DPSUR: Accelerating Differentially Private Stochastic Gradient Descent Using Selective Update and Release

Jie Fu[1,2], Qingqing Ye[2], Haibo Hu[2], Zhili Chen[1,*], Lulu Wang[1], Kuncan Wang[1], Xun Ran[2]

[1]Shanghai Key Laboratory of Trustworthy Computing, East China Noraml University, China

[2]Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, China

jie.fu@stu.ecnu.edu.cn,qqing.ye@polyu.edu.hk,haibo.hu@polyu.edu.hk,zhlchen@sei.ecnu.edu.cn

luluwang@stu.ecnu.edu.cn,10204804424@stu.ecnu.edu.cn,qi-xun.ran@connect.polyu.hk

ABSTRACT

Machine learning models are known to memorize private data to reduce their training loss, which can be inadvertently exploited by privacy attacks such as model inversion and membership inference. To protect against these attacks, differential privacy (DP) has become the de facto standard for privacy-preserving machine learning, particularly those popular training algorithms using stochastic gradient descent, such as DPSGD. Nonetheless, DPSGD still suffers from severe utility loss due to its slow convergence. This is partially caused by the random sampling, which brings bias and variance to the gradient, and partially by the Gaussian noise, which leads to fluctuation of gradient updates.

Our key idea to address these issues is to apply selective updates to the model training, while discarding those useless or even harmful updates. Motivated by this, this paper proposes DPSUR, a Differentially Private training framework based on Selective Updates and Release, where the gradient from each iteration is evaluated based on a validation test, and only those updates leading to convergence are applied to the model. As such, DPSUR ensures the training in the right direction and thus can achieve faster convergence than DPSGD. The main challenges lie in two aspects — privacy concerns arising from gradient evaluation, and gradient selection strategy for model update. To address the challenges, DPSUR introduces a clipping strategy for update randomization and a threshold mechanism for gradient selection. Experiments conducted on MNIST, FMNIST, CIFAR-10, and IMDB datasets show that DPSUR significantly outperforms previous works in terms of convergence speed and model utility.

1 INTRODUCTION

In the past decade, deep learning techniques have achieved remarkable success in many AI tasks, such as image recognition [32, 52, 70], text analysis [10], and recommendation systems [58]. However, even though the training data are not published, adversaries may still learn them by analyzing the model parameters. For example, the contents of training data can be inverted from the models [21, 38, 43, 56, 72], or the membership information of the training set can be inferred [34, 49]. This is of particular concern in those applications which involve sensitive and personal data, such as medical imaging and finance. Recent legislations such as EU's General Data Privacy Regulation (GDPR) and California Consumer Privacy

Act have mandated machine learning practitioners to take legal responsibility for protecting private data [12].

One of the state-of-the-art paradigms to prevent privacy disclosure in machine learning is differential privacy (DP) [19]. Many works [1, 9, 20, 25, 28, 47, 51, 59] have demonstrated that by adding proper DP noise in the training phase, the resulted machine learning models can prevent unintentional leakage of private training data, such as membership inference attacks.

Among these works, the seminal work by Abadi et al. [1] proposes differentially private stochastic gradient descent (DPSGD) as the training algorithm. In DPSGD, each iteration involves four main steps: (i) randomly selecting a batch of samples using Poisson sampling, (ii) computing and clipping the gradient for each sample, (iii) adding random Gaussian noise to each gradient based on a privacy loss analysis, and (iv) updating the model weights using the average noisy gradients in the batch. In these steps, random Gaussian noise and Poisson sampling are the main reasons to cause slower convergence than conventional SGD [57, 61, 65, 67].

(1) **Gaussian Noise.** In DPSGD, Gaussian noise is added to the gradient in each iteration to satisfy differential privacy. However, the noise scale can be forbiddingly large, leading to inaccurate gradient estimation and poor optimization, especially when it is close to convergence [1].

(2) **Poisson Sampling.** Random sampling in SGD is usually implemented as epoch partitioning in practice. However, in DPSGD random sampling is popularly implemented as Possion sampling for its privacy amplification effect [37]. Since Possion sampling may lead to bias and variance in gradient estimation, resulting in unstable and slow convergence to optimize the objective function [6, 24, 26], we attribute the performance issue of DPSGD partially to random sampling.

Although there are a few recent works which optimize random sampling or Gaussian noise separately [31, 57, 66], as shown by the blue trace in Figure 1, all these works still blindly update the model, whether or not the loss is improved in this iteration. This issue is particularly eminent when the training process approaches convergence and the magnitude of loss is small.

In this paper, we propose DPSUR, a Differentially Private Stochastic Gradient Descent training framework based on Selective Update and Releases. In essence, DPSUR only executes differentially private gradient descent if the gradient is in the correct direction to decrease loss and thus leads to a better model, as shown by the red trace in Figure. 1. This idea is inspired by simulated annealing (SA) [27, 35], where the objective function value (loss) is treated as "energy", and the differentially private model parameters obtained
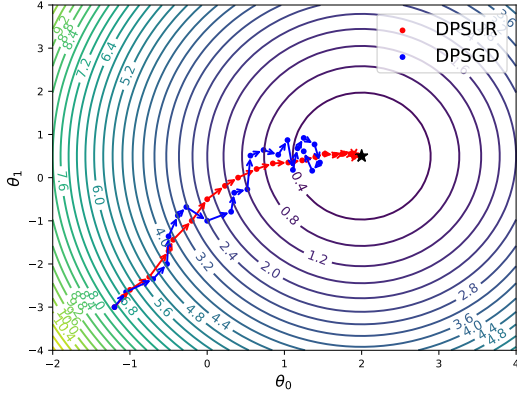
Figure 1: Trajectory visualization of DPSGD and DPSUR schemes on a linear regression model.

in each iteration is a random "solution". As such, we use the difference between the current and previous iteration's loss ΔE as a criterion for accepting or rejecting model parameters.

Selecting those "correct" updates in DPSGD is non-trivial. First, the evaluation of gradient ΔE needs to access a training set, which consumes additional privacy budgets. Second, how to set a threshold for selective update is essential as it directly affects the training performance. These two issues are closely interleaved. To address the first issue, we propose a clipping strategy to ΔE, which employs a minimal clipping bound to minimize the perturbation error. For the second issue, we devise a threshold mechanism for update selection by optimizing the utility gain from selective update.

In addition, we also propose an optimization that selectively releases gradients during multiple iterations, which further reduces the privacy budget consumption. In summary, our contributions are as follows:

- We propose a new DPSUR framework for differentially private deep learning, which uses a validation test to select the model updates and ensures the gradient updates are in the correct direction. To our knowledge, this is the first work to apply selective update for model training under differential privacy.
- A clipping strategy and a threshold mechanism are devised to guarantee DP privacy and optimize utility gain, which enables a faster convergence and better accuracy than DPSGD.
- Furthermore, we propose Gaussian mechanism with selective release to reduce privacy budget consumption across iterations.
- Rigorous theoretical analysis has guaranteed differential privacy of DPSUR, together with a comprehensive empirical evaluation on four public datasets. The results confirm that DPSUR outperforms the state-of-the-art solutions in terms of model accuracy.

The rest of the paper is organized as follows. Section 2 introduces the preliminary knowledge. In Section 3, we present our method of selective update. Section 4 shows the selective release mechanism and the complete algorithm DPSUR. Privacy analysis is conducted in Section 5 and the experimental results are presented in Section 6.

Section 7 introduces related work, followed the conclusion in Section 8.

2 PRELIMINARY KNOWLEDGE

2.1 Differential Privacy

Differential privacy is a rigorous mathematical framework that formally defines data privacy. It requires that a single entry in the input dataset must not lead to statistically significant changes in the output [16, 17, 19] if differential privacy holds.

Definition 2.1. (**Differential Privacy [19]**). The randomized mechanism A provides (ϵ, δ)-Differential Privacy (DP), if for any two neighboring datasets D and D' that differ in only a single entry, $\forall S \subseteq \text{Range}(A)$,

$$\Pr(A(D) \in S) < e^{\epsilon} \times \Pr(A(D') \in S) + \delta. \quad (1)$$

Here, $\epsilon > 0$ controls the level of privacy guarantee in the worst case. The smaller ϵ, the stronger the privacy level is. The factor $\delta > 0$ is the failure probability that the property does not hold. In practice, the value of δ should be negligible [40, 73], particularly less than $\frac{1}{|D|}$.

By adding random noise, we can achieve differential privacy for a function $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ according to Definition 2.1. The l_k-sensitivity determines how much noise is needed and is defined as follow.

Definition 2.2. (**l_k-Sensitivity[17]**) For a function $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$, we define its l_k norm sensitivity (denoted as $\Delta_k f$) over all neighboring datasets $x, x' \in \mathcal{X}^n$ differing in a single sample as

$$\sup_{x,x' \in \mathcal{X}^n} ||f(x) - f(x')||_k \leq \Delta_k f. \quad (2)$$

In this paper, we focus on l_2 sensitivity, i.e., $|| \cdot ||_2$. Additionally, the following Lemma 2.3 ensures the privacy guarantee of post-processing operations.

LEMMA 2.3 (**Post-processing [19]**). *Let $\mathcal{M}$ be a mechanism satisfying (ϵ, δ)-DP. Let f be a function whose input is the output of $\mathcal{M}$. Then $f(\mathcal{M})$ also satisfies (ϵ, δ)-DP.*

2.2 Rényi Differential Privacy

Rényi differential privacy (RDP) is a relaxation of ϵ-differential privacy, which is defined on Rényi divergence as follows.

Definition 2.4. (**Rényi Divergence [54]**) Given two probability distributions P and Q, the Rényi divergence of order $\alpha > 1$ is:

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \ln \mathbf{E}_{x \sim Q} \left[\left(\frac{P(x)}{Q(x)} \right)^\alpha \right], \quad (3)$$

where $\mathbf{E}_{x \sim Q}$ denotes the excepted value of x for the distribution Q, $P(x)$, and $Q(x)$ denotes the density of P or Q at x respectively.

Definition 2.5. (**Rényi Differential Privacy (RDP) [36]**) For any neighboring datasets $x, x' \in \mathcal{X}^n$, a randomized mechanism $\mathcal{M} : \mathcal{X}^n \rightarrow \mathbb{R}^d$ satisfies (α, R)-RDP if

$$D_\alpha(\mathcal{M}(x)||\mathcal{M}(x')) \leq R. \quad (4)$$

The following Definition 2.6 provides a formal definition of Gaussian mechanism, and a formal RDP guarantee by it.

Definition 2.6. (**RDP of Gaussian mechanism [36]**) Assuming f is a real-valued function, and the sensitivity of f is μ, the Gaussian mechanism for approximating f is defined as

$$\mathbf{G}_\sigma f(D) = f(D) + N\left(0, \mu^2 \sigma^2\right), \tag{5}$$

where $N(0, \mu^2\sigma^2)$ is normally distributed random variable with standard deviation $\mu\sigma$ and mean 0. Then the Gaussian mechanism with noise $\mathbf{G}_\sigma$ satisfies $(\alpha, \alpha/2\sigma^2) - RDP$.

The following Lemma 2.7 defines the standard form for converting (α, R)-RDP to (ϵ, δ)-DP.

LEMMA 2.7. (**Conversion from RDP to DP [3]**). *if a randomized mechanism* $f : D \rightarrow \mathbb{R}$ *satisfies* (α, R)-*RDP ,then it satisfies*$(R + \ln((\alpha - 1)/\alpha) - (\ln \delta + \ln \alpha)/(\alpha - 1), \delta)$-*DP for any* $0 < \delta < 1$.

2.3 Deep Learning with Differential Privacy

Differentially Private Stochastic Gradient Descent (DPSGD) is a widely-adopted training algorithm for deep neural networks with differential privacy guarantees. Specifically, in each iteration t, a batch of tuples $\mathcal{B}_t$ is sampled from D with a fixed probability $\frac{b}{|D|}$, where b is the batch size. After computing the gradient of each tuple $x_i \in \mathcal{B}_t$ as $g_t(x_i) = \nabla_{\theta_i} L(\theta_i, x_i)$, where θ_i is model parameter for the i-th sample, DPSGD clips each per-sample gradient according to a fixed ℓ_2 norm (Equation (6)).

$$\begin{aligned} \overline{g}_t(x_i) &= \mathbf{Clip}(g_t(x_i); C) \\ &= g_t(x_i) \Big/ \max\left(1, \frac{\|g_t(x_i)\|_2}{C}\right). \end{aligned} \tag{6}$$

In this way, for any two neighboring datasets, the sensitivity of the query $\sum_{i \in \mathcal{B}_t} g(x_i)$ is bounded by C. Then, it adds Gaussian noise scaling with C to the sum of the gradients when computing the batch-averaged gradients:

$$\tilde{g}_t = \frac{1}{b}\left(\sum_{i \in \mathcal{B}_t} \overline{g}_t(x_i) + \mathcal{N}\left(0, \sigma^2 C^2 \mathbf{I}\right)\right), \tag{7}$$

where σ is the noise multiplier depending on the privacy budget. Last, the gradient descent is performed based on the batch-averaged gradients. Since initial models are randomly generated and independent of the sample data, and the batch-averaged gradients satisfy the differential privacy, the resulted models also satisfy the differential privacy due to the post-processing property.

Privacy accounting. Three factors determine DPSGD's privacy guarantee — the noise multiplier σ, the sampling ratio $\frac{b}{|D|}$, and the number of training iterations T. In reality, given the privacy parameters (ϵ, δ), we can set appropriate values for these three hyper-parameters to optimize the performance. The privacy calibration process is performed using a privacy accountant: a numerical algorithm providing tight upper bounds for the given (ϵ, δ) as a function of the hyper-parameters [1], which in turn can be combined with numerical optimization routines to optimize one hyper-parameter given the other two. In this work we use the RDP [36] for privacy accounting. In practice, given σ, δ and b at each iteration, we select α from $\{2, 3, ..., 64\}$ to determine the smallest ϵ.

3 DPSUR: DP TRAINING FRAMEWORK WITH SELECTIVE UPDATES AND RELEASE

In this section, we present our proposed framework DPSUR, with an overview in Section 3.1. Then two key components of DPSUR, namely, minimal clipping strategy and threshold mechanism,are introduced in Sections 3.2 and 3.3, respectively.

3.1 Overview

As aforementioned, DPSUR does not directly accept the model updates from each iteration due to the influence of random sampling and Gaussian noise. Therefore, we first calculate the loss of the generated model in each iteration, and then compare it with that from the last iteration to determine whether or not to accept the model update.
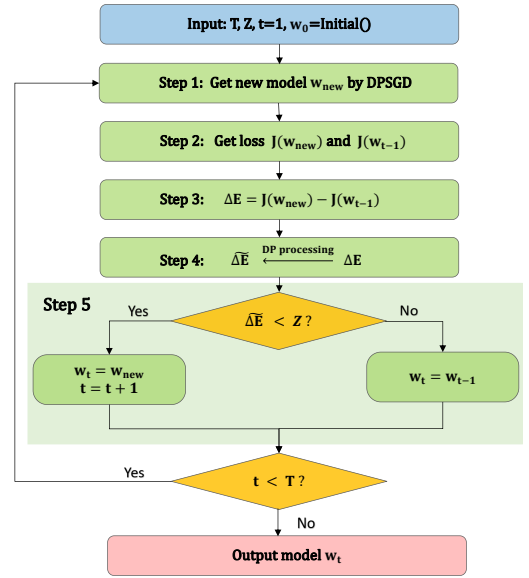
Figure 2: Workflow of DPSUR.

Figure 2 shows the workflow of DPSUR, which takes as inputs the total number of updates T, number of updates accepted t, acceptance threshold Z, and initialization model w_0, executes the following steps, and outputs a final model.

- **Step 1:** In each iteration, we obtain a batch of tuples from the training set via Poisson sampling, and generate an intermediate model w_{new} using the DPSGD algorithm.
- **Step 2:** We evaluate the intermediate model w_{new} and w_{t-1} on the validation batch $\mathcal{B}_v$ resampled from training set to calculate the loss $J(w_{new})$ and $J(w_{t-1})$.
- **Step 3:** We compute the difference of loss $\Delta E = J(w_{new}) - J(w_{t-1})$ to evaluate the performance of intermediate model.
- **Step 4:** We clip ΔE and add noise to it to satisfy differential privacy, obtaining $\widetilde{\Delta E}$.
- **Step 5:** Given the acceptance threshold Z, we accept update of the intermediate model w_{new} and t plus 1 if $\widetilde{\Delta E} < Z$, or reject it otherwise by reverting back to the last model w_{t-1} that was accepted for update.

Note that when t reaches the total number of updates T, we output the trained model w_t.

Figure 3 further shows how we evaluate the model, i.e. obtaining $J(w)$ (step 2 in Figure 2). During each iteration, we first randomly sample a portion of the tuples $\mathcal{B}_t$ from the training set and perform DPSGD training to obtain a model. Then we randomly re-sample a portion of tuples from the training set as validation batch $\mathcal{B}_v$. Finally, the cross-entropy loss function is applied to the $\mathcal{B}_v$ to compute the loss of the current model. It is important to note that sampling from the training set for training and validation serves the purpose of privacy amplification. Specifically, re-sampling from the training set for model validation helps prevent overfitting. The complete DPSUR algorithm will be described in Section 4.2.
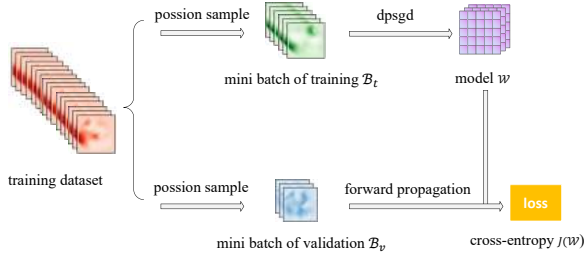

Figure 3: Model evaluation framework in DPSUR.

3.2 Minimal Clipping

As the computation of ΔE accesses the training set (i.e., a portion of private tuples), it is necessary to perform a differential privacy operation on it. Intuitively, we can clip ΔE to a certain range $[-C_v, C_v]$, and then add Gaussian noise with mean 0 and standard deviation $2C_v \cdot \sigma_v$ to ensure differential privacy, as shown in Equation (8). Here, σ_v denotes the noise multiplier, and $2C_v$ is the sensitivity of the clipping operation.

$$\begin{aligned}\widetilde{\Delta E} &= \mathbf{Clip}(\Delta E; C_v) + \sigma_v \cdot 2C_v \cdot \mathcal{N}(0,1), \\ &= min(max(\Delta E, -C_v), C_v) + \sigma_v \cdot 2C_v \cdot \mathcal{N}(0,1)\end{aligned} \quad (8)$$

However, setting an appropriate clipping bound C_v for ΔE is challenging. A large C_v helps avoid loss of fidelity to the original values, but it also leads to large injected noise. A key observation here is that, to assess the quality of the current model, we can simply compare the loss of the model trained in this iteration with that of the previous iteration, i.e., $\Delta E = J(w_{new}) - J(w_{t-1})$. If the loss is lower than the previous iteration, i.e., $\Delta E < 0$, the current model w_{new} is better and we accept it, otherwise we reject it. Therefore, we only need to determine whether ΔE is positive or negative, instead of its absolute value. With that said, we can use a small clipping bound C_v for uniform clipping such that almost all ΔE are outside the interval $[-C_v, C_v]$, which is illustrated by the red line in Figure 4.

$$\overline{\Delta E} = \begin{cases} -C_v, & \Delta E \leq -C_v \\ \Delta E, & -C_v < \Delta E < C_v \\ C_v, & \Delta E \geq C_v \end{cases} . \quad (9)$$

We clip ΔE to a certain range $[-C_v, C_v]$ as shown in Equation 9. If we choose C_v to be extremely small (e.g., $C_v = 1e - 10$), we can
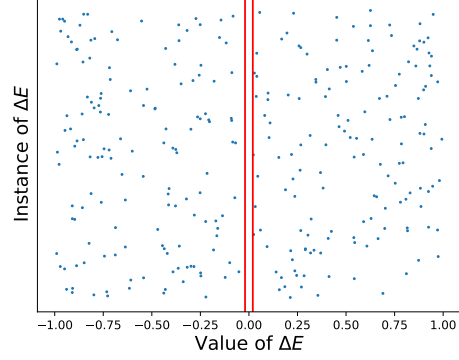

Figure 4: The idea of minimal cilpping.

make almost all ΔE stay outside $[-C_v, C_v]$, and the clipping process is simplified to Equation 10. Although in the worst case, there could be some value of ΔE in the interval of $[-C_v, C_v]$, the probability is so small that we can ignore it without affecting our technical analysis. In practice, we can select the clipping bound C_v to be small enough for loss value in gradient descent, e.g., $C_v = 0.001$.

$$\overline{\Delta E} = \begin{cases} -C_v, & \Delta E < 0 \\ C_v, & \Delta E > 0 \end{cases} . \quad (10)$$

So each possible value of ΔE is discretized as $-C_v$ or C_v after our minimal clipping strategy. A small clipping bound C_v ensures that the injected Gaussian noise is sufficiently small. Such clipping strategy can quantify the impact of the injected Gaussian noise. As plotted in Figure 5, given threshold Z (shown as the red lines), after minimal clipping and adding Gaussian noise, values of ΔE consist of two (partial) Gaussian distributions with a mean of C_v or $-C_v$ and a standard deviation of $2C_v\sigma_v$, respectively. The blue area represents the probability of the Gaussian distribution with a mean of C_v and less than threshold Z, while the orange and blue area together represents the probability of the Gaussian distribution with a mean of $-C_v$ and less than threshold Z.

For a real example, let clipping bound $C_v = 0.1$ and noise level $\sigma_v = 1$. The results by the clipping strategy after adding noise are plotted in Figure 5a, where threshold $Z = 0$ is used to accept those updates whose $\Delta E < Z$. We observe that the probability of accepting a high-quality model ($\Delta E < 0$) is 69.1%, while the probability of accepting a low-quality model ($\Delta E > 0$) is 30.8%. Next, we slightly move the acceptance threshold Z from 0 to -0.2 to investigate its impact on the acceptance probabilities. Figures 5b and 5c show the results of $Z = -0.1$ and $Z = -0.2$, respectively. We observe that, from $Z = 0$ to $Z = -0.1$, the probability of accepting a high-quality model drops much more slowly (from 69.1% to 50%) than that of accepting a low-quality model (from 30.8% to 15.9%). However, as Z further decreases, the situation is reversed, i.e., the probability of accepting a high-quality model drops quickly, which goes against the model convergence. This observation motivates us to explore an optimal threshold Z to maximize the utility gain from model updates.

(a) Z = 0   **(b) Z = -0.1**   **(c) Z = -0.2**
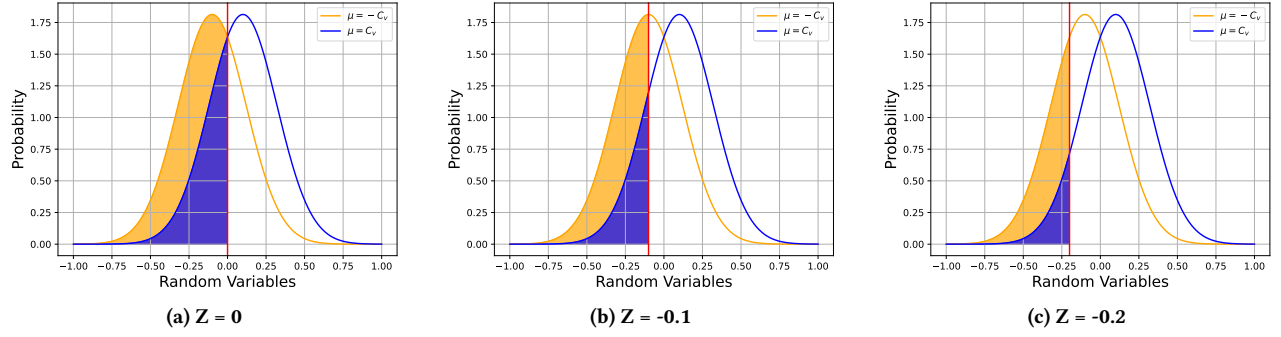
Figure 5: The Gaussian probability distributions under different thresholds Z, where the Gaussian distribution $\mathcal{N}(\mu, (2C_v \cdot \sigma_v)^2)$ and the $\sigma_v = 1, C_v = 0.1$

3.3 Threshold Mechanism

In this subsection, we derive an optimal threshold Z. Formally, let $Z = \beta \cdot C_v$. Then for the case of $\Delta E < 0$, the CDF of the Gaussian distribution $\mathcal{N}(-C_v, (2C_v \cdot \sigma_v)^2)$ less than Z is

$$P(X < Z) = \Phi(\frac{\beta \cdot C_v - (-C_v)}{2C_v \cdot \sigma_v}) = \Phi(\frac{\beta + 1}{2\sigma_v}), \qquad (11)$$

where $X = \mathcal{N}(-C_v, (2C_v \cdot \sigma_v)^2)$ and $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$. Similarly, the CDF of the case $\Delta E > 0$ is

$$P(X < Z) = \Phi(\frac{\beta \cdot C_v - C_v}{2C_v \cdot \sigma_v}) = \Phi(\frac{\beta - 1}{2\sigma_v}), \qquad (12)$$

where $X = \mathcal{N}(C_v, (2C_v \cdot \sigma_v)^2)$. According to Equations (11) and (12), the parameter C_v will be eliminated in the CDF of Gaussian distribution, so the acceptance probability of the algorithm only depends on the parameters β and σ_v.

In Figure 6, we plot the probabilities of accepting high-quality ($\Delta E < 0$) and low-quality ($\Delta E > 0$) models for popular σ_v: 0.8, 1.0, and 1.3, with respect to β ranging from -3.5 to 0.5. Obviously, smaller β value results in decreasing probabilities of accepting both high-quality ($\Delta E < 0$) and low-quality ($\Delta E > 0$) models. To find a β that maximizes the difference between the two probabilities (i.e., the red dashed vertical line), we find that such β seems to be around -1.0 for all three σ_v. As will be shown in Figure 9c of the experimental results, setting $\beta = -1.0$ does achieve excellent results in all datasets.
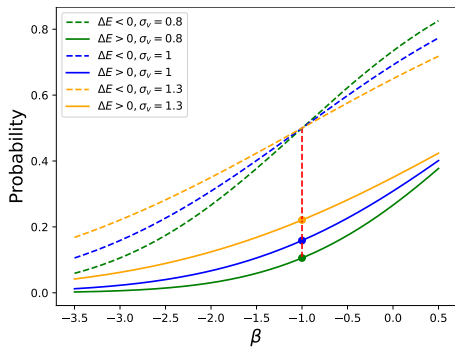
Figure 6: Acceptance probability vs. β.

Summary. To address the privacy concerns arising from the evaluation on selective update, we propose a randomized algorithm coupled with a minimal clipping strategy and a threshold mechanism. In particular, by setting a sufficiently small clipping bound C_v (e.g., 0.001) and the acceptance threshold $Z = \beta \cdot C_v$, the impact of C_v on the selection update is eliminated according to Equations (11) and (12). Further, a suitable β is selected based on probability distributions to achieve maximum utility for selective updates.

4 SELECTIVE RELEASE IN DPSUR: AN OPTIMIZATION

As of now, iteratively calculating the $\widetilde{\Delta E}$ in each iteration requires continual privacy budget consumption. In Section 4.1, we prove that by only releasing the model when a selective update occurs, DPSUR can preserve the privacy budget. Finally, we summarize the overall algorithm of DPSUR in Section 4.2.

4.1 Gaussian Mechanism with Selective Release

Recall that when we obtain $\widetilde{\Delta E}$, to protect privacy, we clip ΔE in $[-C_v, C_v]$ and add Gaussian noise according to Equation 8 in **each** iteration. According to step 5 of Figure 2, we do not update the model until $\widetilde{\Delta E}$ exceeds the threshold Z. If we take one step further and do not release $\widetilde{\Delta E}$ in such cases, as described in Algorithm 1, DPSUR only consumes privacy budget when a selective update occurs, i.e., $\widetilde{\Delta E} < \beta \cdot C_v$. In this subsection, we conduct privacy analysis to ensure Algorithm 1 can satisfy the same RDP guarantee as that of the underlying Gaussian mechanism in Definition 2.6.

In general, for the query result $f(D)$ on dataset D, Algorithm 1 clip the $f(D)$ in $[0, \mu]$ and adds Gaussian noise until the noisy result falls in a designated interval $[a, b]$. The following Theorem 4.1 shows that when $a \rightarrow -\infty$, Algorithm 1 can satisfy the same RDP guarantee.

THEOREM 4.1. *Algorithm 1 satisfies $(\alpha, \alpha/2\sigma^2) - RDP$ when $a \rightarrow -\infty$.*

PROOF. Figure 7a plots two normal distributions, namely $N(0, \mu^2\sigma^2)$ and $N(\mu, \mu^2\sigma^2)$, as the output probability distributions of function f on two neighboring datasets whose sensitivity is μ. By selective update and selective release in the threshold range $[a, b]$, the probability distribution outside of this interval will accumulate within

Algorithm 1: Gaussian mechanism with selective release

Input: function $f(\cdot)$, dataset D, Gaussian distribution N, a given interval $[a, b]$

Output: $A(D)$ that falls in the interval $[a, b]$ after adding Gaussian noise

1. Clipping $f(D)$ to interval $[0, \mu]$ to obtain sensitivity μ;
2. $A(D) = f(D) + N(0, \mu^2\sigma^2)$;
3. **while** $A(D) < a$ or $A(D) > b$ **do**
4. $\quad A(D) = f(D) + N(0, \mu^2\sigma^2)$;
5. **return** $A(D)$

(a) Before truncation.   **(b) After truncation.**

Figure 7: Before and after truncating the normal distribution.

the interval. As such, the final output distribution is truncated and transformed into a truncated normal distribution, as shown in Figure 7b. As x is assumed to follow a normal distribution, the truncated normal distribution with mean 0 and mean μ can be represented as follows:

$$f(x; 0, \mu\sigma, a, b) = \begin{cases} \frac{1}{\mu\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\mu^2\sigma^2}} \cdot \frac{1}{\Phi\left(\frac{b}{\mu\sigma}\right) - \Phi\left(\frac{a}{\mu\sigma}\right)} & a \le x \le b, \\ 0 & \text{otherwise.} \end{cases}$$
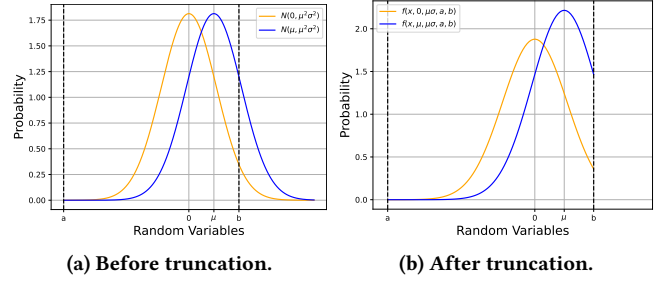
$$f(x; \mu, \mu\sigma, a, b) = \begin{cases} \frac{1}{\mu\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\mu^2\sigma^2}} \cdot \frac{1}{\Phi\left(\frac{b-\mu}{\mu\sigma}\right) - \Phi\left(\frac{a-\mu}{\mu\sigma}\right)} & a \le x \le b, \\ 0 & \text{otherwise.} \end{cases}$$

Here $\mu\sigma$ is the standard deviation of the original normal distribution, whereas a and b are the lower and upper truncation values, respectively. $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution.

Then we substitute the two truncated normal distributions into Rényi divergence [54] to calculate the RDP as follows:

$$D_\alpha(f(x; 0, \mu\sigma, a, b) || f(x; \mu, \mu\sigma, a, b))$$

$$= \frac{1}{\alpha - 1} \cdot \ln \int_a^b \frac{[f(x; 0, \mu\sigma, a, b)]^\alpha}{[f(x; \mu, \mu\sigma, a, b)]^{\alpha-1}} dx$$

$$= \frac{1}{\alpha - 1} \cdot \ln\{ \frac{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^\alpha} \cdot \frac{1}{\mu\sigma\sqrt{2\pi}} \int_a^b \exp[(-x^2$$

$$+ 2(1-\alpha)\mu x - (1-\alpha)\mu^2)/(2\mu^2\sigma^2)] dx\}$$

$$= \frac{1}{\alpha - 1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \ln[\frac{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^\alpha}$$

$$\cdot (\Phi(\frac{b - (1-\alpha)\mu}{\mu\sigma}) - \Phi(\frac{a - (1-\alpha)\mu}{\mu\sigma}))]\}$$

$$= \frac{\alpha}{2\sigma^2} + \frac{1}{\alpha - 1} \cdot \ln\{ \frac{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^\alpha} \cdot [\Phi(\frac{b - (1-\alpha)\mu}{\mu\sigma})$$

$$- \Phi(\frac{a - (1-\alpha)\mu}{\mu\sigma})]\},$$

$$(13)$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt$.

As $a \to -\infty$, we can get:

$$D_\alpha(f(x; 0, \mu\sigma, a, b) || f(x; \mu, \mu\sigma, a, b))$$

$$= \frac{\alpha}{2\sigma^2} + \frac{1}{\alpha - 1} \cdot \ln[\frac{(\Phi(\frac{b-\mu}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b}{\mu\sigma}))^\alpha} \cdot \Phi(\frac{b + (\alpha-1)\mu}{\mu\sigma})] \quad (14)$$

Similarly, we can get :

$$D_\alpha(f(x; \mu, \mu\sigma, a, b) || f(x; 0, \mu\sigma, a, b))$$

$$= \frac{1}{\alpha - 1} \cdot \ln \int_a^b \frac{[f(x; \mu, \mu\sigma, a, b)]^\alpha}{[f(x; 0, \mu\sigma, a, b)]^{\alpha-1}} dx$$

$$= \frac{1}{\alpha - 1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \ln[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha}$$

$$\cdot (\Phi(\frac{b - \alpha\mu}{\mu\sigma}) - \Phi(\frac{a - \alpha\mu}{\mu\sigma}))]\} \quad (15)$$

$$= \frac{\alpha}{2\sigma^2} + \frac{1}{\alpha - 1} \cdot \ln[\frac{(\Phi(\frac{b}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}))^\alpha} \cdot \Phi(\frac{b - \alpha\mu}{\mu\sigma})]$$

According Theorem 4.2, we have:

$$D_\alpha(f(x; 0, \mu\sigma, a, b) || f(x; \mu, \mu\sigma, a, b)) \le \alpha/2\sigma^2, and$$

$$D_\alpha(f(x; \mu, \mu\sigma, a, b) || f(x; 0, \mu\sigma, a, b)) \le \alpha/2\sigma^2.$$

Therefore, Theorem 4.1 is proved. The details of RDP of two truncated normal distributions are presented in Appendix A. □

Next, we will prove Theorem 4.2. This theorem is an inequality proof of the cumulative distribution function (CDF) of the normal distribution, and it provides the foundation for the proof of Theorem 4.1.

THEOREM 4.2. *If* $A = (\Phi(\frac{b-\mu}{\mu\sigma}))^{\alpha-1} \cdot \Phi(\frac{b+(\alpha-1)\mu}{\mu\sigma})/(\Phi(\frac{b}{\mu\sigma}))^\alpha$, $B = (\Phi(\frac{b}{\mu\sigma}))^{\alpha-1} \cdot \Phi(\frac{b-\alpha\mu}{\mu\sigma})/(\Phi(\frac{b-\mu}{\mu\sigma}))^\alpha$, *where* $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt$. $A, B \le 1$, *when* $\mu > 0, \sigma > 0$ *and* $\alpha > 1$.

PROOF. The second derivative of $\ln[\Phi(x)]$ is:

$$\ln''[\Phi(x)] = \frac{\Phi''(x) \cdot \Phi(x) - \Phi'(x) \cdot \Phi'(x)}{[\Phi(x)]^2}$$

$$= \frac{-x \cdot \Phi'(x) \cdot \Phi(x) - \Phi'(x) \cdot \Phi'(x)}{[\Phi(x)]^2}, \quad (16)$$

where $\Phi'(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

When $x \ge 0$, since $\Phi(x), \Phi'(x) > 0$, $\ln''[\Phi(x)] < 0$ always hold.

When $x < 0$, $\Phi'(x) > 0$. As such, proving $-x \cdot \Phi'(x) \cdot \Phi(x) - \Phi'(x) \cdot \Phi'(x) < 0$ is equivalent to proving $-x \cdot \Phi(x) - \Phi'(x) < 0$. We let $K(x) = -x \cdot \Phi(x) - \Phi'(x)$, so the derivative for $K(x)$ is:

$$K'(x) = -\Phi(x) + (-x) \cdot \Phi'(x) - \Phi''(x)$$
$$= -\Phi(x) + (-x) \cdot \Phi'(x) - (-x) \cdot \Phi'(x) \qquad (17)$$
$$= -\Phi(x) < 0$$

Since $\lim_{x \to -\infty} K(x) = 0$, $K(x) < 0$ when $x < 0$. In summary, $\ln''[\Phi(x)] < 0$ always holds. In other words, $\ln''[\Phi(x)]$ is the concave function when $x \in (-\infty, \infty)$. According to the properties of the concave function, we can obtain:

$\lambda \cdot \ln[\Phi(x_1)] + (1 - \lambda) \ln[\Phi(x_2)] \le \ln[\Phi(\lambda x_1 + (1 - \lambda)x_2)]$,

where $\lambda \in (0, 1)$ and $x_1, x_2 \in (-\infty, \infty)$.

Let $\lambda = \frac{\alpha-1}{\alpha} (\alpha > 1)$, $x_1 = \frac{b-\mu}{\mu\sigma}$, $x_2 = \frac{b+(\alpha-1)\mu}{\mu\sigma}$ or $x_1 = \frac{b}{\mu\sigma}$, $x_2 = \frac{b-\alpha\mu}{\mu\sigma}$. Then the following two inequalities hold:

$$\frac{(\alpha-1)}{\alpha} \cdot \ln[\Phi(\frac{b-\mu}{\mu\sigma})] + \frac{1}{\alpha} \ln[\Phi(\frac{b+(\alpha-1)\mu}{\mu\sigma})] \le \ln[\Phi(\frac{b}{\mu\sigma})]$$
$$\frac{(\alpha-1)}{\alpha} \cdot \ln[\Phi(\frac{b}{\mu\sigma})] + \frac{1}{\alpha} \ln[\Phi(\frac{b-\alpha\mu}{\mu\sigma})] \le \ln[\Phi(\frac{b-\mu}{\mu\sigma})]$$
$$(18)$$

In the end, we can obtain the following:

$$(\Phi(\frac{b-\mu}{\mu\sigma}))^{\alpha-1} \cdot \Phi(\frac{b+(\alpha-1)\mu}{\mu\sigma}) \le (\Phi(\frac{b}{\mu\sigma}))^{\alpha}$$
$$(\Phi(\frac{b}{\mu\sigma}))^{\alpha-1} \cdot \Phi(\frac{b-\alpha\mu}{\mu\sigma}) \le (\Phi(\frac{b-\mu}{\mu\sigma}))^{\alpha}, \qquad (19)$$

where $\mu, \sigma > 0$ and $\alpha > 1$.
Theorem 4.2 is proved. □

This proves Algorithm 1 satisfies the same $(\alpha, \alpha/2\sigma^2)$-RDP as Gaussian mechanism of RDP [36]. In other words, DPSUR only consumes privacy budget when the update is selected and released based on the interval. As such, in Figure 2, the privacy budget of computing $\widetilde{\Delta E}$ is consumed only if $\widetilde{\Delta E} < Z$.

4.2 DPSUR: Putting Things Together

Now we describe the overall algorithm of DPSUR in Algorithm 2, which consists of the following two steps.

i. DPSGD (Lines 3-8). This part is the traditional DPSGD procedure. First, a small batch of samples $\mathcal{B}_t$ are randomly selected from the training datasets (Line 3). For each sample $x_i \in \mathcal{B}_t$, its gradient values are calculated and clipped so that the l_2 norm of the gradients is not greater than the clipping bound C_t (Lines 4-6). In Line 7, the clipped gradients are first summed up, and then added Gaussian noise $\mathcal{N}(0, C_t^2\sigma_t^2)$ to satisfy differential privacy, and finally averaged. As such, the sensitivity is C_t here. Gradient descent is then performed using these noisy gradients to obtain a new temporary model w_{new} for the current iteration (Line 8).

ii. Selective update (Lines 9-18). First, a small batch of samples $\mathcal{B}_v$ are randomly selected from the training set (Line 9). Then we calculate the loss for temporary model $J(w_{new})$ and the latest accepted model $J(w_{t-1})$, where $J(w) = 1/|\mathcal{B}_v| \sum_{x \in \mathcal{B}_v} \mathcal{L}(w, x)$, and subtract them to get ΔE (Lines 10- 11). To get the sensitivity, ΔE is clipped to $[-C_v, C_v]$, which means that one less or one more sample

produces a maximum variation of $2C_v$ for ΔE (Line 12). To ensure differential privacy, Gaussian noise $\mathcal{N}(0, \sigma_v^2 \cdot (2C_v)^2)$ is added to obtain a noisy version $\widetilde{\Delta E}$ (Lines 13). The temporary model w_{new} will be accepted and t plus 1 if $\widetilde{\Delta E} < \beta \cdot C_v$, where β is the acceptance threshold parameter (Lines 15- 16). Otherwise, the temporary model w_{new} is rejected, and the model from the last iteration is returned (Line 18).

The above two steps are repeated until the entire privacy budget is consumed. We will provide rigorous privacy analysis in the next section.

Algorithm 2: Overall algorithm of DPSUR

Input: training datasets $\{x_1, x_2, \ldots, x_N\}$, loss function $\mathcal{L}(\theta, x)$. Parameters: learning rate η, batch size for training B_t, noise multiplier for training σ_t, clipping bound for training C_t, batch size for validation B_v, noise multiplier for validation σ_v, clipping bound for validation C_v, threshold parameter β

Output: the final trained model w_t

1 Initialize $t = 1$, $w_0 = \text{Initial}()$;
2 **while** $t < T$ **do**
3   Randomly sample a batch $\mathcal{B}_t$ with probability $\frac{B_t}{N}$;
4   **for** $x_i \in \mathcal{B}_t$ **do**
5   Compute $g_t(x_i) \leftarrow \nabla \mathcal{L}_{(w_t, x_i)}$;
6        $\bar{g}_t(x_i) \leftarrow g_t(x_i)/\max(1, \frac{||g_t(x_i)||_2}{C_t})$;
7     $\widetilde{g}_t \leftarrow \frac{1}{|\mathcal{B}_t|}(\sum_{i \in \mathcal{B}_t} \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C_t^2))$;
8     $w_{new} = w_{t-1} - \eta_t \widetilde{g}_t$;
9   Poisson sampling a batch $\mathcal{B}_v$ with probability $\frac{B_v}{N}$;
10   Compute loss $J(w_{new})$ and $J(w_{t-1})$ by batch $\mathcal{B}_v$;
11    $\Delta E = J(w_{new}) - J(w_{t-1})$;
12    $\overline{\Delta E} = \min(\max(\Delta E, -C_v), C_v)$;
13    $\widetilde{\Delta E} = \overline{\Delta E} + \mathcal{N}(0, \sigma_v^2 \cdot (2C_v)^2)$;
14   **if** $\widetilde{\Delta E} < \beta \cdot C_v$ **then**
15       $w_t = w_{new}$;
16       $t = t + 1$;
17   **else**
18       $w_t = w_{t-1}$;
19 **return** w_t;

---

5 PRIVACY ANALYSIS

This section establishes the privacy guarantee of DPSUR. Since DPSUR is non-interactive, it only releases the final model accumulated from all accepted model updates. In the following, we first analyze the privacy loss of each accepted model update, and then derive the total privacy loss based on sequential composition.

For each accepted model update in Algorithm 2, there are only two places where the raw training data is accessed. One is the computation of model updates w_{new} in the training phase, and the other is the computation of test values $\widetilde{\Delta E}$ in the validation phase. As shown in Figure 8, the model update and test value sequences are in the form of "rejected, ..., rejected, accepted".

In the validation phase, the computation of each test value $\widetilde{\Delta E}$ takes as input the training set and the corresponding model update w_{new}, and the latter is a function of the training set. Due to the function composition, the computation of $\widetilde{\Delta E}$ can be regarded as a function of the training set whose output is a value in $[-C_v, C_v]$. Therefore, Line 13 in Algorithm 2 ensures that the computation satisfies differential privacy due to the Gaussian mechanism. Furthermore, computing and testing the values $\widetilde{\Delta E}$ until one value is accepted satisfies the same differential privacy according to Theorem 4.1.

In the training phase, since the accepted model update is selected solely based on the test value $\widetilde{\Delta E}$ which satisfies differential privacy, the model update selection satisfies differential privacy due to the post-processing property. Additionally, all rejected model updates are never released and thus consume no privacy budget, even though they have been computed. Therefore, the privacy loss in the training phase comes only from the computation of accepted model updates.

Based on the above analysis, we will use the Rényi Differential Privacy (RDP) approach to calculate the privacy loss in the training and validation phases in Sections 5.1 and 5.2 respectively, compose them sequentially, and finally convert the RDP into (ϵ, δ)-DP in Section 5.3.
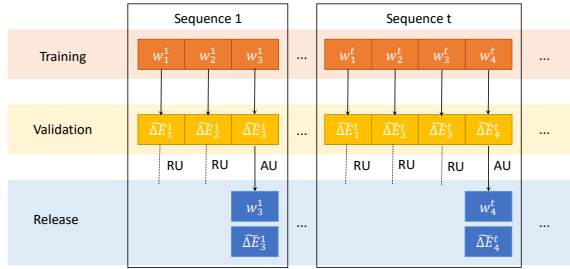

Figure 8: The privacy analysis of DPSUR. (RU: Rejected Update, AU: Accepted Update)

5.1 Privacy Analysis of Training

As mentioned above, the RDP of the training phase is only resulted from the accepted model updates. Theorem 5.1 gives the proof of RDP in the training phase.

THEOREM 5.1. *After accepting t model updates, the RDP of the training phase of DPSUR satisfies:*

$$R_{train}(\alpha) = \frac{t}{\alpha - 1} \ln \left[\sum_{i=0}^{\alpha} \left(\begin{array}{c} \alpha \\ i \end{array} \right) (1-q)^{\alpha - i} q^i \exp \left(\frac{i^2 - i}{2\sigma_t^2} \right) \right],$$
(20)

where $q = \frac{B_t}{N}$, σ_t is noise multiplier of the training phase, and $\alpha > 1$ is the order.

PROOF. We will prove this in the following two steps: (i) use the RDP of the sampling Gaussian mechanism to calculate the privacy cost of each accepted model update, which relies on Definitions 5.2 and 5.3, and (ii) use the composition of RDP mechanisms to compute the privacy cost of multiple accepted model updates by Lemma 5.4.

Definitions 5.2 and 5.3 define Sampled Gaussian Mechanism (SGM) and its Rényi Differential Privacy (RDP), respectively.

Definition 5.2. (**Sampled Gaussian Mechanism (SGM) [37]**). Let f be a function mapping subsets of S to $\mathbb{R}^d$. We define the Sampled Gaussian Mechanism (SGM) parameterized with the sampling rate $0 < q \leq 1$ and the $\sigma > 0$ as

$$SG_{q,\sigma}(S) \triangleq f(\{x : x \in S \text{ is sampled with probability } q\})$$
$$+ \mathcal{N}\left(0, \sigma^2 \mathbb{I}^d\right).$$
(21)

In DPSUR, f is the clipped gradient evaluation on sampled data points $f(\{x_i\}_{i \in B}) = \sum_{i \in B} \bar{g}_t(x_i)$. If $\bar{g}_t$ is obtained by clipping g_t with a gradient norm bound C, then the sensitivity of f is C.

Definition 5.3. (**RDP privacy budget of SGM [37]**). Let $SG_{q,\sigma}$, be the Sampled Gaussian Mechanism for some function f. If f has a sensitivity of 1, $SG_{q,\sigma}$ satisfies (α, R)-RDP if

$$R \leq \frac{1}{\alpha - 1} \ln[max(A_\alpha(q, \sigma), B_\alpha(q, \sigma))],$$
(22)

where

$$\begin{cases} A_\alpha(q, \sigma) \triangleq \mathbb{E}_{z \sim \mu_0} \left[(\mu(z)/\mu_0(z))^\alpha \right] \\ B_\alpha(q, \sigma) \triangleq \mathbb{E}_{z \sim \mu} \left[(\mu_0(z)/\mu(z))^\alpha \right] \end{cases}$$
(23)

with $\mu_0 \triangleq \mathcal{N}\left(0, \sigma^2\right), \mu_1 \triangleq \mathcal{N}\left(1, \sigma^2\right)$ and $\mu \triangleq (1-q)\mu_0 + q\mu_1$.

Furthermore, it holds for $\forall (q, \sigma) \in (0, 1] \times \mathbb{R}^+, A_\alpha(q, \sigma) \geq B_\alpha(q, \sigma)$. Thus, $SG_{q,\sigma}$ satisfies $\left(\alpha, \frac{1}{\alpha-1} \ln \left(A_\alpha(q, \sigma)\right)\right)$-RDP.

Finally, the existing work [37] describes a procedure to compute $A_\alpha(q, \sigma)$ depending on integer α as Eq. (24).

$$A_\alpha = \sum_{k=0}^{\alpha} \left(\begin{array}{c} \alpha \\ k \end{array} \right) (1-q)^{\alpha-k} q^k \exp \left(\frac{k^2 - k}{2\sigma^2} \right)$$
(24)

Lemma 5.4 shows the composition property of RDP mechanisms.

LEMMA 5.4. *(**Composition of RDP [36]**). For two randomized mechanisms f, g such that f is (α, R_1)-RDP and g is (α, R_2)-RDP the composition of f and g which is defined as (X, Y)(a sequence of results), where $X \sim f$ and $Y \sim g$, satisfies $(\alpha, R_1 + R_2) - RDP$*

According to Definitions 5.2 and 5.3, and Lemma 5.4, Theorem 5.1 is proved.

□

5.2 Privacy Analysis of Validation

As Section 4.1 mentioned above, the RDP of validation will be accumulated only when $\widetilde{\Delta E} < \beta \cdot C_v$.

THEOREM 5.5. *After accepting t tests of $\tilde{\Delta} E$, the RDP of the validation phase satisfies:*

$$R_{valid}(\alpha) = \frac{t}{\alpha - 1} \ln \left[\sum_{i=0}^{\alpha} \left(\begin{array}{c} \alpha \\ i \end{array} \right) (1-q)^{\alpha - i} q^i \exp \left(\frac{i^2 - i}{2\sigma_v^2} \right) \right], \text{ (25)}$$

where $q = \frac{B_v}{N}$, σ_v is noise multiplier of the validation phase, and $\alpha > 1$ is the order.

The proof is similar to that of the training phase, so we omit it.

5.3 Overall Privacy Analysis of DPSUR

Since both training and validation phases access the same training set, we need to combine their RDPs sequentially using Lemma 5.4, and then use Lemma 2.7 to convert it to (ϵ, δ)-DP. Therefore, the final privacy loss of DPSUR is as follows:

THEOREM 5.6. (*Privacy loss of DPSUR*). *The privacy loss of DPSUR satisfies:*

$$
\begin{aligned}
(\epsilon, \delta) = &(R_{train}(\alpha) + R_{valid}(\alpha) + \ln((\alpha - 1)/\alpha) \\
&- (\ln \delta + \ln \alpha)/(\alpha - 1), \delta),
\end{aligned}
\tag{26}
$$

where $0 < \delta < 1$, $R_{train}(\alpha)$ is the RDP of training which is computed by Theorem 5.1, and $R_{valid}(\alpha)$ is the RDP of validation which is computed by Theorem 5.5.

5.4 Discussion of Privacy

Our privacy analysis shows that DPSUR strictly adheres to the principles of differential privacy, limiting adversaries to conduct differential attacks solely based on the algorithm's output. However, it is worth noting that DPSUR may be susceptible to interactive side-channel attacks. For instance, a strong adversary (e.g., the hypervisor of a guest OS in the cloud) who has access to DPSUR's internal update/release status could measure the time interval of two adjacent model updates to infer the number of rejections in between and thus cause privacy breaches. To mitigate such threats, we suggest introducing random waiting time for update acceptance cases. Nonetheless, we emphasize that DPSGD is supposed to work in a non-interactive training scenario where the attacker can only access the final output model. Since most real-world machine learning systems are non-interactive during training, the privacy guarantee provided by DPSUR remains sufficient and consistent with other DPSGD variants.

To further validate such privacy guarantee in real-world scenarios, in Section 6.4 we conduct membership inference attacks on the trained models. The experimental results indicate that DPSUR exhibits strong defense against membership inference attacks, thus safeguarding the privacy of training data.

6 EXPERIMENTAL EVALUATION

In this section, we conduct experiments to demonstrate the performance of DPSUR over four real datasets and popular machine learning models. And we perform experiments involving two member inference attacks to show the privacy-preserving effect of DPSUR. All experiments are implemented in Python using PyTorch [45]. Codes to reproduce our experiments are available at https://github.com/JeffffffFu/DPSUR.

6.1 Experimental Setting

6.1.1 Baseline. We compare DPSUR with DPSGD [1] and four state-of-the-art variants, namely DPSGD with important sampling [57], handcrafted features [53], tempered sigmoid activation [41], and adaptive learning rate [31], which we refer to as DPSGD-IS, DPSGD-HF, DPSGD-TS, and DPAGD respectively. Note that we do not compare DPSUR with those approaches that modify the structures of over-parameterized models [13] or the semi-supervised model PATE [39, 40], as they differ significantly from the scope of this work.

6.1.2 Datasets and Models. Experimental evaluation is conducted over three image classification datasets, including MNIST [30], Fashion MNIST (FMNIST) [60], and CIFAR-10 [29], and a movie review dataset IMDB [33].

MNIST contains 60,000 training samples and 10,000 testing samples of handwritten digits, divided into 10 categories with 7,000 grayscale images per category. Each sample consists of a grayscale image of size 28×28 and a corresponding label indicating its category. The model trained using handcrafted features as inputs achieves 99.11% accuracy after 20 epochs in the non-private setting [53].

FMNIST consists of 60,000 training samples and 10,000 testing samples of fashion products categorized into 10 categories, with each category containing 7,000 grayscale images of size 28×28. The dataset also includes labels indicating the category of each image. The model trained using handcrafted features as inputs achieves 90.98% accuracy after 20 epochs in the non-private setting [53].

CIFAR-10 comprises 50,000 training samples and 10,000 testing samples of colored objects categorized into 10 categories. Each category contains 6,000 color images of size 32×32 with three color channels. Additionally, each sample is accompanied by a label indicating the category to which it belongs. The model trained using handcrafted features as inputs achieves 71.12% accuracy after 20 epochs in the non-private setting [53].

IMDb consists of 50,000 reviews of movies, each review encoded as a list of word indexes and labeled with an obvious bias towards either positive or negative sentiment. The dataset is divided into a training set of 25,000 reviews and a test set of 25,000 reviews. In the non-private setting, the model trained using cross-entropy loss function, Adam optimizer, and an expected batch size of 32 achieves an accuracy of 79.97% after 20 epochs.

We apply the same convolutional neural network architecture as [41, 53, 57] to three image datasets, i.e., MNIST, FMNIST, and CIFAR-10. Additionally, we used a same five-layer recurrent neural network as in [57] for the IMDB dataset. We use the categorical cross-entropy loss function for all datasets. The details of model architectures are presented in Appendix B.

Table 1: Noise multiplier for validation σ_v

Dataset	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$
MNIST	1.3	1.0	0.9	0.8
FMNIST	1.3	1.3	0.8	0.8
CIFAR-10	1.3	1.3	1.1	1.1
IMDB	1.3	1.2	1.0	0.9

6.1.3 Parameter Settings. In our experiments, we set the privacy budget ϵ from 1 to 4 for each dataset while fixing $\delta = 10^{-5}$. For image datasets, we user the SGD optimizer with a momentum parameter set to 0.9; for the IMDB dataset, we employ the Adam optimizer whose parameters are the same as [11].

During the DPSGD phase, for the three image datasets, we adopt the best parameters recommended in [53]. Specifically, we fine-tune the noise multiplier σ_t for various ϵ, following the approach outlined in [53, 57]. This fine-tuning process is a common practice in all privacy-preserving machine learning methods, and it does not incur any privacy loss. For the IMDB dataset, we enumerate different values and choose the best for each parameter since the competitor method [57] does nshiot specify them.

Based on our analysis in Section 3.2 and 3.3, we set the clipping bound $C_v = 0.001$ for ΔE and acceptance parameter $\beta = -1$ to all privacy budgets ϵ and datasets. In addition, for MNIST, FMNIST and CIFAR-10, we set the batch size of validation set $B_v = 256$. While IMDB, which have fewer training samples, we set the batch size of validation set $B_v = 128$. The noise multiplier for validation σ_v ranges from 0.8 to 1.3 for all datasets and privacy budgets, as shown in Table 1. Intuitively, when the privacy budget is small, we increase σ_v to add more iteration rounds.

6.2 Overall Performance

Table 2 shows the classification accuracies of DPSUR and five competitive methods. It is noteworthy that DPSUR consistently outperforms all competitors across all datasets and privacy budgets, except for a less eminent advantage on the MNIST dataset where the accuracy of [57] already approaches that of the non-private setting, leaving little room for further improvement. For the other three datasets, the classification accuracy of DPSUR is at least 1% higher than the second best, which shows a huge improvement over DPSGD.

Notably, DPSUR performs almost as well as in the non-private setting at $\epsilon = 4$ in three image datasets. The superior performance of the DPSUR is attributed to our objective of selecting model updates to minimize the loss function. Moreover, We derive the RDP for the selective Gaussian mechanism, which allows us to reduce the consumption of privacy loss. In particular, on the CIFAR-10 dataset, we observe that DPSUR even outperforms non-private results when $\epsilon = 4$. This is because moderate noise in SGD sometimes helps the neural network escape from local minima [22].

6.3 Impact of Various Parameters

In this subsection, we study the impact of various parameters of DPSUR, including the learning rate, the noise multiplier of validation, the cilpping bound of loss, and the threshold parameter. Due to space limitation, we only show the results of CIFAR-10 dataset. In all experiments, if not specified, we use the SGD optimizer with the momentum 0.9, and set the learning rate $\eta = 4.0$, batch size for training $B_t = 8192$, noise multiplier for training $\sigma_t = 5.67$, batch size for validation $B_v = 128$, noise multiplier for validation $\sigma_v = 1.1$, clipping bound for validation $C_v = 0.001$, the threshold parameter $\beta = -1$, and privacy budget $(3, 10^{-5})$.

Learning rate η. As plotted in Figure 9a, the highest accuracy achieved is 70.83% when using a learning rate of $\eta = 4$. If the learning rate is larger than 4, the accuracy starts to decrease. This is because a large learning rate may cause slow convergence to the trained model. However, thanks to selective update, DPSUR becomes adaptive to different learning rates, and the accuracy remains stable at 70.02% when using a extremely high learning rate $\eta = 7$.

Noise multiplier for validation σ_v. A large σ_v can save privacy budget running more rounds, which degrades the quality of the accepted model. On the contrary, a small σ_v can guarantee accepted model quality, but consumes more privacy budget. According to Figure 9b, we find that the test accuracy of DPSUR is

quite stable at around 70.3% when σ_v increases from 0.8 to 1.3, and the highest accuracy is 70.83% when $\sigma_v = 1.1$.

Clipping bound for validation C_v. As aforementioned, using a sufficiently small clipping bound C_v can discretize the difference of the loss ΔE. With a shallow model training on CIFAR-10, loss values ranging from 0.01 to 0.0001 are all considered sufficiently small, a smaller C_v (e.g. $1e - 05$) than the front does not bring performance gains. As shown in Figure 9c, the accuracy for the cases of $C_v \in [0.01, 1e - 05]$ achieves slightly better performance. However, when C_v is set to 1 or 10, we observe a significant decline in performance, which is consistent with our analysis that setting a large C_v does not provide any performance benefits.

Threshold parameter β. The acceptance probability is influenced by the parameter β, with smaller values leading to a decreasing probability of accepting both low-quality and high-quality models. As shown in Figure 9d, the best performance is achieved when $\beta = -1.0$. This is consistent with our analysis in Section 3.3 that a smaller β leads to a higher rejection probability for low-quality updates, thereby guiding the model towards the correct direction during iterations. It is worth noting that setting β too small does not gain more benefits, as a very small β causes the model to reject almost all high-quality and low-quality solutions, contributing nothing to the model convergence.

Batch size of validation set B_v. A small B_v can help conserve the privacy budget, enabling more rounds of computation. However, this can result in a decrease in the quality of the accepted model. Conversely, a larger value of B_v can ensure better model quality but consumes a greater portion of the privacy budget. As shown in Figure 9e, we observe that as B_v increases from 32 to 256, the test accuracy of DPSUR improves from 69.10% to 70.83%, but it declines to 68.36% when $B_v = 1024$.

6.4 Resilience Against Member Inference Attacks

Differential privacy protection is naturally resistant to membership inference attacks. To empirically verify if DPSUR achieves the same privacy guarantee as DPSGD, we conduct membership inference attacks on models trained on FMNIST and CIFAR-10, where their models are trained from DPSUR and DPSGD algorithms, respectively.
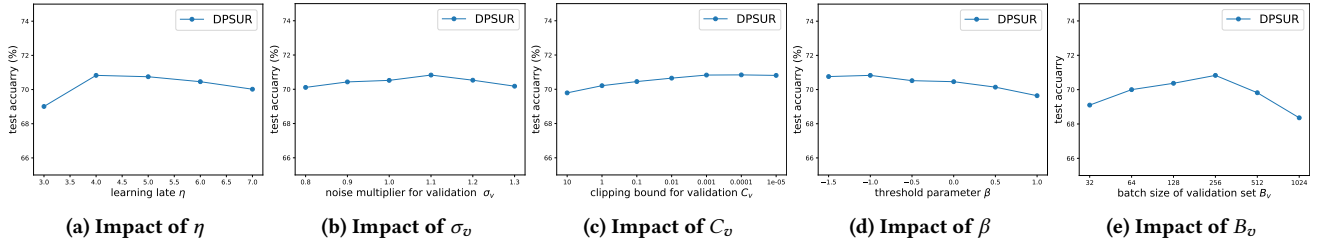
6.4.1 Attack overview. We adopt two membership inference attacks, Black-Box/Shadow [46] and White-Box/Partial [38], which are the SOTA methods in membership inference attack to our knowledge.

Black-Box/Shadow. In the Black-Box/Shadow attack scenario, the adversary has a shadow auxiliary dataset. The dataset is divided into two parts, with one part used to train a shadow model for the same task. The shadow model is then queried using the entire shadow dataset. For each query sample, the shadow model provides its posterior probability and predicted label. The adversary labels the sample as a member if it belongs to the training set of the shadow model, otherwise, it is labeled as a non-member. Using this labeled dataset, the adversary trains an attack model, which serves as a binary classifier to distinguish between members and non-members. To determine if a sample belongs to the target model's training dataset, it is inputted into the target model for prediction.

Since the scattering network used in DPSGD-HF [53] is not applicable to natural language processing, we omit the results of this method on the IMDb dataset.

Table 2: Results of classification accuracy

Dataset	Method	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	non-private
MNIST	DPSUR	**97.93%**	**98.70%**	**98.88%**	**98.95%**	
(Image Dataset)	DPIS [57]	97.79%	98.51%	98.62%	98.78%	
	DPSGD-HF [53]	97.78%	98.39%	98.32%	98.56%	99.11%
	DPSGD-TS [41]	97.06%	97.87%	98.22%	98.51%	
	DPAGD [31]	95.91%	97.30%	97.52%	97.83%	
	DPSGD [1]	95.11%	96.10%	96.82%	97.25%	
FMNIST	DPSUR	**88.38%**	**89.34%**	**89.71%**	**90.18%**	
(Image Dataset)	DPIS [57]	86.25%	88.24%	88.82%	89.21%	
	DPSGD-HF [53]	85.54%	87.96%	89.01%	89.06%	90.98%
	DPSGD-TS [41]	83.63%	85.33%	86.29%	86.86%	
	DPAGD [31]	81.26%	84.50%	86.04%	86.78%	
	DPSGD [1]	80.25%	82.63%	84.72%	85.40%	
CIFAR-10	DPSUR	**64.41%**	**69.40%**	**70.83%**	**71.45%**	
(Image Dataset)	DPIS [57]	63.23%	67.94%	69.63%	70.55%	
	DPSGD-HF [53]	63.15%	66.55%	69.35%	70.28%	71.12%
	DPSGD-TS [41]	51.52%	56.78%	60.42%	61.75%	
	DPAGD [31]	45.78%	53.30%	56.21%	60.31%	
	DPSGD [1]	46.03%	51.33%	54.67%	58.89%	
IMDb	DPSUR	**66.50%**	**71.02%**	**72.16%**	**74.14%**	
(Text Dataset)	DPIS [57]	63.56%	66.11%	68.49%	70.12%	
	DPSGD-TS [41]	65.08%	68.34%	70.10%	70.85%	79.97%
	DPAGD [31]	58.72%	63.48%	64.59%	66.01%	
	DPSGD [1]	64.13%	68.55%	70.41%	71.57%	

(a) Impact of η (b) Impact of σ_v (c) Impact of C_v (d) Impact of β (e) Impact of B_v

Figure 9: The impact of different parameters on the test accuracy in CIFAR-10.

The resulting posterior probability and predicted label (converted into a binary indicator of prediction correctness) are then fed into the attack model.

White-Box/Partial. In the White-Box/Partial attack scenario, the adversary has partial training dataset as the auxiliary dataset. One advantage in the White-Box/Partial attack is that the adversary has access to the target model. This allows adversary to utilize various resources, including gradients with respect to model parameters, embeddings from intermediate layers, classification loss, as well as posterior probabilities and labels of the target samples.

6.4.2 Attack setting. For each dataset, we randomly split it into four subsets: the target training dataset, target testing dataset, shadow training dataset, and shadow testing dataset. The ratio of the sample sizes in each subset is 2:1:2:1. Our target model and training parameters are consistent with those described above.

6.4.3 Results. Table 3 and Table 4 report the accuracy of two inference attacks against target models protected by DPSUR and DPSGD on FMNIST and CIFAR-10, respectively. We observe that member inference attacks are quite effective against the non-private methods (non-dp), especially on CIFRA-10. As for the two DP algorithms, the attack accuracy drops from 0.58 to around 0.50 on the FMNIST, and drops from 0.73 to around 0.50 on the CIFAR-10, which almost equals to random guess. It's noteworthy that the attack performance on FMNIST is consistently poor, as models trained on FMNIST generalize well on non-member data samples [48]. These results show that the model under DP protection can defend very well against membership inference attacks, and our DPSUR algorithm can provide the same level of privacy protection as DPSGD.

Table 3: Accuracy of Member Inference Attack on FMNIST

Attack	Algorithm	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	non-private
Black-Box/Shadow	DPSUR	0.498	0.500	0.503	0.506	0.582
	DPSGD	0.498	0.503	0.493	0.494	
White-Box/Partial	DPSUR	0.499	0.504	0.501	0.502	0.584
	DPSGD	0.501	0.502	0.502	0.505	

Table 4: Accuracy of Member Inference Attack on CIFAR-10

Attack	Algorithm	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	non-private
Black-Box/Shadow	DPSUR	0.495	0.498	0.503	0.504	0.732
	DPSGD	0.504	0.505	0.504	0.505	
White-Box/Partial	DPSUR	0.499	0.501	0.502	0.503	0.743
	DPSGD	0.500	0.501	0.501	0.503	

7 RELATED WORK

Privacy-preserving model training was first proposed in [5, 50]. Subsequently, Abadi et al. [1] proposed a generalized algorithm, DPSGD, for deep learning with differential privacy, and since then many works aimed at improving DPSGD from different aspects.

Gradient clipping. At each iteration of training, Zhang et al. [68] used public data to obtain an approximate bound on gradient norm and clip the gradients at this approximate bound. The work in [55] proposed adaptive clipping in each layer of the neural network. Andrew et al. [2] designed a method for adaptively tuning the clipping threshold to track a given quantile of the update norm distribution during training, especially in federated learning. Venkatadheeraj et al. [44] proposed AdaCliP, which using coordinate-wise adaptive clipping of the gradient. However, a recent work [62] has shown that by redefining the clipping equation as $Clip_C(g) = C/||g||_2$, clipping is actually equivalent to normalization by setting the clipping bound small enough.

Our paper does not employ any adaptive clipping technique during the training phase. Instead, our minimal clipping is orthogonal to theirs as it is for threshold evaluation in the validation phase, not the training phase.

Gaussian noise. The work of [42] implemented adaptive noise addition using a hierarchical correlation propagation protocol approach, adding a small amount of noise to features with high correlation to the model's output. Balle and Wang [4] introduced an optimized Gaussian mechanism that directly calibrates variance using the Gaussian cumulative density function instead of relying on a tail-bound approximation. Their work is orthogonal to ours and can be incorporated ours as the underlying perturbation mechanism. Lee et al. [31] selected the best learning rate from a candidate set based on model evaluation and implemented adaptive privacy budget allocation in each round of DPSGD training. While both our work and theirs involve adaptive DPSGD, there are three significant distinctions. First, their adaptiveness is from adaptive learning rates, which modifies the gradient descent's step size but not its direction. Second, the DP privacy guarantee is different. Ours relies on differences in loss values before and after iterations,

whereas theirs adopts NoisyMax[18] method on loss values from a variety of models obtained through different learning rates in a single iteration. In contrast, we aim to ensure each gradient descent moves in the desired direction through resampling and re-noising. Third, we do not perform adaptive privacy budget allocation but instead introduce selective release techniques to preserve the privacy budget.Further, Xu et al. [61] employed the Root Mean Square Prop (RMSProp) gradient descent technique to adaptively add noise to coordinates of the gradient. Since then, many works [23, 65, 71] have focused on reducing the dimensionality of the model during training to reduce the impact of noise on the overall model.

Poisson sampling. Wei et al. [57] first explored the problem of bias due to Poisson sampling in DPSGD and proposed DPIS, which weights the importance sampling by the gradient norm of the sample. Our algorithm mitigates the impact of Poisson sampling on convergence speed through the process of resampling.

Models, pre-processing and parameter tuning. Papernot et al. [41] found that using a family of bounded activation functions (tempered sigmoids) instead of the unbounded activation function ReLU in DPSGD can achieve good performance. Tramer et al. [53] used Scattering Network to traverse the image in advance to extract features before DPSGD training. Soham et al. [13] combined careful hyper-parameter tuning with group normalization and weight standardization to yield remarkable performance benefits. These works are orthogonal to our work.

Privacy accounting. Abadi et al. [1] proposed a method called the Moments Accountant (MA) for giving an upper bound the privacy curve of a composition of DPSGD. The Moments Accountant was subsumed into the framework of Renyi Differential Privacy (RDP) introduced by [36]. Bu et al. [7] introduced the notion of Gaussian Differential Privacy (GDP) base hypothesis test. There also exits other variants of DP, for example Concentrated DP (CDP) [8] and zero Concentrated-DP [8]. These variants are tailored for specific scenarios and can be converted into one another under certain conditions. Our primary focus is on (ϵ, δ)-differential privacy, as it is the most prevalent and widely adopted in both academic literature and practical applications. Besides, many works [14, 15, 63, 64, 69] based on local differential privacy focus on ϵ- differential privacy.

8 CONCLUSION

We propose DPSUR, a differentially private scheme for deep learning based on selective update and release. Our scheme utilizes the validation test to select appropriate model updates in each iteration, thereby speeding up model convergence and enhancing utility. To reduce the injected Gaussian noise, we incorporate a clipping strategy and a threshold mechanism for gradient selection in each iteration. Furthermore, we apply the Gaussian mechanism with selective release to reduce privacy budget consumption across iterations. We conduct a comprehensive privacy analysis of our approach using RDP and validate our scheme through extensive experiments. The results indicate that DPSUR significantly outperforms state-of-the-art solutions in terms of model utility and downstream tasks. Our scheme is widely applicable to various neural networks, and can serve as a flexible optimizer for new DPSGD variants. For future work, we plan to extend DPSUR to larger models and datasets and theoretically analyze its convergence speed.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. 2021. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems* 34 (2021), 17455–17466.

[3] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. 2020. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2496–2506.

[4] Borja Balle and Yu-Xiang Wang. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*. PMLR, 394–403.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*. IEEE, 464–473.

[6] Léon Bottou. 2009. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science, Paris*, Vol. 8. Citeseer, 2624–2633.

[7] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. 2020. Deep learning with gaussian differential privacy. *Harvard data science review* 2020, 23 (2020), 10–1162.

[8] Mark Bun and Thomas Steinke. 2016. *Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds*. 635–658. https://doi.org/10.1007/978-3-662-53641-4_24

[9] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*. 267–284.

[10] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2287–2295.

[11] François Chollet et al. 2015. Keras. https://keras.io.

[12] Rachel Cummings and Deven Desai. 2018. The role of differential privacy in gdpr compliance. In *FAT'18: Proceedings of the Conference on Fairness, Accountability, and Transparency*. 20.

[13] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. 2022. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650* (2022).

[14] Rong Du, Qingqing Ye, Yue Fu, Haibo Hu, Jin Li, Chengfang Fang, and Jie Shi. 2023. Differential Aggregation against General Colluding Attackers. In *Proceedings of the IEEE International Conference on Data Engineering*.

[15] Jiawei Duan, Qingqing Ye, and Haibo Hu. 2022. Utility analysis and enhancement of LDP mechanisms in high-dimensional space. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 407–419.

[16] Cynthia Dwork. 2011. A firm foundation for private data analysis. *Commun. ACM* 54, 1 (2011), 86–95.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *TCS* 9 (2014), 211–407. Issue 3-4.

[19] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[20] Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 954–959.

[21] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[22] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. 2015. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*. PMLR, 797–842.

[23] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. 2022. Mixed differential privacy in computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8376–8386.

[24] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo A Parrilo. 2021. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming* 186 (2021), 49–84.

[25] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*. 1895–1912.

[26] Angelos Katharopoulos and François Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*. PMLR, 2525–2534.

[27] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

[28] Antti Koskela and Antti Honkela. 2018. Learning rate adaptation for differentially private stochastic gradient descent.

[29] A. Krizhevsky and G. Hinton. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto* (2009).

[30] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[31] Jaewoo Lee and Daniel Kifer. 2018. 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*. 1656–1665.

[32] Alexander Selvikvåg Lundervold and Arvid Lundervold. 2019. An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift für Medizinische Physik* 29, 2 (2019), 102–127.

[33] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. The Association for Computer Linguistics, 142–150.

[34] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.

[35] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.

[36] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[37] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv: Learning* (2019).

[38] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[39] Nicolas Papernot, Martın Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. SEMI-SUPERVISED KNOWLEDGE TRANSFER FOR DEEP LEARNING FROM PRIVATE TRAINING DATA. *stat* 1050 (2017), 3.

[40] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908* (2018).

[41] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. 2021. Tempered sigmoid activations for deep learning with differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9312–9321.

[42] NhatHai Phan, Xintao Wu, Han Hu, and Dejing Dou. 2017. Adaptive laplace mechanism: Differential privacy preservation in deep learning. In *2017 IEEE international conference on data mining (ICDM)*. IEEE, 385–394.

[43] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2017. Privacy-preserving deep learning: Revisited and enhanced. In *Applications and Techniques in Information Security: 8th International Conference, ATIS 2017, Auckland, New Zealand, July 6–7, 2017, Proceedings*. Springer, 100–110.

[44] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X Yu, Sashank J Reddi, and Sanjiv Kumar. 2019. AdaCliP: Adaptive clipping for private SGD. *arXiv preprint arXiv:1908.07643* (2019).

[45] Automatic Differentiation In Pytorch. 2018. Pytorch.

[46] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed Systems Security (NDSS) Symposium 2019*.

[47] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[48] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[49] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine learning models that remember too much. (2017), 587–601.

[50] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*. IEEE, 245–248.

[51] Pierre Stock, Igor Shilov, Ilya Mironov, and Alexandre Sablayrolles. 2022. Defending against Reconstruction Attacks with Rényi Differential Privacy. *arXiv e-prints* (2022), arXiv–2202.

[52] Kenji Suzuki. 2017. Overview of deep learning in medical imaging. *Radiological physics and technology* 10, 3 (2017), 257–273.

[53] Florian Tramer and Dan Boneh. 2020. Differentially Private Learning Needs Better Features (or Much More Data). In *International Conference on Learning Representations*.

[54] Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

[55] KoenS.vander Veen, Ruben Seggers, Peter Bloem, and Giorgio Patrini. 2018. Three Tools for Practical Differential Privacy.

[56] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2512–2520.

[57] Jianxin Wei, Ergute Bao, Xiaokui Xiao, and Yin Yang. 2022. Dpis: An enhanced mechanism for differentially private sgd with importance sampling. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2885–2899.

[58] Shaomei Wu, Hermes Pique, and Jeffrey Wieland. 2016. Using artificial intelligence to help blind people 'see'facebook.

[59] Liyao Xiang, Jingbo Yang, and Baochun Li. 2019. Differentially-private deep learning from an optimization perspective. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 559–567.

[60] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints* (2017), arXiv–1708.

[61] Zhiying Xu, Shuyu Shi, Alex X Liu, Jun Zhao, and Lin Chen. 2020. An adaptive and fast convergent approach to differentially private deep learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1867–1876.

[62] Xiaodong Yang, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2022. Normalized/clipped sgd with perturbation for differentially private non-convex optimization. *arXiv preprint arXiv:2206.13033* (2022).

[63] Qingqing Ye, Haibo Hu, Kai Huang, Man Ho Au, and Qiao Xue. 2023. Stateful switch: Optimized time series release with local differential privacy. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[64] Qingqing Ye, Haibo Hu, Xiaofeng Meng, Huadi Zheng, Kai Huang, Chengfang Fang, and Jie Shi. 2023. PrivKVM*: Revisiting Key-Value Statistics Estimation With Local Differential Privacy. *IEEE Transactions on Dependable and Secure Computing* (Jan 2023), 17–35. https://doi.org/10.1109/tdsc.2021.3107512

[65] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2020. Do not Let Privacy Overbill Utility: Gradient Embedding Perturbation for Private Learning. In *International Conference on Learning Representations*.

[66] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially private model publishing for deep learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 332–349.

[67] Jianyi Zhang, Yang Zhao, and Changyou Chen. 2020. Variance reduction in stochastic particle-optimization sampling. In *International Conference on Machine Learning*. PMLR, 11307–11316.

[68] Xinyang Zhang, Shouling Ji, and Ting Wang. 2018. Differentially Private Releasing via Deep Generative Model (Technical Report). *arXiv e-prints* (2018), arXiv–1801.

[69] Yuemin Zhang, Qingqing Ye, Rui Chen, Haibo Hu, and Qilong Han. 2023. Trajectory Data Collection with Local Differential Privacy. *Proceedings of the VLDB Endowment* 16, 10 (2023), 2591–2604.

[70] S Kevin Zhou, Hayit Greenspan, Christos Davatzikos, James S Duncan, Bram Van Ginneken, Anant Madabhushi, Jerry L Prince, Daniel Rueckert, and Ronald M Summers. 2021. A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises. *Proc. IEEE* 109, 5 (2021), 820–838.

[71] Yingxue Zhou, Steven Wu, and Arindam Banerjee. 2020. Bypassing the Ambient Dimension: Private SGD with Gradient Subspace Identification. In *International Conference on Learning Representations*.

[72] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019).

[73] Tianqing Zhu, Gang Li, Wanlei Zhou, and S Yu Philip. 2017. *Differential Privacy and Applications*. Vol. 69. Springer.

A RDP OF TWO TRUNCATED NORMAL DISTRIBUTIONS

$$D_\alpha(f(x; \mu, \mu\sigma, a, b) || f(x; 0, \mu\sigma, a, b))$$

$$= \frac{1}{\alpha - 1} \cdot \log \int_a^b \frac{[f(x; \mu, \mu\sigma, a, b)]^\alpha}{[f(x; 0, \mu\sigma, a, b)]^{\alpha-1}} dx$$

$$= \frac{1}{\alpha - 1} \cdot \log\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \int_a^b \frac{1}{\mu\sigma\sqrt{2\pi}}$$

$$\cdot \exp(\frac{-\alpha(x-\mu)^2}{2\mu^2\sigma^2}) \cdot \exp(\frac{-(1-\alpha)x^2}{2\mu^2\sigma^2}) dx\}$$

$$= \frac{1}{\alpha - 1} \cdot \log\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\mu\sigma\sqrt{2\pi}} \int_a^b \exp\left[\left(-x^2 + 2\alpha\mu x - \alpha\mu^2\right) \Big/ \left(2\mu^2\sigma^2\right)\right] dx\}$$

$$= \frac{1}{\alpha - 1} \cdot \log\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\sqrt{\pi}} \int_a^b exp(\frac{\alpha(\alpha-1)}{2\sigma^2})$$

$$\cdot exp(-(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})^2) d(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})\}$$

$$= \frac{1}{\alpha - 1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \log[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\sqrt{\pi}}$$

$$\cdot \int_a^b exp(-(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})^2) d(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})]\}$$

$$= \frac{1}{\alpha - 1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \log[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha}$$

$$\cdot \int_{\frac{a-\alpha\mu}{\mu\alpha}}^{\frac{b-\alpha\mu}{\mu\alpha}} \frac{1}{\sqrt{2\pi}} \cdot exp(-\frac{x^2}{2}) d(x)]\}$$

$$= \frac{1}{\alpha - 1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \log[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha}$$

$$\cdot (\Phi(\frac{b-\alpha\mu}{\mu\sigma}) - \Phi(\frac{a-\alpha\mu}{\mu\sigma}))]\}$$

$$= \frac{\alpha}{2\sigma^2} + \frac{1}{\alpha - 1} \cdot \log\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot [\Phi(\frac{b-\alpha\mu}{\mu\sigma}) - \Phi(\frac{a-\alpha\mu}{\mu\sigma})]\},$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt.$

Table 5: MNIST and FMNIST model architecture

Layer	Parameters
Convolution	16 filters of 8×8, stride 2,padding 2
Max-Pooling	2×2, stride 1
Convolution	32 filters of 4×4, stride 2, padding 0
Max-Pooling	2×2, stride 1
Fully connected	32 units
Fully connected	10 units

Table 6: CIFAR-10 model architecture

Layer	Parameters
Convolution×2	32 filters of 3×3, stride 1, padding 1
Max-Pooling	2×2, stride 2
Convolution×2	64 filters of 3×3, stride 1, padding 1
Max-Pooling	2×2, stride 2
Convolution×2	128 filters of 3×3, stride 1, padding 1
Max-Pooling	2×2, stride 2
Fully connected	128 units
Fully connected	10 units

Table 7: IMDb model architecture.

Layer	Parameters
Embedding	100 units
Fully connected	32 units
Bidirectional LSTM	32 units
Fully connected	16 units
Fully connected	2 units

$$D_\alpha(f(x;\mu,\mu\sigma,a,b)||f(x;0,\mu\sigma,a,b))$$

$$= \frac{1}{\alpha-1} \cdot \ln \int_a^b \frac{[f(x;\mu,\mu\sigma,a,b)]^\alpha}{[f(x;0,\mu\sigma,a,b)]^{\alpha-1}} \, dx$$

$$= \frac{1}{\alpha-1} \cdot \ln\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \int_a^b \frac{1}{\mu\sigma\sqrt{2\pi}}$$

$$\cdot \exp(\frac{-\alpha(x-\mu)^2}{2\mu^2\sigma^2}) \cdot \exp(\frac{-(1-\alpha)x^2}{2\mu^2\sigma^2}) dx\}$$

$$= \frac{1}{\alpha-1} \cdot \ln\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\mu\sigma\sqrt{2\pi}} \int_a^b \exp\left[\left(-x^2 + 2\alpha\mu x - \alpha\mu^2 \right) / \left(2\mu^2\sigma^2 \right) \right] dx\}$$

$$= \frac{1}{\alpha-1} \cdot \ln\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\sqrt{\pi}} \int_a^b exp(\frac{\alpha(\alpha-1)}{2\sigma^2}) \cdot exp(-(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})^2) d(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})\}$$

$$= \frac{1}{\alpha-1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \ln[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \frac{1}{\sqrt{\pi}} \cdot \int_a^b exp(-(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})^2) d(\frac{x-\alpha\mu}{\sqrt{2}\mu\sigma})]\}$$

$$= \frac{1}{\alpha-1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \ln[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot \int_{\frac{a-\alpha\mu}{\mu\alpha}}^{\frac{b-\alpha\mu}{\mu\alpha}} \frac{1}{\sqrt{2\pi}} \cdot exp(-\frac{x^2}{2}) d(x)]\}$$

$$= \frac{1}{\alpha-1} \cdot \{ \frac{\alpha(\alpha-1)}{2\sigma^2} + \ln[\frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot (\Phi(\frac{b-\alpha\mu}{\mu\sigma}) - \Phi(\frac{a-\alpha\mu}{\mu\sigma}))]\}$$

$$= \frac{\alpha}{2\sigma^2} + \frac{1}{\alpha-1} \cdot \ln\{ \frac{(\Phi(\frac{b}{\mu\sigma}) - \Phi(\frac{a}{\mu\sigma}))^{\alpha-1}}{(\Phi(\frac{b-\mu}{\mu\sigma}) - \Phi(\frac{a-\mu}{\mu\sigma}))^\alpha} \cdot [\Phi(\frac{b-\alpha\mu}{\mu\sigma}) - \Phi(\frac{a-\alpha\mu}{\mu\sigma})]\},$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt.$

B MODEL ARCHITECTURES