## A. Notations and Terminologies

*Table 1.* Notations and Terminologies.

| | |
|---|---|
| $D$ and $x$ | Training data with benign examples $x \in [-1, 1]^d$ |
| $y = \{y_1, \dots, y_K\}$ | 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 $f(x) = \{f_1(x), \dots, f_K(x)\}$ |
| $y_x \in y$ | 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 $\alpha$ is the perturbation |
| $l_p(\mu) = \{\alpha \in \mathbb{R}^d : \|\alpha\|_p \leq \mu\}$ | The $l_p$-norm ball of attack radius $\mu$ |
| $(\epsilon_r, \delta_r)$ | Robustness budget $\epsilon_r$ and broken probability $\delta_r$ |
| $\mathbb{E} f_k(x)$ | 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 $\theta_1$ |
| $B_t$ | 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 $\theta_2$ |
| $\Upsilon\big(f(\overline{x}_j^{\text{adv}}, \theta_2), y_j\big)$ | Loss function of DP adversarial examples $\overline{x}_j^{\text{adv}}$, given $\theta_2$ |
| $\overline{\mathcal{L}}_{\overline{B}_t}(\theta_2)$ | 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 |
| $f(\mathcal{M}_1, \dots, \mathcal{M}_s \| x)$ | Composition scoring function given independent randomizing mechanisms $\mathcal{M}_1, \dots, \mathcal{M}_s$ |
| $\Delta_r^x$ and $\Delta_r^h$ | 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 $D$ |
| $(\kappa + \varphi)_{max}$ | 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 $\epsilon$-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 $\theta$, given a private training dataset $D$. The mechanism exploits the polynomial representation of $L_D(\theta)$. The model parameter $\theta$ is a vector that contains $\mathbf{d}$ values $\theta_1, \dots, \theta_{\mathbf{d}}$. Let $\phi(\theta)$ denote a product of $\theta_1, \dots, \theta_{\mathbf{d}}$, namely, $\phi(\theta) = \theta_1^{c_1} \cdot \theta_2^{c_2} \cdots \theta_{\mathbf{d}}^{c_{\mathbf{d}}}$ for some $c_1, \dots, c_{\mathbf{d}} \in \mathbb{N}$. Let $\Phi_j (j \in \mathbb{N})$ denote the set of all products of $\theta_1, \dots, \theta_{\mathbf{d}}$ with degree $j$, i.e., $\Phi_j = \big\{\theta_1^{c_1} \cdot \theta_2^{c_2} \cdots \theta_{\mathbf{d}}^{c_{\mathbf{d}}} \big| \sum_{a=1}^{\mathbf{d}} c_a = j\big\}$. By the Stone-Weierstrass Theorem (Rudin, 1976), any continuous and differentiable $L(x_i, \theta)$ can always be written as a polynomial of $\theta_1, \dots, \theta_{\mathbf{d}}$, 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: $L(x_i, \theta) = (y_i - x_i^\top \theta)^2 = y_i^2 - \sum_{j=1}^{d} (2 y_i x_{ij}) \theta_j + \sum_{1 \leq j, a \leq d} (x_{ij} x_{ia}) \theta_j \theta_a$, where $d$ $(= \mathbf{d})$ is the number of features in $x_i$. In fact, $L(x_i, \theta)$ only involves monomials in $\Phi_0 = \{1\}, \Phi_1 = \{\theta_1, \dots, \theta_d\}$, and $\Phi_2 = \{\theta_i \theta_a \big| i, a \in [1, d]\}$. Each $\phi(\theta)$ has its own coefficient, e.g., for $\theta_j$, 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 $\epsilon$-DP, $L_D(\theta)$ is perturbed by injecting Laplace noise $Lap(\frac{\Delta}{\epsilon})$ into its polynomial coefficients $\lambda_\phi$, 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 $\epsilon$-DP without accessing the original data, i.e., that may potentially incur additional privacy leakage, grid search-based approaches are applied to learn the $\epsilon$-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 $\mu$. 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 $\theta$ 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 $T_\mu$ small steps, each of which has a size of $\mu/T_\mu$.

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 $B_t^{\text{adv}}$ 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 $\theta$, batch sizes $m_1$ and $m_2$, learning rate $\varrho_t$, parameter $\xi$
1: Initialize $\theta$ 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 $\mu_a$)
1: Compute robustness size $(\kappa + \varphi)_{max}$ in Eq. 13 of $x$
2: if $(\kappa + \varphi)_{max} \geq \mu_a$ then
3:     Return $isRobust(x) = True$, label $k$, $(\kappa + \varphi)_{max}$
4: else
5:     Return $isRobust(x) = False$, label $k$, $(\kappa + \varphi)_{max}$
```

**Algorithm 4 StoBatch Training**

**Input:** Database $D$, loss function $L$, parameters $\theta$, batch size $m$, learning rate $\varrho_t$, privacy budgets: $\epsilon_1$ and $\epsilon_2$, robustness parameters: $\epsilon_r$, $\Delta_r^x$, and $\Delta_r^h$, adversarial attack size $\mu_a$, the number of invocations $n$, ensemble attacks $A$, parameters $\psi$ and $\xi$, the size $|\mathbf{h}_\pi|$ of $\mathbf{h}_\pi$, a number of $\mathbb{N}$ random local trainers ($\mathbb{N} \leq N/(2m)$)

```
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: 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 $i \in [1, N/(2m)]$
5: for $t \in [T]$ do
6:   Randomly Pick $\mathbb{N}$ local trainers, each of which Gets the latest global parameters $\theta$ from the parameter server
7:   for $i \in [1, \mathbb{N}]$ do
8:     Assign $\overline{B}_{t,i} \leftarrow \overline{B}_{i1}$
9:     Ensemble DP Adversarial Examples:
10:    Draw Random Perturbation Value $\mu_t \in (0, 1]$, Assign $\overline{B}^{\text{adv}}_{t,i} \leftarrow \emptyset$
11:    for $l \in A$ do
12:      Take the next batch $\overline{B}_a \subset \overline{B}_{i2}$ with the size $m/|A|$
13:      $\forall \overline{x}_j \in \overline{B}_a$: Craft $\overline{x}^{\text{adv}}_j$ by using attack algorithm $A[l]$ with $l_\infty(\mu_t)$, $\overline{B}^{\text{adv}}_{t,i} \leftarrow \overline{B}^{\text{adv}}_{t,i} \cup \overline{x}^{\text{adv}}_j$
14:    Compute $\nabla_i\theta_1 \leftarrow \nabla_{\theta_1}\overline{\mathcal{R}}_{\overline{B}_{t,i} \cup \overline{B}^{\text{adv}}_{t,i}}(\theta_1)$, $\nabla_i\theta_2 \leftarrow \nabla_{\theta_2}\overline{L}_{\overline{B}_{t,i} \cup \overline{B}^{\text{adv}}_{t,i}}(\theta_2)$ with the noise $\frac{\chi_3}{m}$
15:    Send $\nabla_i\theta_1$ and $\nabla_i\theta_2$ to the parameter server
16:  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 $\epsilon_r$ 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 $\theta_1$'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 $\epsilon_1$-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 $\chi_2$ 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:*

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

*This results in an $(\epsilon_1/\gamma)$-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 $\chi_1$ drawn as a Laplace noise $[Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})]^d$ and $\chi_2$ 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:*

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

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

*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:*

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

*Consequently, the computation of $\overline{\mathcal{R}}_{\overline{B}_t}(\theta_1)$ preserves $\epsilon_1$-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 $\theta_1$, 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 $(\epsilon_1/\gamma)$-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 $(\epsilon_1/\gamma)$-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 $\theta_{1j}$, with $\forall j \in [1, d]$, can be computed as follows:*

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

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

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

*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 $\Delta_g$, 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:*

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

*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 $(\epsilon_1/\gamma)$-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 $\epsilon_1/\gamma_{\mathbf{x}} \gg \epsilon_1/\frac{\Delta_{\mathcal{R}}}{\Delta_g} \approxeq \epsilon_1 \times 1e-3$, 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*

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

*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 $\chi_3$ 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 $(\epsilon_1/\gamma)$-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*

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

*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*

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

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

*As a result, the computation of $\overline{\mathbf{h}}_{1D}$ is $(\epsilon_1/\gamma)$-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 $\overline{x}'_e \in \overline{D}'$:*

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

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

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

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

*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 $\epsilon_1/\gamma$-DP and $\epsilon_1$-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* $\overline{\mathbf{h}}_{1\overline{B}^{adv}_t} = \{\overline{\theta}_1^T \overline{x}_j^{adv} + \frac{2}{m}Lap(\frac{\Delta_{\mathcal{R}}}{\epsilon_1})\}_{\overline{x}_j^{adv} \in \overline{B}^{adv}_t}$ *given the batch of DP adversarial examples* $\overline{B}^{adv}_t$*, is* $(\epsilon_1/\gamma)$*-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* $\overline{L}_{\overline{B}_t \cup \overline{B}^{adv}_t}(\theta_2)$ *(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* $\overline{B}^{adv}_t$*, 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:*

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

*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*

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

*Furthermore, by applying group privacy, we have that*

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

*By applying Proof 8, it is straight to have*

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

*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 $\epsilon_1$ 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 $\chi_1$ and $2\chi_2$ need to be correspondingly added into $x$ and $h$ in order to make correct predictions. $\chi_1$ 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 $\epsilon_1$ 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 $\psi$ 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 $\chi_1$ 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

$$\frac{1}{|B_t|}|\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1) - \widehat{\mathcal{R}}_{B_t}(\widehat{\theta}_1)| \leq \frac{4e \times d}{(1+e)^2} \tag{34}$$

$$\frac{1}{|B_t|}|\widehat{\mathcal{L}}_{B_t}(\widetilde{\theta}_2) - \widehat{\mathcal{L}}_{B_t}(\widehat{\theta}_2)| \leq \frac{e^2+2e-1}{e(1+e)^2} \times K \tag{35}$$

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

$$\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1) - \widetilde{\mathcal{R}}_{B_t}(\widetilde{\theta}_1) - \widehat{\mathcal{R}}_{B_t}(\theta_1^*) + \widetilde{\mathcal{R}}_{B_t}(\theta_1^*) \leq U - S \tag{36}$$

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

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

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

*If* $U \geq 0$ *and* $S \leq 0$ *then we have:*

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

*Eq. 39 holds for every* $\theta_1^*$, *including* $\widehat{\theta}_1$. *Eq. 39 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:*

$$\widehat{\mathcal{R}}_{B_t}(\theta_1) - \widetilde{\mathcal{R}}_{B_t}(\theta_1) = \sum_{j=1}^{d}\big(\widehat{\mathcal{R}}_{B_t}(\theta_{1j}) - \widetilde{\mathcal{R}}_{B_t}(\theta_{1j})\big) \tag{40}$$

$$= \sum_{j=1}^{d}\Big(\sum_{i=1}^{|B_t|}\sum_{l=1}^{2}\sum_{r=3}^{\infty}\frac{\mathbf{F}_{lj}^{(r)}(z_{lj})}{r!}\big(g_{lj}(x_i, \theta_{1j}) - z_{lj}\big)^r\Big) \tag{41}$$

*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* $|\frac{1}{|B_t|}\big(\widehat{\mathcal{R}}_{B_t}(\theta_1) - \widetilde{\mathcal{R}}_{B_t}(\theta_1)\big)| \leq \sum_{j=1}^{d} \sum_l \frac{\max_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2 - \min_{z_j} \mathbf{F}_{lj}^{(2)}(z_j)(z_j - z_{lj})^2}{2!}$*. 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:*

$$\frac{1}{|B_t|}|\widehat{\mathcal{R}}_{B_t}(\widetilde{\theta}_1) - \widehat{\mathcal{R}}_{B_t}(\widehat{\theta}_1)| \leq \Big[\Big(\frac{e}{(1+e)^2} - \frac{-e}{(1+e)^2}\Big) + \frac{2e}{(1+e)^2}\Big] \times d = \frac{4e \times d}{(1+e)^2} \tag{42}$$

*Consequently, Eq. 34 does hold. Similarly, by looking into the remainder of Taylor expansion for each label k, Eq. 35 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 $3 \times 3$ unit patch, and $\beta$ 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 $\varrho_t$ was set to $1e-4$. Given a predefined total privacy budget $\epsilon$, $\epsilon_2$ is set to be 0.1, and $\epsilon_1$ 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 $\varrho_t$ was set to $5e-2$. In the CIFAR-10 dataset, $\epsilon_2$ 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 $\epsilon$ 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 $\varrho_t$ was set to $1e-2$. In the Tiny ImageNet dataset, $\epsilon_2$ 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 $\epsilon$ 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 $\epsilon$ 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 $\epsilon$ 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 $\mu_a$ 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 $\mu_a$ 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 $\mu_a$. The privacy budget is set to 1.0, offering a reasonable privacy protection. In PixelDP, the construction attack bound $\epsilon_r$ 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 $\mu_a$ 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.

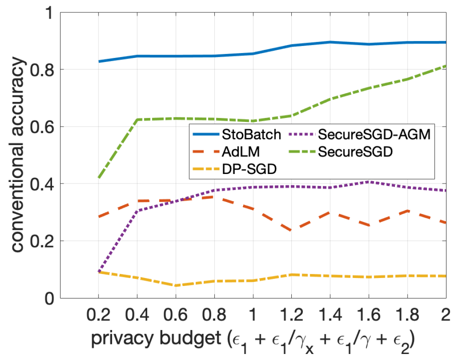
**Results on the CIFAR-10 Dataset** further strengthen our observations. In Figure 3, our StoBatch clearly outperforms baseline models in all cases ($p < 6.17e - 9$), especially when the privacy budget is small ($\epsilon < 4$), yielding strong privacy protections. On average conventional accuracy, our 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 $\mu_a$ under a rigorous DP protection ($\epsilon = 2.0$), compared with baseline approaches (Figure 5). In fact, when the attack size $\mu_a$ 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., $\epsilon_r$ 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 ($p < 2.04e-18$).

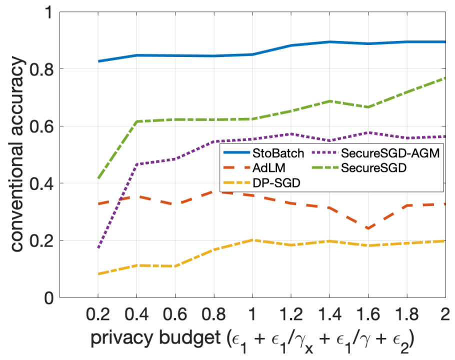
**Scalability under Strong Iterative Attacks.** First, we scale our model in terms of *adversarial training* in the CIFAR-10 dataset, in which the number of iterative attack steps is increased from $T_\mu = 3$ to $T_\mu$ = *200 in training*, and up to $T_a$ = *2,000 in testing*. Note that the traditional 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 ($p < 1.55e-42$).

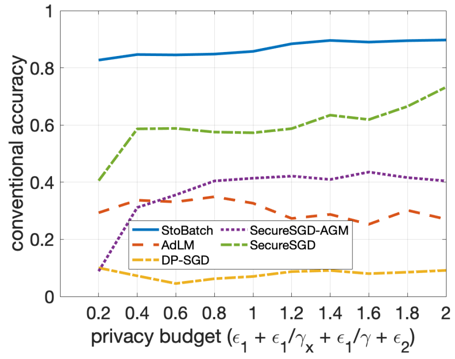
**Key observations: (1)** Incorporating ensemble adversarial learning into DP preservation, tightened sensitivity bounds, a random perturbation size $\mu_t$ at each training step, and composition robustness bounds in both input and latent spaces does enhance the consistency, robustness, and accuracy 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.



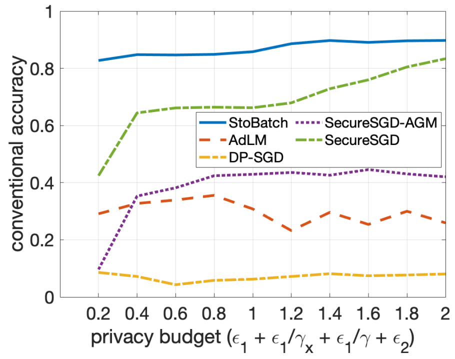
(a) I-FGSM attacks



(b) FGSM attacks

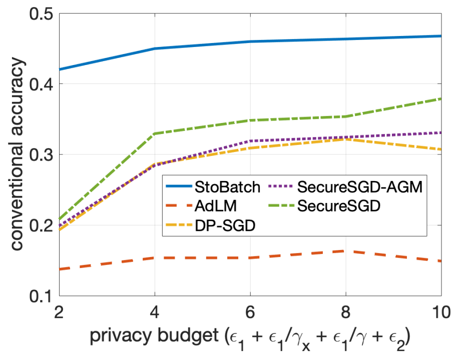


(c) MIM attacks

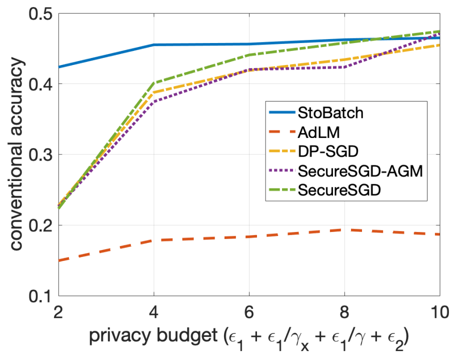


(d) MadryEtAl attacks

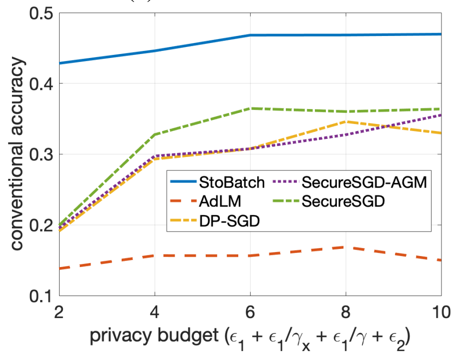
*Figure 2.* Conventional accuracy on the MNIST dataset given $\epsilon$, under $l_\infty(\mu_a = 0.2)$ and $T_a = 10$.



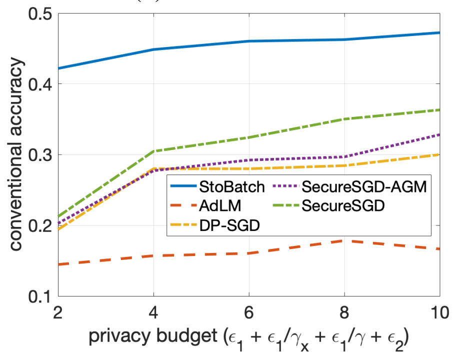
(a) I-FGSM attacks



(b) FGSM attacks

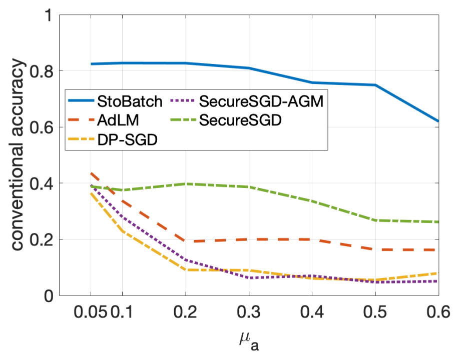


(c) MIM attacks

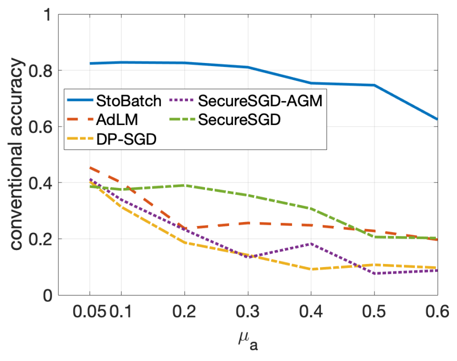


(d) MadryEtAl attacks

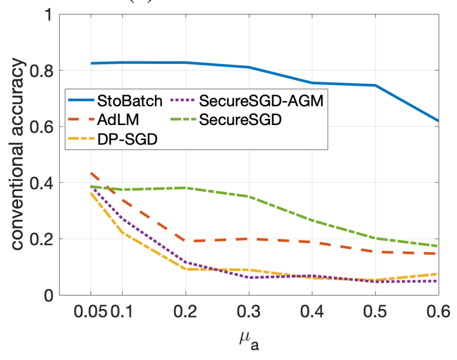
*Figure 3.* Conventional accuracy on the CIFAR-10 dataset given $\epsilon$, under $l_\infty(\mu_a = 0.2)$ and $T_a = 3$.



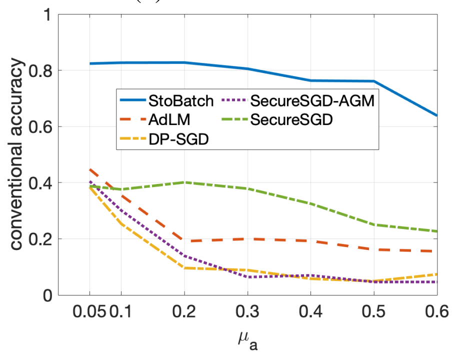
(a) I-FGSM attacks



(b) FGSM attacks

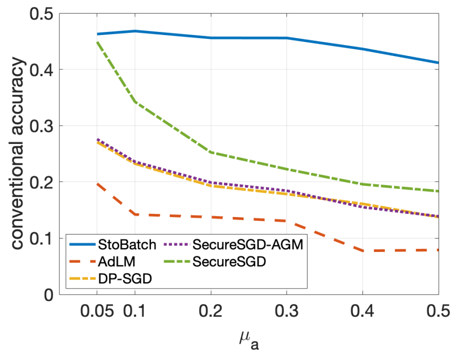


(c) MIM attacks

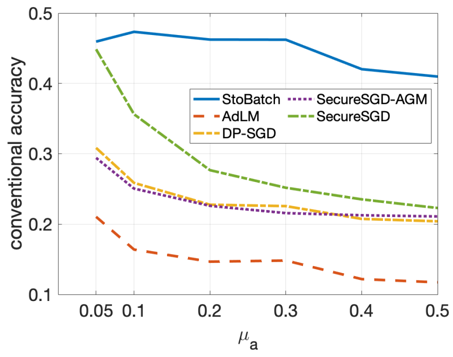


(d) MadryEtAl attacks

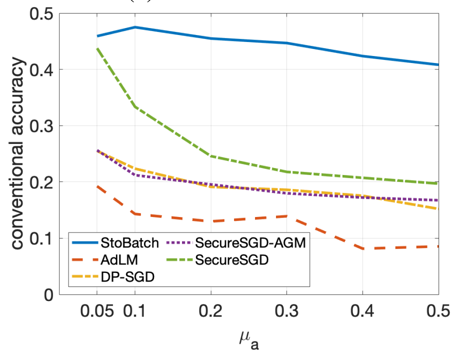
*Figure 4.* Conventional accuracy on the MNIST dataset given $\mu_a$ ($\epsilon = 0.2$, tight DP protection) and $T_a = 10$.



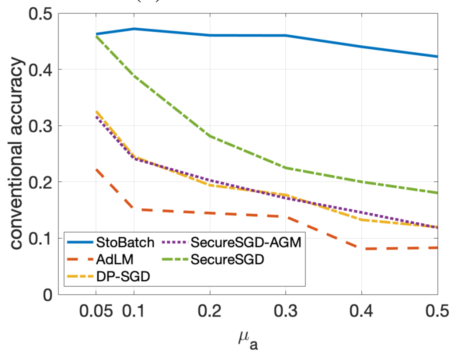
(a) I-FGSM attacks



(b) FGSM attacks

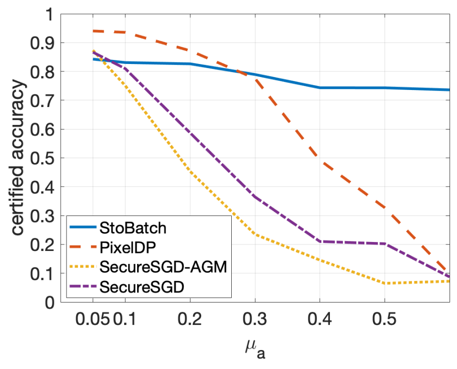


(c) MIM attacks

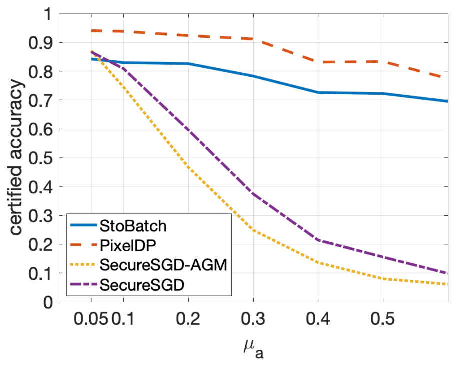


(d) MadryEtAl attacks

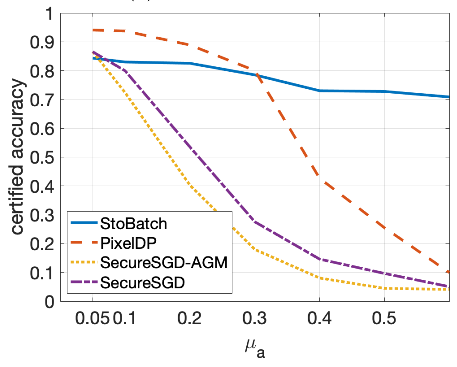
*Figure 5.* Conventional accuracy on the CIFAR-10 dataset given $\mu_a$ ($\epsilon = 2$, tight DP protection) and $T_a = 3$.



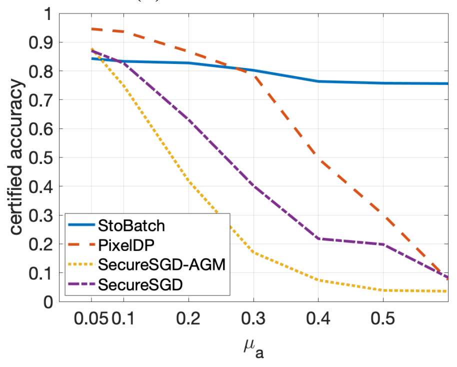
(a) I-FGSM attacks



(b) FGSM attacks

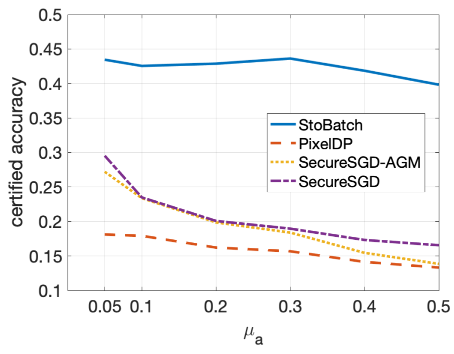


(c) MIM attacks

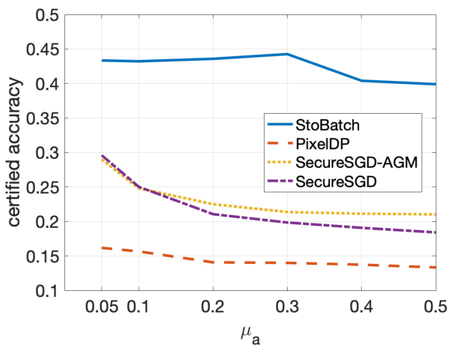


(d) MadryEtAl attacks

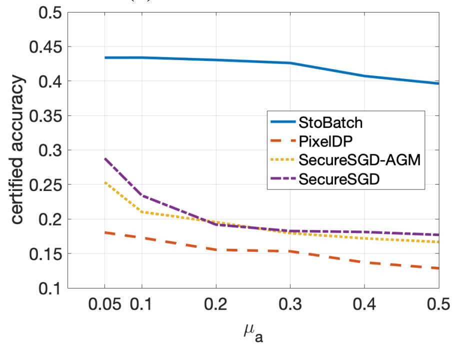
*Figure 6.* Certified accuracy on the MNIST dataset. $\epsilon$ is set to 1.0 (tight DP protection) and $T_a = 10$.



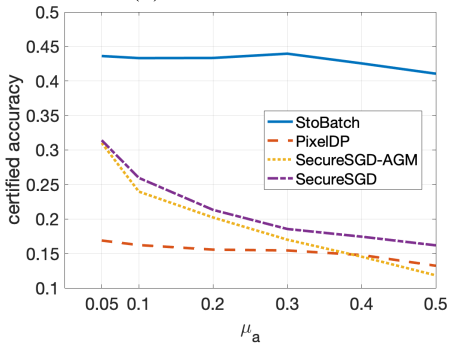
(a) I-FGSM attacks



(b) FGSM attacks

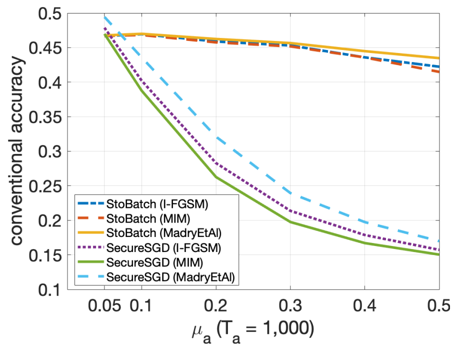


(c) MIM attacks

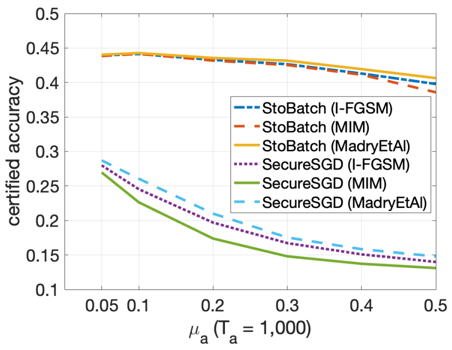


(d) MadryEtAl attacks

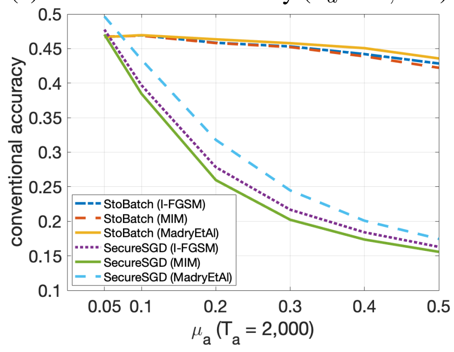
*Figure 7.* Certified accuracy on the CIFAR-10 dataset. $\epsilon$ is set to 2 (tight DP protection) and and $T_a = 3$.



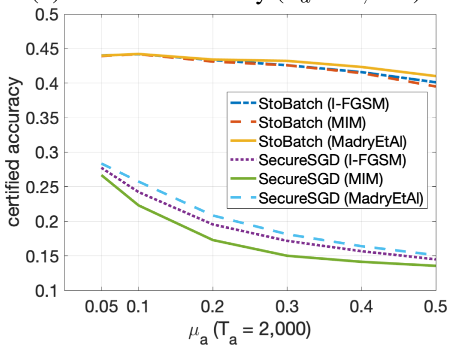
(a) Conventional Accuracy ($T_a = 1,000$)



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

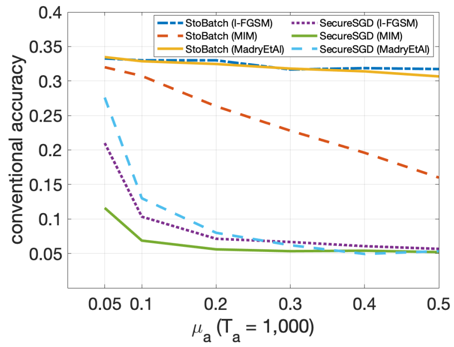


(c) Conventional Accuracy ($T_a = 2,000$)

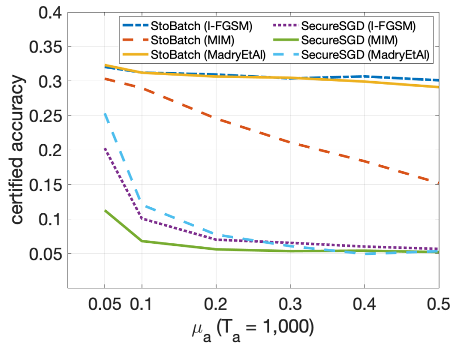


(d) Certified Accuracy ($T_a = 2,000$)

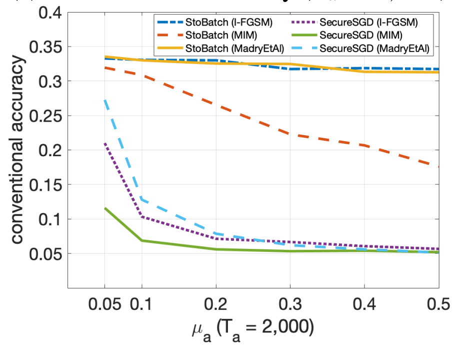
*Figure 8.* Accuracy on the CIFAR-10 dataset, under Strong Iterative Attacks ($T_a = 1,000; 2,000$). $\epsilon$ is set to 2 (tight DP protection).



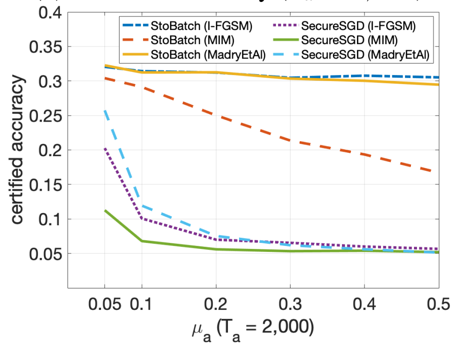
(a) Conventional Accuracy ($T_a = 1,000$)



(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 ($T_a = 1,000; 2,000$). $\epsilon$ is set to 5.