assume that $\overline{B}_t$ and $\overline{B}'_t$ differ in the last tuple, and $\overline{x}_m$ ($\overline{x}'_m$) be the last tuple in $\overline{B}_t$ ($\overline{B}'_t$), we have that*

$$\Delta_{\mathcal{L}2} = \sum_{k=1}^{K} \Big\| \sum_{\overline{x}_i \in \overline{B}_t} (\mathbf{h}_{\pi i} y_{ik}) - \sum_{\overline{x}'_i \in \overline{B}'_t} (\mathbf{h}'_{\pi i} y'_{ik}) \Big\|_1 = \sum_{k=1}^{K} \big\| \mathbf{h}_{\pi m} y_{mk} - \mathbf{h}'_{\pi m} y'_{mk} \big\|_1$$

Since y_{mk} and y'_{mk} are one-hot encoding, we have that $\Delta_{\mathcal{L}2} \leq 2 \max_{\overline{x}_i} \|\mathbf{h}_{\pi i}\|_1$. Given $\mathbf{h}_{\pi i} \in [-1, 1]$, we have

$$\Delta_{\mathcal{L}2} \leq 2|\mathbf{h}_{\pi}| \tag{23}$$

Lemma 4 does hold.

I. Proof of Theorem 3

Proof 5 *Let $\overline{B}_t$ and $\overline{B}'_t$ be neighboring batches of benign examples, and χ_3 drawn as Laplace noise $[Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})]^{|\mathbf{h}_\pi|}$, the perturbations of the coefficients $\mathbf{h}_{\pi i}y_{ik}$ can be rewritten as:*

$$\overline{\mathbf{h}}_{\pi i}\overline{y}_{ik} = \sum_{\overline{x}_i}(\mathbf{h}_{\pi i}y_{ik} + \frac{\chi_3}{m}) = \sum_{\overline{x}_i}(\mathbf{h}_{\pi i}y_{ik}) + [Lap(\frac{\Delta_{\mathcal{L}2}}{\epsilon_2})]^{|\mathbf{h}_\pi|}$$

Since all the coefficients are perturbed, and given $\Delta_{\mathcal{L}2} = 2|\mathbf{h}_\pi|$, we have that

$$\begin{aligned}
\frac{Pr(\mathcal{L}_{\overline{B}_t}(\theta_2))}{Pr(\mathcal{L}_{\overline{B}'_t}(\theta_2))} &= \frac{Pr(\mathcal{L}_{1\overline{B}_t}(\theta_2))}{Pr(\mathcal{L}_{1\overline{B}'_t}(\theta_2))} \times \frac{Pr(\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2))}{Pr(\overline{\mathcal{L}}_{2\overline{B}'_t}(\theta_2))} \\
&\leq e^{\epsilon_1/\gamma}\sum_{k=1}^{K}\frac{\exp(-\frac{\epsilon_2\|\sum_{\overline{x}_i}\mathbf{h}_{\pi i}y_{ik} - \overline{\mathbf{h}}_{\pi i}\overline{y}_{ik}\|_1}{\Delta_{\mathcal{L}2}})}{\exp(-\frac{\epsilon_2\|\sum_{\overline{x}'_i}\mathbf{h}_{\pi i}y_{ik} - \overline{\mathbf{h}}_{\pi i}\overline{y}_{ik}\|_1}{\Delta_{\mathcal{L}2}})} \\
&\leq e^{\epsilon_1/\gamma}\sum_{k=1}^{K}\exp(\frac{\epsilon_2}{\Delta_{\mathcal{L}2}}\big\|\sum_{\overline{x}_i}\mathbf{h}_{\pi i}y_{ik} - \sum_{\overline{x}'_i}\mathbf{h}_{\pi i}y_{ik}\big\|_1) \\
&\leq e^{\epsilon_1/\gamma}\exp(\frac{\epsilon_2}{\Delta_{\mathcal{L}2}}2\max_{\overline{x}_i}\|\mathbf{h}_{\pi i}\|_1) = e^{\epsilon_1/\gamma+\epsilon_2}
\end{aligned}$$

The computation of $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)$ preserves $(\epsilon_1/\gamma + \epsilon_2)$-differential privacy. Similar to Theorem 1, the gradient descent-based optimization of $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)$ does not access additional information from the original input $x_i \in B_t$. It only reads the (ϵ_1/γ)-DP $\overline{\mathbf{h}}_{1\overline{B}_t} = \{h_i + \frac{2\chi_2}{m}\}_{\overline{x}_i\in\overline{B}_t}$. Consequently, the optimal perturbed parameters $\overline{\theta}_2$ derived from $\overline{\mathcal{L}}_{2\overline{B}_t}(\theta_2)$ are $(\epsilon_1/\gamma + \epsilon_2)$-DP.

J. Proofs of Theorem 2 and Theorem 4

Proof 6 *First, we optimize for a single draw of noise during training (Line 3, Alg. 1) and all the batches of perturbed benign examples are disjoint and fixed across epochs. As a result, the computation of $\overline{x}_i$ is equivalent to a data preprocessing step with DP, which does not incur any additional privacy budget consumption over T training steps (the post-processing property of DP) (**Result 1**). That is different from repeatedly applying a DP mechanism on either the same or overlapping datasets causing the accumulation of the privacy budget.*

Now, we show that our algorithm achieves DP at the dataset level D. Let us consider the computation of the first hidden layer, given any two neighboring datasets D and D' differing at most one tuple $\overline{x}_e \in D$ and $\overline{x}'_e \in D'$. For any $O = \prod_{i=1}^{N/m} o_i \in \prod_{i=1}^{N/m}\overline{\mathbf{h}}_{1\overline{B}_i}(\in \mathbb{R}^{\beta\times m})$, we have that

$$\frac{P(\overline{\mathbf{h}}_{1D} = O)}{P(\overline{\mathbf{h}}_{1D'} = O)} = \frac{P(\overline{\mathbf{h}}_{1\overline{B}_1} = o_1)\dots P(\overline{\mathbf{h}}_{1\overline{B}_{N/m}} = o_{N/m})}{P(\overline{\mathbf{h}}_{1\overline{B}'_1} = o_1)\dots P(\overline{\mathbf{h}}_{1\overline{B}'_{N/m}} = o_{N/m})} \tag{24}$$

By having disjoint and fixed batches, we have that:

$$\exists!\tilde{B} \in \overline{\mathbf{B}} \text{ s.t. } x_e \in \tilde{B} \text{ and } \exists!\tilde{B}' \in \overline{\mathbf{B}}' \text{ s.t. } x'_e \in \tilde{B}' \tag{25}$$

From Eqs. 24, 25, and Lemma 3, we have that

$$\forall\overline{B} \in \mathbf{B}, \overline{B} \neq \tilde{B} : \overline{B} = \overline{B}' \Rightarrow \frac{P(\overline{\mathbf{h}}_{1\overline{B}} = o)}{P(\overline{\mathbf{h}}_{1\overline{B}'} = o)} = 1 \tag{26}$$

$$\textit{Eqs. 25 and 26} \Rightarrow \frac{P(\overline{\mathbf{h}}_{1D} = O)}{P(\overline{\mathbf{h}}_{1D'} = O)} = \frac{P(\overline{\mathbf{h}}_{1\tilde{B}} = \tilde{o})}{P(\overline{\mathbf{h}}_{1\tilde{B}'} = \tilde{o})} \leq e^{\epsilon_1/\gamma} \tag{27}$$

As a result, the computation of $\overline{\mathbf{h}}_{1D}$ is (ϵ_1/γ)-DP given the data D, since the Eq. 27 does hold for any tuple $x_e \in D$. That is consistent with the parallel composition property of DP, in which batches can be considered disjoint datasets given $\overline{\mathbf{h}}_{1\overline{B}}$ as a DP mechanism (Dwork & Roth, 2014).

*This does hold across epochs, since batches $\overline{\mathbf{B}}$ are disjoint and fixed among epochs. At each training step $t \in [1, T]$, the computation of $\overline{\mathbf{h}}_{1\overline{B}_t}$ does not access the original data. It only reads the perturbed batch of inputs $\overline{B}_t$, which is $(\epsilon_1/\gamma_{\mathbf{x}})$-DP (Lemma 3). Following the post-processing property in DP (Dwork & Roth, 2014), the computation of $\overline{\mathbf{h}}_{1\overline{B}_t}$ does not incur any additional information from the original data across T training steps. (**Result 2**)*

Similarly, we show that the optimization of the function $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP across T training steps. As in Theorem 1 and Proof 3, we have that $Pr\big(\overline{\mathcal{R}}_{\overline{B}}(\theta_1)\big) = \prod_{j=1}^{d}\prod_{\phi\in\Phi}\exp\big(-\frac{\epsilon_1\|\sum_{x_i\in B}\phi_{x_i}-\overline{\phi}\|_1}{\Delta_{\mathcal{R}}}\big)$, where $\overline{B} \in \overline{\mathbf{B}}$. Given any two perturbed neighboring datasets $\overline{D}$ and $\overline{D}'$ differing at most one tuple $\overline{x}_e \in \overline{D}$ and $\overline{x}'_e \in \overline{D}'$:

$$\frac{Pr\big(\overline{\mathcal{R}}_{\overline{D}}(\theta_1)\big)}{Pr\big(\overline{\mathcal{R}}_{\overline{D}'}(\theta_1)\big)} = \frac{Pr\big(\overline{\mathcal{R}}_{\overline{B}_1}(\theta_1)\big)\ldots Pr\big(\overline{\mathcal{R}}_{\overline{B}_{N/m}}(\theta_1)\big)}{Pr\big(\overline{\mathcal{R}}_{\overline{B}'_1}(\theta_1)\big)\ldots Pr\big(\overline{\mathcal{R}}_{\overline{B}'_{N/m}}(\theta_1)\big)} \tag{28}$$

From Eqs. 25, 28, and Theorem 1, we have that

$$\forall \overline{B} \in \mathbf{B}, \overline{B} \neq \tilde{B} : \overline{B} = \overline{B}' \Rightarrow \frac{P\big(\overline{\mathcal{R}}_{\overline{B}}(\theta_1)\big)}{P\big(\overline{\mathcal{R}}_{\overline{B}'}(\theta_1)\big)} = 1 \tag{29}$$

$$\text{Eqs. 28 and 29} \Rightarrow \frac{P\big(\overline{\mathcal{R}}_{\overline{D}}(\theta_1)\big)}{P\big(\overline{\mathcal{R}}_{\overline{D}'}(\theta_1)\big)} = \frac{P\big(\overline{\mathcal{R}}_{\tilde{B}}(\theta_1)\big)}{P\big(\overline{\mathcal{R}}_{\tilde{B}'}(\theta_1)\big)} \leq e^{\epsilon_1} \tag{30}$$

*As a result, the optimization of $\overline{\mathcal{R}}_{\overline{D}}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP given the data $\overline{D}$ (which is $\epsilon_1/\gamma_{\mathbf{x}}$-DP (Lemma 3)), since the Eq. 30 does hold for any tuple $\overline{x}_e \in \overline{D}$. This is consistent with the parallel composition property in DP (Dwork & Roth, 2014), in which batches can be considered disjoint datasets and the optimization of the function on one batch does not affect the privacy guarantee in any other batch, even the objective function given one batch can be slightly different from the objective function given any other batch in $\overline{\mathbf{B}}$. In addition, $\forall t \in [1, T]$, the optimization of $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ does not use any additional information from the original data D. Consequently, the privacy budget is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$ across T training steps, following the post-processing property in DP (Dwork & Roth, 2014) (**Result 3**).*

Similarly, we can also prove that optimizing the data reconstruction function $\overline{\mathcal{R}}_{\overline{B}_t^{adv}}(\theta_1)$ given the DP adversarial examples crafted in Eqs. 7 and 8, i.e., $\overline{x}_j^{adv}$, is also $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP given $t \in [1, T]$ on the training data D. First, DP adversarial examples $\overline{x}_j^{adv}$ are crafted from perturbed benign examples $\overline{x}_j$. As a result, the computation of the batch $\overline{B}_t^{adv}$ of DP adversarial examples is 1) $(\epsilon_1/\gamma_{\mathbf{x}})$-DP (the post-processing property of DP (Dwork & Roth, 2014)), and 2) does not access the original data $\forall t \in [1, T]$. In addition, the computation of $\overline{\mathbf{h}}_{1\overline{B}_t^{adv}}$ and the optimization of $\overline{\mathcal{R}}_{\overline{B}_t^{adv}}(\theta_1)$ correspondingly are ϵ_1/γ-DP and ϵ_1-DP. In fact, the data reconstruction function $\overline{\mathcal{R}}_{\overline{B}_t^{adv}}$ is presented as follows:

$$\begin{aligned}
\overline{\mathcal{R}}_{\overline{B}_t^{adv}}(\theta_1) &= \sum_{\overline{x}_j^{adv}\in\overline{B}_t^{adv}}\Big[\sum_{i=1}^{d}\big(\frac{1}{2}\theta_{1i}\overline{h}_j^{adv}\big) - \overline{x}_j^{adv}\widetilde{x}_j^{adv}\Big] \\
&= \sum_{\overline{x}_j^{adv}\in\overline{B}_t^{adv}}\Big[\sum_{i=1}^{d}\big(\frac{1}{2}\theta_{1i}\overline{h}_j^{adv}\big) - \overline{x}_j\widetilde{x}_j^{adv} - \mu\cdot sign\Big(\nabla_{\overline{x}_j}\mathcal{L}\big(f(\overline{x}_j,\theta), y(\overline{x}_j)\big)\Big)\widetilde{x}_j^{adv}\Big] \\
&= \sum_{\overline{x}_j^{adv}\in\overline{B}_t^{adv}}\Big[\sum_{i=1}^{d}\big(\frac{1}{2}\theta_{1i}\overline{h}_j^{adv}\big) - \overline{x}_j\widetilde{x}_j^{adv}\Big] - \sum_{\overline{x}_j^{adv}\in\overline{B}_t^{adv}}\mu\cdot sign\Big(\nabla_{\overline{x}_j}\mathcal{L}\big(f(\overline{x}_j,\theta), y(\overline{x}_j)\big)\Big)\widetilde{x}_j^{adv}
\end{aligned} \tag{31}$$

*where $h_j^{adv} = \theta_1^T\overline{x}_j^{adv}, \overline{h}_j^{adv} = h_j^{adv} + \frac{2}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})$, and $\widetilde{x}_j^{adv} = \theta_1\overline{h}_j^{adv}$. The right summation component in Eq. 31 does not disclose any additional information, since the $sign(\cdot)$ function is computed from perturbed benign examples (the post-processing property in DP (Dwork & Roth, 2014)). Meanwhile, the left summation component has the same form with $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ in Eq. 6. Therefore, we can employ the Proof 3 in Theorem 1, by replacing the coefficients $\Phi = \{\frac{1}{2}h_i, x_i\}$ with $\Phi = \{\frac{1}{2}h_j^{adv}, x_j\}$ to prove that the optimization of $\overline{\mathcal{R}}_{\overline{B}_t^{adv}}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP. As a result, Theorem 2 does hold. (**Result 4**)*

In addition to the Result 4, by applying the same analysis in Result 3, we can further show that the optimization of $\overline{\mathcal{R}}_{D^{adv}}(\theta_1)$ is $(\epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1)$-DP given the DP adversarial examples D^{adv} crafted using the data $\overline{D}$ across T training steps,

since batches used to created DP adversarial examples are disjoint and fixed across epochs. It is also straightforward to conduct the same analysis in Result 2, in order to prove that the computation of the first affine transformation $\overline{\mathbf{h}}_{1\overline{B}_t^{adv}} = \{\overline{\theta}_1^T \overline{x}_j^{adv} + \frac{2}{m} Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})\}_{\overline{x}_j^{adv} \in \overline{B}_t^{adv}}$ *given the batch of DP adversarial examples* $\overline{B}_t^{adv}$*, is* (ϵ_1/γ)*-DP with* $t \in [1, T]$ *training steps. This is also true given the data level* D^{adv}*. (**Result 5**)*

Regarding the output layer, the Algorithm 1 preserves $(\epsilon_1/\gamma + \epsilon_2)$*-DP in optimizing the adversarial objective function* $\overline{L}_{\overline{B}_t \cup \overline{B}_t^{adv}}(\theta_2)$ *(Theorem 3). We apply the same technique to preserve* $(\epsilon_1/\gamma + \epsilon_2)$*-DP across* T *training steps given disjoint and fixed batches derived from the private training data* D*. In addition, as our objective functions* $\overline{\mathcal{R}}$ *and* $\overline{L}$ *are always optimized given two disjoint batches* $\overline{B}_t$ *and* $\overline{B}_t^{adv}$*, the privacy budget used to preserve DP in these functions is* $(\epsilon_1 + \epsilon_1/\gamma + \epsilon_2)$*, following the parallel composition property in DP (Dwork & Roth, 2014). (**Result 6**)*

*With the **Results 1-6**, all the computations and optimizations in the Algorithm 1 are DP following the post-processing property in DP (Dwork & Roth, 2014), by working on perturbed inputs and perturbed coefficients. The crafting and utilizing processes of DP adversarial examples based on the perturbed benign examples do not disclose any additional information. The optimization of our DP adversarial objective function at the output layer is DP to protect the ground-truth labels. More importantly, the DP guarantee in learning given the whole dataset level* $\overline{D}$ *is equivalent to the DP guarantee in learning on disjoint and fixed batches across epochs. Consequently, Algorithm 1 preserves* $(\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2)$*-DP in learning private parameters* $\overline{\theta} = \{\overline{\theta}_1, \overline{\theta}_2\}$ *given the training data* D *across* T *training steps. Note that the* $\epsilon_1/\gamma_{\mathbf{x}}$ *is counted for the perturbation on the benign examples. Theorem 4 does hold.*

K. Proof of Lemma 5

Proof 7 *Thanks to the sequential composition theory in DP (Dwork & Roth, 2014),* $f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)$ *is* $(\sum_s \epsilon_s)$*-DP, since for any* $O = \prod_{s=1}^S o_s \in \prod_{s=1}^S f^s(x) (\in \mathbb{R}^K)$*, we have that*

$$\frac{P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x) = O\big)}{P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha) = O\big)} = \frac{P(\mathcal{M}_1 f(x) = o_1) \ldots P(\mathcal{M}_S f(x) = o_S)}{P(\mathcal{M}_1 f(x + \alpha) = o_1) \ldots P(\mathcal{M}_S f(x + \alpha) = o_S)}$$
$$\leq \prod_{s=1}^S \exp(\epsilon_s) = e^{(\sum_{s=1}^S \epsilon_s)}$$

As a result, we have

$$P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)\big) \leq e^{(\sum_i \epsilon_i)} P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha)\big)$$

The sequential composition of the expected output is as:

$$\mathbb{E} f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x) = \int_0^1 P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x) > t\big) dt$$
$$\leq e^{(\sum_s \epsilon_s)} \int_0^1 P\big(f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha) > t\big) dt$$
$$= e^{(\sum_s \epsilon_s)} \mathbb{E} f(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha)$$

Lemma 5 does hold.

L. Proof of Theorem 5

Proof 8 $\forall \alpha \in l_p(1)$*, from Lemma 5, with probability* $\geq \eta$*, we have that*

$$\hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha) \geq \frac{\hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)}{e^{(\sum_{s=1}^s \epsilon_s)}} \geq \frac{\hat{\mathbb{E}}_{lb} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)}{e^{(\sum_{s=1}^S \epsilon_s)}} \tag{32}$$

In addition, we also have

$$\forall i \neq k : \hat{\mathbb{E}} f_{i:i \neq k}(\mathcal{M}_1, \ldots, \mathcal{M}_S|x + \alpha) \leq e^{(\sum_{s=1}^S \epsilon_s)} \hat{\mathbb{E}} f_{i:i \neq k}(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)$$

$$\Rightarrow \forall i \neq k : \hat{\mathbb{E}} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha) \leq e^{(\sum_{s=1}^{S} \epsilon_s)} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x) \tag{33}$$

Using the hypothesis (Eq. 12) and the first inequality (Eq. 32), we have that

$$\begin{aligned}\hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha) &> \frac{e^{2(\sum_{s=1}^{S} \epsilon_s)} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)}{e^{(\sum_{s=1}^{S} \epsilon_s)}} \\ &> e^{(\sum_{s=1}^{S} \epsilon_s)} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x)\end{aligned}$$

Now, we apply the third inequality (Eq. 33), we have that

$$\begin{aligned}&\forall i \neq k : \hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha) > \hat{\mathbb{E}} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha) \\ &\Leftrightarrow \hat{\mathbb{E}} f_k(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha) > \max_{i:i\neq k} \hat{\mathbb{E}} f_i(\mathcal{M}_1, \ldots, \mathcal{M}_S|x+\alpha)\end{aligned}$$

The Theorem 5 does hold.

M. Proof of Corollary 1

Proof 9 $\forall \alpha \in l_p(1)$*, by applying Theorem 5, we have*

$$\begin{aligned}\hat{\mathbb{E}}_{lb} f_k(\mathcal{M}_h, \mathcal{M}_x|x) &> e^{2(\frac{\epsilon_r}{\kappa} + \frac{\epsilon_r}{\varphi})} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_h, \mathcal{M}_x|x) \\ &> e^{2(\frac{\kappa+\varphi}{\kappa\varphi})\epsilon_r} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_h, \mathcal{M}_x|x) = e^{2(\epsilon_r/\frac{\kappa\varphi}{\kappa+\varphi})} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_h, \mathcal{M}_x|x)\end{aligned}$$

Furthermore, by applying group privacy, we have that

$$\forall \alpha \in l_p(\frac{\kappa\varphi}{\kappa+\varphi}) : \hat{\mathbb{E}}_{lb} f_k(\mathcal{M}_h, \mathcal{M}_x|x) > e^{2\epsilon_r} \max_{i:i\neq k} \hat{\mathbb{E}}_{ub} f_i(\mathcal{M}_h, \mathcal{M}_x|x) \tag{34}$$

By applying Proof 8, it is straight to have

$$\forall \alpha \in l_p(\frac{\kappa\varphi}{\kappa+\varphi}) : \hat{\mathbb{E}} f_k(\mathcal{M}_h, \mathcal{M}_x|x+\alpha) > \max_{i:i\neq k} \hat{\mathbb{E}} f_k(\mathcal{M}_h, \mathcal{M}_x|x+\alpha)$$

with probability $\geq \eta$*. Corollary 1 does hold.*

N. Effective Monte Carlo Estimation of $\hat{\mathbb{E}} f(x)$

Recall that the Monte Carlo estimation is applied to estimate the expected value $\hat{\mathbb{E}} f(x) = \frac{1}{n}\sum_n f(x)_n$, where n is the number of invocations of $f(x)$ with independent draws in the noise, i.e., $\frac{1}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ and $\frac{2}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ in our case. When ϵ_1 is small (indicating a strong privacy protection), it causes a *notably large distribution shift between training and inference, given independent draws of the Laplace noise*.

In fact, let us denote a single draw in the noise as $\chi_1 = \frac{1}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ used to train the function $f(x)$, the model converges to the point that the noise χ_1 and $2\chi_2$ need to be correspondingly added into x and h in order to make correct predictions. χ_1 can be approximated as $Lap(\chi_1, \varrho)$, where $\varrho \to 0$. It is clear that independent draws of the noise $\frac{1}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1})$ have *distribution shifts* with the fixed noise $\chi_1 \approxeq Lap(\chi_1, \varrho)$. These distribution shifts can also be large, when noise is large. We have experienced that these distribution shifts in having independent draws of noise to estimate $\hat{\mathbb{E}} f(x)$ can notably degrade the inference accuracy of the scoring function, when privacy budget ϵ_1 is small resulting in a large amount of noise injected to provide strong privacy guarantees.

To address this, one solution is to increase the number of invocations of $f(x)$, i.e., n, to a huge number per prediction. However, this is impractical in real-world scenarios. We propose a novel way to draw independent noise following the distribution of $\chi_1 + \frac{1}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1}/\psi)$ for the input x and $2\chi_2 + \frac{2}{m} Lap(0, \frac{\Delta_{\mathcal{R}}}{\epsilon_1}/\psi)$ for the affine transformation h, where ψ is a hyper-parameter to control the distribution shifts. This approach works well and does not affect the DP bounds and the certified robustness condition, since: **(1)** Our mechanism achieves both DP and certified robustness in the

training process; and **(2)** It is clear that $\hat{\mathbb{E}}f(x) = \frac{1}{n}\sum_n f(x)_n = \frac{1}{n}\sum_n g\big(a(x+\chi_1+\frac{1}{m}Lap_n(0,\frac{\Delta_{\mathcal{R}}}{\epsilon_1}/\psi),\theta_1)+2\chi_2+\frac{2}{m}Lap_n(0,\frac{\Delta_{\mathcal{R}}}{\epsilon_1}/\psi),\theta_2\big)$, where $Lap_n(0,\frac{\Delta_{\mathcal{R}}}{\epsilon_1}/\psi)$ is the n-th draw of the noise. When $n\to\infty$, $\hat{\mathbb{E}}f(x)$ will converge to $\frac{1}{n}\sum_n g\big(a(x+\chi_1,\theta_1)+2\chi_2,\theta_2\big)$, which aligns well with the convergence point of the scoring function $f(x)$. Injecting χ_1 and $2\chi_2$ to x and h during the estimation of $\hat{\mathbb{E}}f(x)$ yields better performance, without affecting the DP and the composition robustness bounds.

O. Approximation Error Bounds

To compute how much error our polynomial approximation approaches (i.e., truncated Taylor expansions), $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$ (Eq. 5) and $\mathcal{L}_{\overline{B}_t}\big(\theta_2\big)$, incur, we directly apply Lemma 4 in (Phan et al., 2016), Lemma 3 in (Zhang et al., 2012), and the well-known error bound results in (Apostol, 1967). Note that $\widetilde{\mathcal{R}}_{B_t}(\theta_1)$ is the 1st-order Taylor series and $\mathcal{L}_{\overline{B}_t}\big(\theta_2\big)$ is the 2nd-order Taylor series following the implementation of (TensorFlow). Let us closely follow (Phan et al., 2016; Zhang et al., 2012; Apostol, 1967) to adapt their results into our scenario, as follows:

Given the truncated function $\widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\sum_{l=1}^{2}\sum_{r=0}^{1}\frac{\mathbf{F}_{lj}^{(r)}(0)}{r!}\big(\theta_{1j}h_i\big)^r$, the original Taylor polynomial function $\widehat{\mathcal{R}}_{B_t}(\theta_1) = \sum_{x_i\in B_t}\sum_{j=1}^{d}\sum_{l=1}^{\infty}\sum_{r=0}^{1}\frac{\mathbf{F}_{lj}^{(r)}(0)}{r!}\big(\theta_{1j}h_i\big)^r$, the average error of the approximation is bounded as

$$\frac{1}{|B_t|}|\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widehat{\mathcal{R}}_{B_t}(\widehat{\theta}_1)| \le \frac{4e\times d}{(1+e)^2} \tag{35}$$

$$\frac{1}{|B_t|}|\widehat{\mathcal{L}}_{B_t}(\widetilde{\theta}_2)-\widehat{\mathcal{L}}_{B_t}(\widehat{\theta}_2)| \le \frac{e^2+2e-1}{e(1+e)^2}\times K \tag{36}$$

where $\widehat{\theta}_1 = \arg\min_{\theta_1}\widehat{\mathcal{R}}_{B_t}(\theta_1)$, $\widetilde{\theta}_1 = \arg\min_{\theta_1}\widetilde{\mathcal{R}}_{B_t}(\theta_1)$, $\widehat{\mathcal{L}}_{B_t}(\theta_2)$ is the original Taylor polynomial function of $\sum_{x_i\in B_t}\mathcal{L}\big(f(\overline{x}_i,\theta_2),y_i\big)$, $\widehat{\theta}_2 = \arg\min_{\theta_2}\widehat{\mathcal{L}}_{B_t}(\theta_2)$, $\widetilde{\theta}_2 = \arg\min_{\theta_2}\mathcal{L}_{B_t}(\theta_2)$.

Proof 10 *Let* $U = \max_{\theta_1}\big(\widehat{\mathcal{R}}_{B_t}(\theta_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1)\big)$ *and* $S = \min_{\theta_1}\big(\widehat{\mathcal{R}}_{B_t}(\theta_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1)\big)$.

We have that $U \ge \widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widetilde{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)$ *and* $\forall\theta_1^*: S \le \widehat{\mathcal{R}}_{B_t}(\theta_1^*)-\widetilde{\mathcal{R}}_{B_t}(\theta_1^*)$. *Therefore, we have*

$$\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widetilde{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widehat{\mathcal{R}}_{B_t}(\theta_1^*)+\widetilde{\mathcal{R}}_{B_t}(\theta_1^*) \le U-S \tag{37}$$

$$\Leftrightarrow\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widehat{\mathcal{R}}_{B_t}(\theta_1^*) \le U-S+\big(\widetilde{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1^*)\big) \tag{38}$$

In addition, $\widetilde{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1^*)\le 0$, *it is straightforward to have:*

$$\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widehat{\mathcal{R}}_{B_t}(\theta_1^*) \le U-S \tag{39}$$

If $U\ge 0$ *and* $S\le 0$ *then we have:*

$$|\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1)-\widehat{\mathcal{R}}_{B_t}(\theta_1^*)| \le U-S \tag{40}$$

Eq. 40 holds for every θ_1^*, *including* $\widehat{\theta}_1$. *Eq. 40 shows that the error incurred by truncating the Taylor series approximate function depends on the maximum and minimum values of* $\widehat{\mathcal{R}}_{B_t}(\theta_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1)$. *This is consistent with (Phan et al., 2016; Zhang et al., 2012). To quantify the magnitude of the error, we rewrite* $\widehat{\mathcal{R}}_{B_t}(\theta_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1)$ *as:*

$$\widehat{\mathcal{R}}_{B_t}(\theta_1)-\widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{j=1}^{d}\big(\widehat{\mathcal{R}}_{B_t}(\theta_{1j})-\widetilde{\mathcal{R}}_{B_t}(\theta_{1j})\big) \tag{41}$$

$$= \sum_{j=1}^{d}\Big(\sum_{i=1}^{|B_t|}\sum_{l=1}^{2}\sum_{r=3}^{\infty}\frac{\mathbf{F}_{lj}^{(r)}(z_{lj})}{r!}\big(g_{lj}(x_i,\theta_{1j})-z_{lj}\big)^r\Big) \tag{42}$$

where $g_{1j}(x_i,\theta_{1j}) = \theta_{1j}h_i$ *and* $g_{2j}(x_i,\theta_{1j}) = \theta_{1j}h_i$.

By looking into the remainder of Taylor expansion for each j *(i.e., following (Phan et al., 2016; Apostol, 1967)), with* $z_j\in[z_{lj}-1, z_{lj}+1]$, $\frac{1}{|B_t|}\big(\widehat{\mathcal{R}}_{B_t}(\theta_{1j})-\widetilde{\mathcal{R}}_{B_t}(\theta_{1j})\big)$ *must be in the interval* $\Big[\sum_l\frac{\min_{z_j}\mathbf{F}_{lj}^{(2)}(z_j)(z_j-z_{lj})^2}{2!},\sum_l\frac{\max_{z_j}\mathbf{F}_{lj}^{(2)}(z_j)(z_j-z_{lj})^2}{2!}\Big]$.

If $\sum_l \frac{\max_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2}{2!} \geq 0$ *and* $\sum_l \frac{\min_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2}{2!} \leq 0$*, then we have that* $|\frac{1}{|B_t|}(\widehat{\mathcal{R}}_{B_t}(\theta_1) - \widetilde{\mathcal{R}}_{B_t}(\theta_1))| \leq \sum_{j=1}^{d} \sum_l \frac{\max_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2 - \min_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2}{2!}$*. This can be applied to the case of our auto-encoder, as follows:*

For the functions $\mathbf{F}_{1j}(z_j) = x_{ij} \log(1 + e^{-z_j})$ *and* $\mathbf{F}_{2j}(z_j) = (1 - x_{ij}) \log(1 + e^{z_j})$*, we have* $\mathbf{F}_{1j}^{(2)}(z_j) = \frac{x_{ij} e^{-z_j}}{(1+e^{-z_j})^2}$ *and* $\mathbf{F}_{2j}^{(2)}(z_j) = (1 - x_{ij}) \frac{e^{z_j}}{(1+e^{z_j})^2}$*. It can be verified that* $\arg\min_{z_j} \mathbf{F}_{1j}^{(2)}(z_j) = \frac{-e}{(1+e)^2} < 0$, $\arg\max_{z_j} \mathbf{F}_{1j}^{(2)}(z_j) = \frac{e}{(1+e)^2} > 0$, $\arg\min_{z_j} \mathbf{F}_{2j}^{(2)}(z_j) = 0$*, and* $\arg\max_{z_j} \mathbf{F}_{2j}^{(2)}(z_j) = \frac{2e}{(1+e)^2} > 0$*. Thus, the average error of the approximation is at most:*

$$\frac{1}{|B_t|} |\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1) - \widehat{\mathcal{R}}_{B_t}(\widehat{\theta}_1)| \leq \Big[\big(\frac{e}{(1+e)^2} - \frac{-e}{(1+e)^2} \big) + \frac{2e}{(1+e)^2} \Big] \times d = \frac{4e \times d}{(1+e)^2} \tag{43}$$

Consequently, Eq. 35 does hold. Similarly, by looking into the remainder of Taylor expansion for each label k*, Eq. 36 can be proved straightforwardly. In fact, by using the 2nd-order Taylor series with* K *categories, we have that:* $\frac{1}{|B_t|} |\widehat{\mathcal{L}}_{B_t}(\widetilde{\theta}_2) - \widehat{\mathcal{L}}_{B_t}(\widehat{\theta}_2)| \leq \frac{e^2 + 2e - 1}{e(1+e)^2} \times K$.

P. Model Configurations

The MNIST database consists of handwritten digits (Lecun et al., 1998). Each example is a 28×28 size gray-level image. The CIFAR-10 dataset consists of color images belonging to 10 classes, i.e., airplanes, dogs, etc. The dataset is split into 50,000 training samples and 10,000 test samples (Krizhevsky & Hinton, 2009). Tiny Imagenet $(64 \times 64 \times 3)$ has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. We used the first thirty classes with data augmented, including horizontal flip and random brightness, in the Tiny ImageNet dataset in our experiment. In general, the dataset is split into 45,000 training samples and 1,500 test samples (TinyImageNet; Hendrycks & Dietterich, 2019). The experiments were conducted on a server of 4 GPUs, each of which is an NVIDIA TITAN Xp, 12 GB with 3,840 CUDA cores. All the models share the same structure, consisting of 2 and 3 convolutional layers, respectively for MNIST and CIFAR-10 datasets, and a ResNet18 model for the Tiny ImageNet dataset.

Both fully-connected and convolution layers can be applied in the representation learning model $a(x, \theta_1)$. Given convolution layer, the computation of each feature map needs to be DP; since each of them independently reads a local region of input neurons. Therefore, the sensitivity $\Delta_{\mathcal{R}}$ can be considered the maximal sensitivity given any single feature map in the first affine transformation layer. In addition, each hidden neuron can only be used to reconstruct a unit patch of input units. That results in d (Lemma 2) being the size of the unit patch connected to each hidden neuron, e.g., $d = 9$ given a 3×3 unit patch, and β is the number of hidden neurons in a feature map.

MNIST: We used two convolutional layers (32 and 64 features). Each hidden neuron connects with a 5x5 unit patch. A fully-connected layer has 256 units. The batch size m was set to 2,499, $\xi = 1$, $\psi = 2$. I-FGSM, MIM, and MadryEtAl were used to draft $l_\infty(\mu)$ adversarial examples in training, with $T_\mu = 10$. Learning rate ϱ_t was set to $1e - 4$. Given a predefined total privacy budget ϵ, ϵ_2 is set to be 0.1, and ϵ_1 is computed as: $\epsilon_1 = \frac{\epsilon - \epsilon_2}{(1 + 1/\gamma + 1/\gamma_{\mathbf{x}})}$. This will guarantee that $(\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2) = \epsilon$. $\Delta_{\mathcal{R}} = (14^2 + 2) \times 25$ and $\Delta_{\mathcal{L}2} = 2 \times 256$. The number of Monte Carlo sampling for certified inference n is set to 2,000.

CIFAR-10: We used three convolutional layers (128, 128, and 256 features). Each hidden neuron connects with a 4x4 unit patch in the first layer, and a 5x5 unit patch in other layers. One fully-connected layer has 256 neurons. The batch size m was set to 1,851, $\xi = 1.5$, $\psi = 10$, and $T_\mu = 3$. The ensemble of attacks A includes I-FGSM, MIM, and MadryEtAl. We use data augmentation, including random crop, random flip, and random contrast. Learning rate ϱ_t was set to $5e - 2$. In the CIFAR-10 dataset, ϵ_2 is set to $(1 + r/3.0)$ and $\epsilon_1 = (1 + 2r/3.0)/(1 + 1/\gamma + 1/\gamma_{\mathbf{x}})$, where $r \geq 0$ is a ratio to control the total privacy budget ϵ in our experiment. For instance, given $r = 0$, we have that $\epsilon = (\epsilon_1 + \epsilon_1/\gamma_{\mathbf{x}} + \epsilon_1/\gamma + \epsilon_2) = 2$. $\Delta_{\mathcal{R}} = 3 \times (14^2 + 2) \times 16$ and $\Delta_{\mathcal{L}2} = 2 \times 256$. $\mathbb{N}$ and $\mathbb{M}$ are set to 1 and 4 in the distributed training. The number of Monte Carlo sampling for certified inference n is set to 1,000.

Tiny ImageNet: We used a ResNet-18 model. Each hidden neuron connects with a 7x7 unit patch in the first layer, and 3x3 unit patch in other layers. The batch size m was set to 4,500, $\xi = 1.5$, $\psi = 10$, and $T_\mu = 10$. The ensemble of attacks A includes I-FGSM, MIM, and MadryEtAl. Learning rate ϱ_t was set to $1e - 2$. In the Tiny ImageNet dataset, ϵ_2 is set to 1 and $\epsilon_1 = (1 + r)/(1 + 1/\gamma + 1/\gamma_{\mathbf{x}})$, where $r \geq 0$ is a ratio to control the total privacy budget ϵ in our experiment. $\Delta_{\mathcal{R}} = 3 \times (32^2 + 2) \times 49$ and $\Delta_{\mathcal{L}2} = 2 \times 256$. $\mathbb{N}$ and $\mathbb{M}$ are set to 1 and 20 in the distributed training. The number of Monte Carlo sampling for certified inference n is set to 1,000.

Q. Complete and Detailed Experimental Results

Results on the MNIST Dataset. Figure 2 illustrates the conventional accuracy of each model as a function of the privacy budget ϵ on the MNIST dataset under $l_\infty(\mu_a)$-norm attacks, with $\mu_a = 0.2$ (a pretty strong attack). It is clear that our StoBatch outperforms AdLM, DP-SGD, SecureSGD, and SecureSGD-AGM, in all cases, with $p < 1.32e-4$. On average, we register a 22.36% improvement over SecureSGD ($p < 1.32e-4$), a 46.84% improvement over SecureSGD-AGM ($p < 1.83e-6$), a 56.21% improvement over AdLM ($p < 2.05e-10$), and a 77.26% improvement over DP-SGD ($p < 5.20e-14$), given our StoBatch mechanism. AdLM and DP-SGD achieve the worst conventional accuracies. There is no guarantee provided in AdLM and DP-SGD. Thus, the accuracy of the AdLM and DPSGD algorithms seem to show no effect against adversarial examples, when the privacy budget is varied. This is in contrast to our StoBatch model, the SecureSGD model, and the SecureSGD-AGM model, whose accuracies are proportional to the privacy budget.

When the privacy budget $\epsilon = 0.2$ (a tight DP protection), there are significant drops, in terms of conventional accuracy, given the baseline approaches. By contrast, our StoBatch mechanism only shows a small degradation in the conventional accuracy (6.89%, from 89.59% to 82.7%), compared with a 37% drop in SecureSGD (from 78.64% to 41.64%), and a 32.89% drop in SecureSGD-AGM (from 44.1% to 11.2%) on average, when the privacy budget ϵ goes from 2.0 to 0.2. At $\epsilon = 0.2$, our StoBatch mechanism achieves 82.7%, compared with 11.2% and 41.64% correspondingly for SecureSGD-AGM and SecureSGD. This is an important result, showing the ability to offer tight DP protections under adversarial example attacks in our model, compared with existing algorithms.

• Figure 4 presents the conventional accuracy of each model as a function of the attack size μ_a on the MNIST dataset, under a strong DP guarantee, $\epsilon = 0.2$. Our StoBatch mechanism outperforms the baseline approaches in all cases. On average, our StoBatch model improves 44.91% over SecureSGD ($p < 7.43e-31$), a 61.13% over SecureSGD-AGM ($p < 2.56e-22$), a 52.21% over AdLM ($p < 2.81e-23$), and a 62.20% over DP-SGD ($p < 2.57e-22$). More importantly, our StoBatch model is resistant to different adversarial example algorithms with different attack sizes. When $\mu_a \geq 0.2$, AdLM, DP-SGD, SecureSGD, and SecureSGD-AGM become defenseless. We further register significantly drops in terms of accuracy, when μ_a is increased from 0.05 (a weak attack) to 0.6 (a strong attack), i.e., 19.87% on average given our StoBatch, across all attacks, compared with 27.76% (AdLM), 29.79% (DP-SGD), 34.14% (SecureSGD-AGM), and 17.07% (SecureSGD).

• Figure 6 demonstrates the certified accuracy as a function of μ_a. The privacy budget is set to 1.0, offering a reasonable privacy protection. In PixelDP, the construction attack bound ϵ_r is set to 0.1, which is a pretty reasonable defense. With (small perturbation) $\mu_a \leq 0.2$, PixelDP achieves better certified accuracies under all attacks; since PixelDP does not preserve DP to protect the training data, compared with other models. Meanwhile, our StoBatch model outperforms all the other models when $\mu_a \geq 0.3$, indicating a stronger defense to more aggressive attacks. More importantly, our StoBatch has a consistent certified accuracy to different attacks given different attack sizes, compared with baseline approaches. In fact, when μ_a is increased from 0.05 to 0.6, our StoBatch shows a small drop (11.88% on average, from $84.29\%(\mu_a = 0.05)$ to $72.41\%(\mu_a = 0.6)$), compared with a huge drop of the PixelDP, i.e., from $94.19\%(\mu_a = 0.05)$ to $9.08\%(\mu_a = 0.6)$ on average under I-FGSM, MIM, and MadryEtAl attacks, and to $77.47\%(\mu_a = 0.6)$ under FGSM attack. Similarly, we also register significant drops in terms of certified accuracy for SecureSGD (78.74%, from 86.74% to 7.99%) and SecureSGD-AGM (81.97%, from 87.23% to 5.26%) on average. This is promising.

Results on the CIFAR-10 Dataset further strengthen our observations. In Figure 3, our StoBatch clearly outperforms baseline models in all cases ($p < 6.17e-9$), especially when the privacy budget is small ($\epsilon < 4$), yielding strong privacy protections. On average conventional accuracy, our StoBatch mechanism has an improvement of 10.42% over SecureSGD ($p < 2.59e-7$), an improvement of 14.08% over SecureSGD-AGM ($p < 5.03e-9$), an improvement of 29.22% over AdLM ($p < 5.28e-26$), and a 14.62% improvement over DP-SGD ($p < 4.31e-9$). When the privacy budget is increased from 2 to 10, the conventional accuracy of our StoBatch model increases from 42.02% to 46.76%, showing a 4.74% improvement on average. However, the conventional accuracy of our model under adversarial example attacks is still low, i.e., 44.22% on average given the privacy budget at 2.0. This opens a long-term research avenue to achieve better robustness under strong privacy guarantees in adversarial learning.

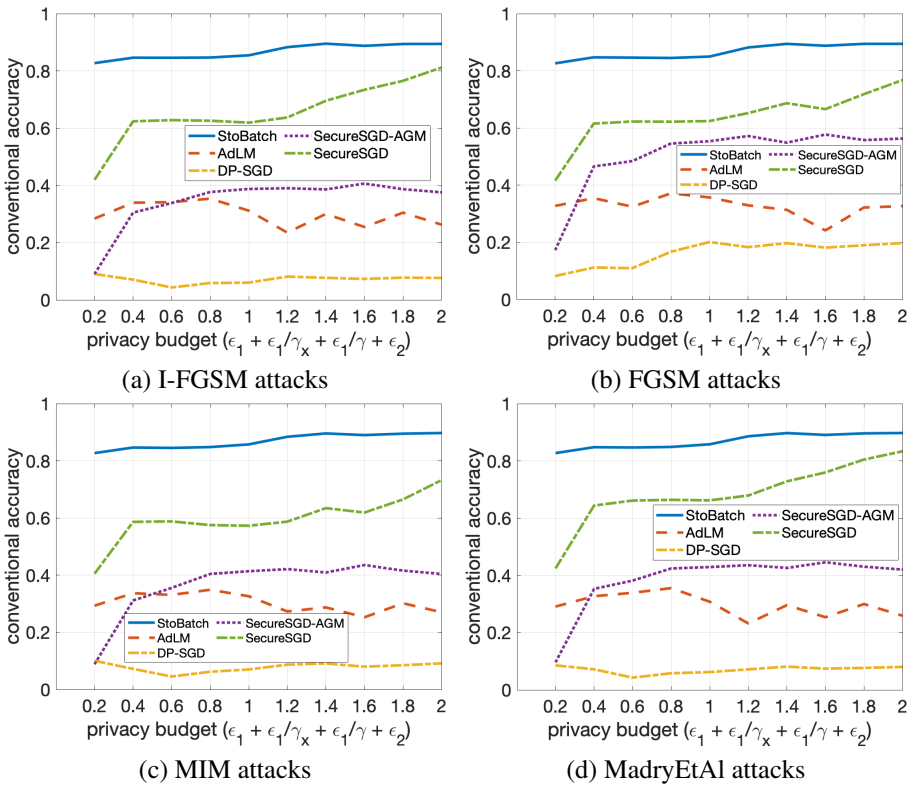
• The accuracy of our model is consistent given different attacks with different adversarial perturbations μ_a under a rigorous DP protection ($\epsilon = 2.0$), compared with baseline approaches (Figure 5). In fact, when the attack size μ_a increases from 0.05 to 0.5, the conventional accuracies of the baseline approaches are remarkably reduced, i.e., a drop of 25.26% on average given the most effective baseline approach, SecureSGD. Meanwhile, there is a much smaller degradation (4.79% on average) in terms of the conventional accuracy observed in our StoBatch model. Our model also achieves better accuracies compared with baseline approaches in all cases ($p < 8.2e-10$). Figure 7 further shows that our StoBatch model is more accurate than

baseline approaches (i.e., ϵ_r is set to 0.1 in PixelDP) in terms of certified accuracy in all cases, with a tight privacy budget set to 2.0 ($p < 2.04e-18$). We register an improvement of 21.01% in our StoBatch model given the certified accuracy over SecureSGD model, which is the most effective baseline approach ($p < 2.04e-18$).

Scalability under Strong Iterative Attacks. First, we scale our model in terms of *adversarial training* in the CIFAR-10 dataset, in which the number of iterative attack steps is increased from $T_\mu = 3$ to T_μ = *200 in training*, and up to T_a = *2,000 in testing*. Note that the traditional iterative batch-by-batch DP adversarial training (Alg. 1) is nearly infeasible in this setting, taking over 30 days for one training with 600 epochs. Thanks to the parallel and distributed training, our StoBatch only takes $\approxeq$ 3 days to finish the training. More importantly, our StoBatch achieves consistent conventional and certified accuracies under strong iterative attacks with $T_a = 1,000$, compared with the best baseline, i.e., SecureSGD (Figure 8). Across attack sizes $\mu_a \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and steps $T_a \in \{100, 500, 1000, 2000\}$, on average, our StoBatch achieves 44.87±1.8% and 42.18±1.8% in conventional and certified accuracies, compared with 29.47±12.5% and 20±6.1% of SecureSGD ($p < 1.05e-9$).

• We achieve a similar improvement over the **Tiny ImageNet**, i.e., following (Hendrycks & Dietterich, 2019), with a ResNet18 model, i.e., *a larger dataset on a larger network* (Figure 9). On average, across attack sizes $\mu_a \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and steps $T_a \in \{100, 500, 1000, 2000\}$, our StoBatch achieves 29.78±4.8% and 28.31±1.58% in conventional and certified accuracies, compared with 8.99±5.95% and 8.72±5.5% of SecureSGD ($p < 1.55e-42$).

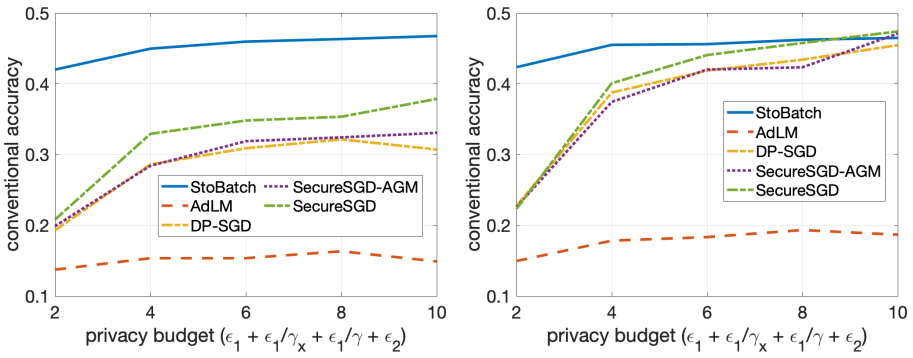
Key observations: (1) Incorporating ensemble adversarial learning into DP preservation, tightened sensitivity bounds, a random perturbation size μ_t at each training step, and composition robustness bounds in both input and latent spaces does enhance the consistency, robustness, and accuracy of DP model against different attacks with different levels of perturbations. These are key advantages of our mechanism; **(2)** As a result, our StoBatch model outperforms baseline algorithms, in terms of conventional and certified accuracies in most of the cases. It is clear that existing DP-preserving approaches have not been designed to withstand against adversarial examples; and **(3)** Our StoBatch training can help us to scale our mechanism to larger DP DNNs and datasets with distributed adversarial learning, without affecting the model accuracies and DP protections.



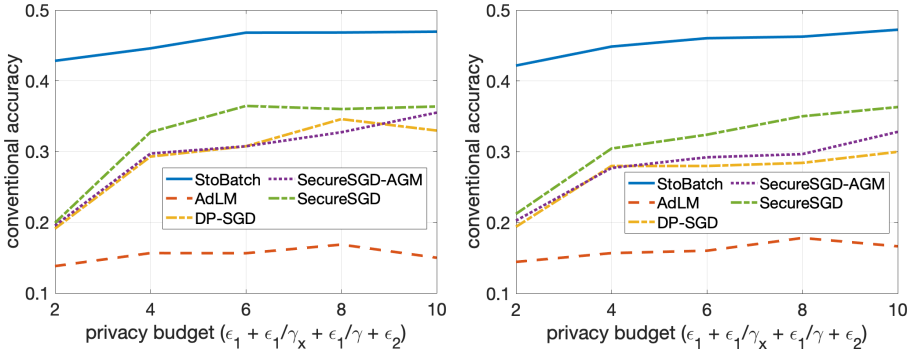
(a) I-FGSM attacks (b) FGSM attacks

(c) MIM attacks (d) MadryEtAl attacks

Figure 2. Conventional accuracy on the MNIST dataset given ϵ, under $l_\infty(\mu_a = 0.2)$ and $T_a = 10$.

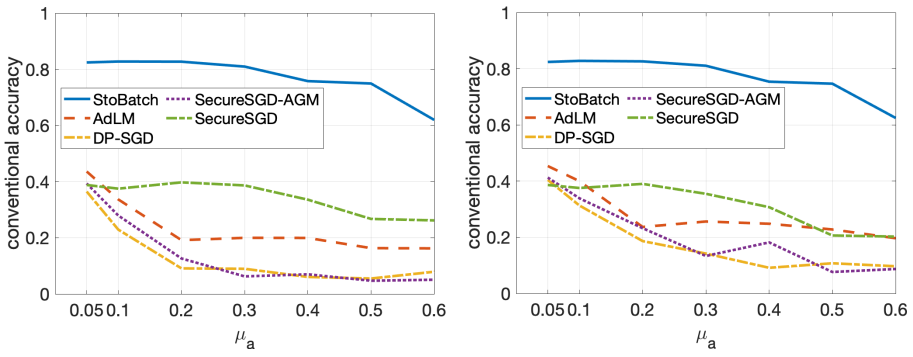


(a) I-FGSM attacks (b) FGSM attacks

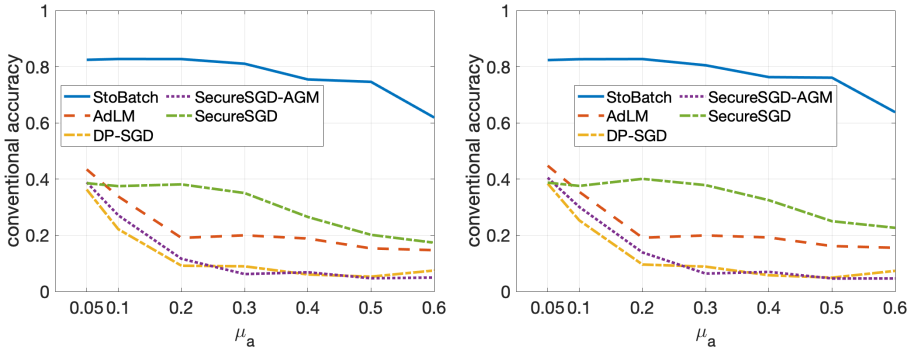


(c) MIM attacks (d) MadryEtAl attacks

Figure 3. Conventional accuracy on the CIFAR-10 dataset given ϵ, under $l_\infty(\mu_a = 0.2)$ and $T_a = 3$.

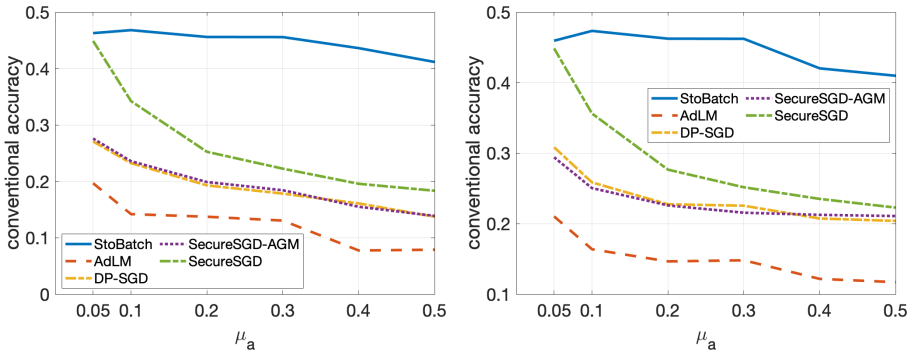


(a) I-FGSM attacks (b) FGSM attacks

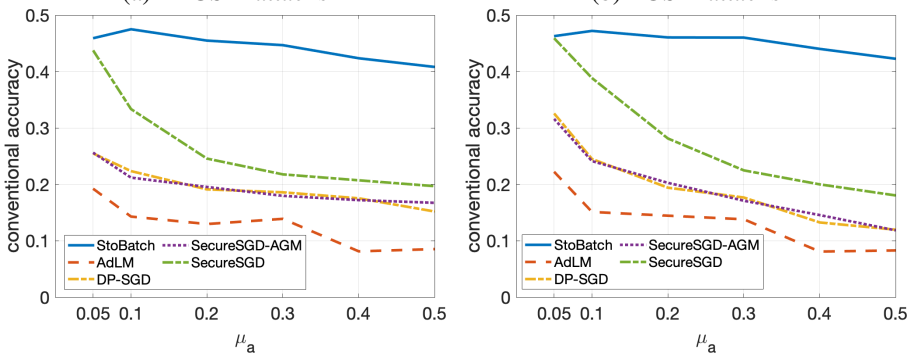


(c) MIM attacks (d) MadryEtAl attacks

Figure 4. Conventional accuracy on the MNIST dataset given μ_a ($\epsilon = 0.2$, tight DP protection) and $T_a = 10$.

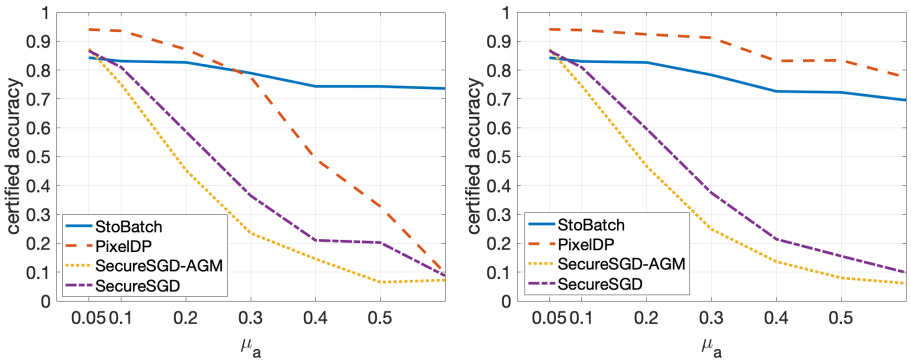


(a) I-FGSM attacks (b) FGSM attacks

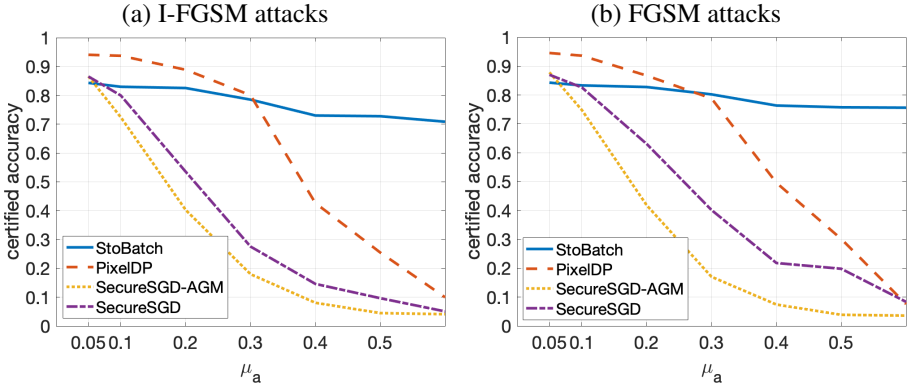


(c) MIM attacks (d) MadryEtAl attacks

Figure 5. Conventional accuracy on the CIFAR-10 dataset given μ_a ($\epsilon = 2$, tight DP protection) and $T_a = 3$.

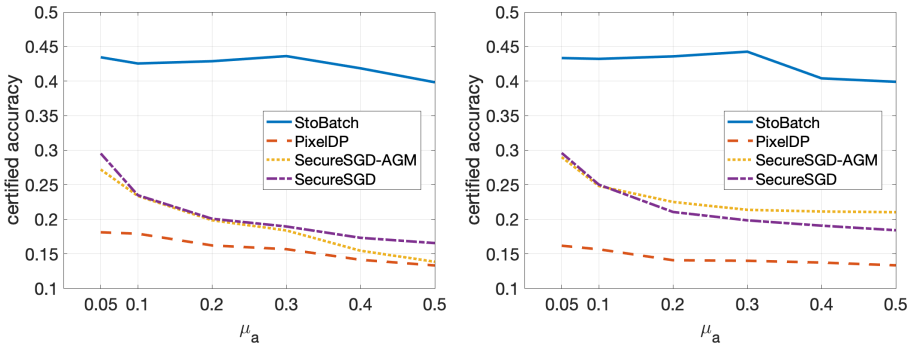


(a) I-FGSM attacks (b) FGSM attacks

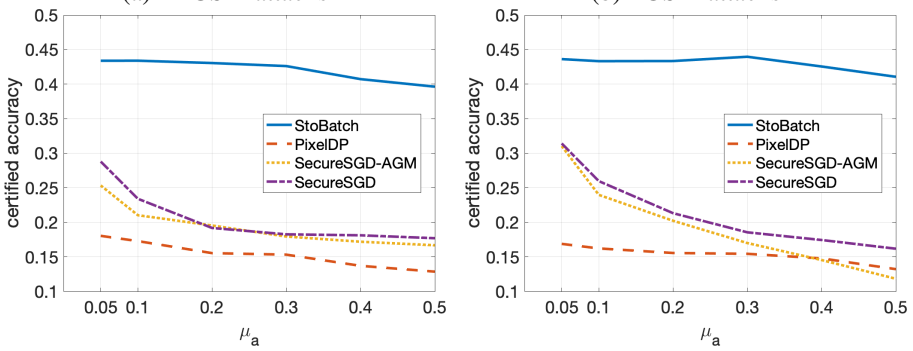


(c) MIM attacks (d) MadryEtAl attacks

Figure 6. Certified accuracy on the MNIST dataset. ϵ is set to 1.0 (tight DP protection) and $T_a = 10$.

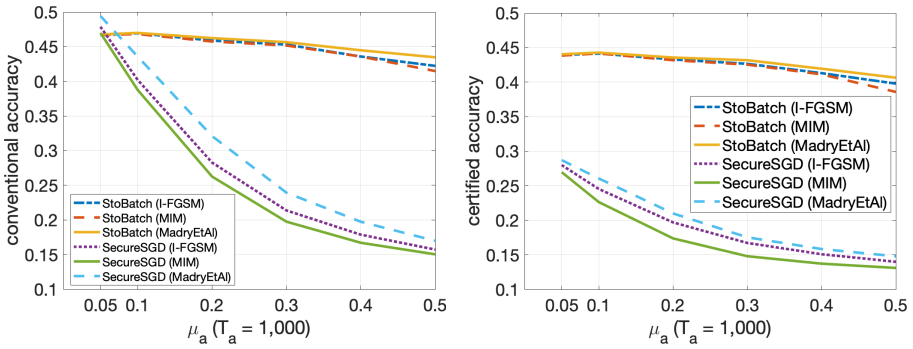


(a) I-FGSM attacks (b) FGSM attacks

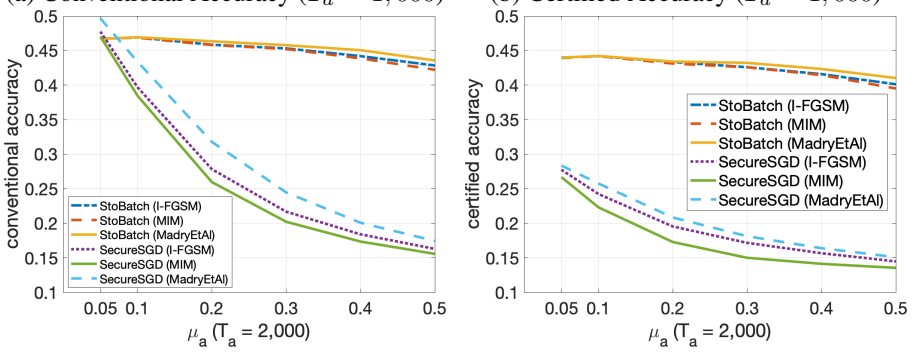


(c) MIM attacks (d) MadryEtAl attacks

Figure 7. Certified accuracy on the CIFAR-10 dataset. ϵ is set to 2 (tight DP protection) and and $T_a = 3$.

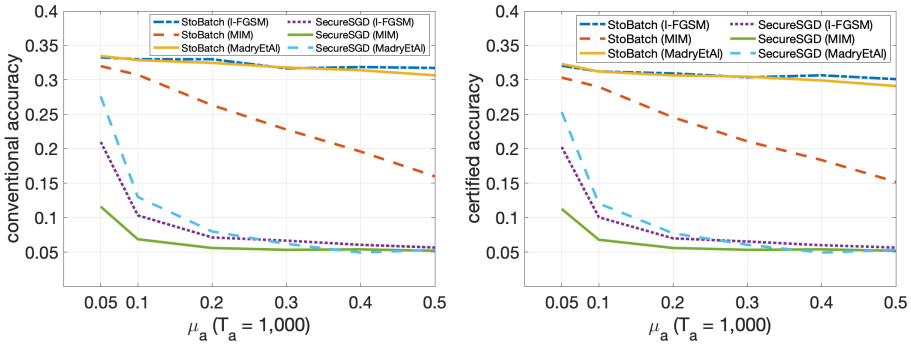


(a) Conventional Accuracy ($T_a = 1,000$) (b) Certified Accuracy ($T_a = 1,000$)

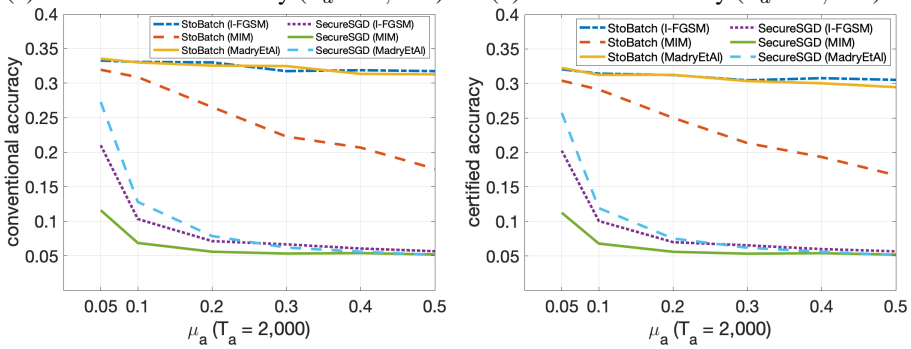


(c) Conventional Accuracy ($T_a = 2,000$) (d) Certified Accuracy ($T_a = 2,000$)

Figure 8. Accuracy on the CIFAR-10 dataset, under Strong Iterative Attacks ($T_a = 1,000; 2,000$). ϵ is set to 2 (tight DP protection).



(a) Conventional Accuracy ($T_a = 1,000$) (b) Certified Accuracy ($T_a = 1,000$)



(c) Conventional Accuracy ($T_a = 2,000$) (d) Certified Accuracy ($T_a = 2,000$)

Figure 9. Accuracy on the Tiny ImageNet dataset, under Strong Iterative Attacks ($T_a = 1,000; 2,000$). ϵ is set to 5.