SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2026, Vol. 69, Iss. 4, 142106:1–142106:13
https://doi.org/10.1007/s11432-025-4580-x

An epoch-weighted privacy budget allocation framework for fine-tuning large language models

Peisheng ZHANG[1], Naiyu WANG[1], Longfei WU[2], Liehuang ZHU[3] & Zhitao GUAN[1*]

[1]*School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China*
[2]*Department of Mathematics and Computer Science, Fayetteville State University, Fayetteville NC 28301, USA*
[3]*School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China*

Received 20 March 2025/Revised 3 July 2025/Accepted 4 September 2025/Published online 5 March 2026

Abstract Fine-tuning large language models (LLMs) has demonstrated outstanding performance across various downstream tasks. However, fine-tuning LLMs on private training data poses significant privacy risks, as adversaries can employ various attack methods to extract sensitive information during training. Existing differential privacy frameworks for LLMs often rely on a uniform privacy budget allocation strategy, which neglects the increasing sensitivity of model parameters to noise perturbations as optimization progresses. To address these issues, we propose an epoch-weighted privacy budget allocation framework for fine-tuning LLMs (EW-FT), which incorporates epoch factors into the privacy budget allocation algorithm and utilizes stacked autoencoders to mitigate the curse of dimensionality. By injecting less noise into the forward hidden embeddings of more sensitive fine-tuning epochs, EW-FT achieves more targeted local differential privacy perturbations. Extensive experiments on three downstream tasks demonstrate that, while maintaining the same level of privacy protection, our EW-FT achieves higher model accuracy compared with state-of-the-art techniques.

Keywords differential privacy, large language model, fine-tuning, utility improvement, distributed scenario

Citation Zhang P S, Wang N Y, Wu L F, et al. An epoch-weighted privacy budget allocation framework for fine-tuning large language models. Sci China Inf Sci, 2026, 69(4): 142106, https://doi.org/10.1007/s11432-025-4580-x

1 Introduction

With continuous advancements in artificial intelligence technology, large language models (LLMs), such as bidirectional encoder representations from transformers (BERT) [1], have become a research hotspot in natural language processing [2, 3]. LLMs typically undergo unsupervised pre-training on large-scale text corpora and are then fine-tuned for specific downstream tasks [4], including sentiment analysis [5], text generation [6], and reading comprehension [7]. However, fine-tuning LLMs on private training data introduces significant privacy risks, as malicious actors can exploit various attacks—such as membership inference attacks [8], extraction attacks [9], and attribute inference attacks [10]—to extract sensitive information during model training.

To mitigate these risks, there has been significant interest in developing data privacy solutions for LLMs. Differential privacy (DP) [11] has become a widely recognized approach for privacy protection in LLMs. The classical differentially private stochastic gradient descent (DP-SGD) [12] protects model gradients through per-sample clipping, but suffers from high memory consumption and degraded model performance. To address these challenges, researchers have explored various solutions. Yu et al. [13] proposed a reparameterization method that computes gradient projections instead of directly calculating gradients, achieving performance comparable to non-private baselines. They further introduced parameter-efficient fine-tuning methods, which simplify low-rank decomposition and projection steps to reduce computational overhead [14]. Li et al. [15] developed a memory-saving technique called ghost clipping, which employs an additional backward pass to compute the gradient norms and clipping factors without instantiating the gradients for individual samples. Bu et al. [16] proposed the BK algorithm, which optimizes ghost clipping by calculating clipped gradients with a single backward pass, greatly reducing the training time.

Although these studies alleviate the computational resource consumption in DP-SGD to some extent, they all possess common limitations.

* Corresponding author (email: guan@ncepu.edu.cn)

(1) Non-local privacy paradigm. These methods are designed to provide central differential privacy (CDP) during training. While secure aggregation [17] allows for a distributed implementation, it inevitably introduces additional costs and trust assumptions, as service providers retain access to sensitive data and model parameters.

(2) Limitations of privacy budget allocation methods. These methods typically allocate the same privacy budget for each training round. However, different stages of model training have varying impacts on the convergence of model parameters. Simply injecting an equal magnitude of noise in all iterations is overly "rigid" and can reduce model performance since the model's sensitivity to noise perturbations increases as training progresses towards optimization convergence [18].

(3) Curse of dimensionality [13]. Due to the constraints of privacy budget ε, the magnitude of noise increases with feature dimensions to maintain overall privacy guarantees. This excessive noise can overwhelm meaningful features, leading to poor model performance. Additionally, perturbations in high-dimensional data can distort gradient directions, making it difficult for the model to converge [19].

We have designed three targeted solutions to resolve the above limitations.

(1) Forward-pass privacy protection. Instead of protecting gradients, we apply differential privacy by clipping hidden embeddings and adding Gaussian noise during the forward pass. This method can achieve comparable privacy guarantees to DP-SGD and is well-suited for distributed privacy protection scenarios [20].

(2) Epoch-weighted privacy budget allocation. In the early fine-tuning stage, the pre-trained layer weights are far from the local optimum, hence requiring less precision and allowing for greater noise injection. As training progresses and parameters approach their optimal values, the gradient magnitudes decrease and tend to be overwhelmed by evenly allocated noise [21, 22]. To solve this issue, we designed an epoch-weighted privacy budget allocation algorithm, which allocates more budget to the later stages for better fine-tuning when gradient sensitivity increases.

(3) Dimensionality reduction via stacked autoencoders. To address dimensionality reduction and data reconstruction in the local differential privacy paradigm, we introduced stacked autoencoders [23]—a powerful nonlinear feature reduction model—to obtain a high-level low-dimensional representation of hidden embeddings. These representations are reconstructed after applying differential privacy mechanisms to mitigate the curse of dimensionality.

Integrating the three targeted solutions above, we present an epoch-weighted privacy budget allocation framework for fine-tuning LLMs (EW-FT) to provide reliable, efficient, and minimally disruptive local differential privacy during model fine-tuning.

We summarize the main contributions as follows.

(1) We propose the epoch-weighted privacy budget allocation (EWPBA) algorithm, which divides the fine-tuning process into two stages and dynamically adjusts noise injection across epochs.

(2) We introduce stacked autoencoders to mitigate the curse of dimensionality while providing users with independent control over noise scale and autoencoder depth.

(3) We conduct extensive experiments on three datasets: SST-2 (Stanford Sentiment Treebank), QNLI (Question Natural Language Inference), and IMDB (Internet Movie Database). The results demonstrate that our EW-FT framework achieves better model accuracy than state-of-the-art techniques while maintaining the same level of privacy protection.

2 Related work

2.1 Privacy budget allocation

Privacy budget allocation, which is crucial for balancing privacy protection and data utility in differential privacy, has evolved from early fixed-budget strategies to dynamic and adaptive methods.

In the field of deep learning, Hong et al. [21] introduced a dynamic noise magnitude adjustment strategy for differentially private gradient descent. This method adjusts the noise level based on each gradient step's effect on model performance. Noise is reduced during critical optimization phases to minimize utility loss, and is increased during less critical phases to enhance privacy protection. Chen et al. [18] proposed a privacy budget allocation method based on layer-wise relevance propagation (LRP), which calculates the relevance of input features to the model's output and allocates privacy budgets accordingly. Specifically, features with higher relevance are assigned less noise to preserve utility, while less relevant features receive more noise to enhance privacy protection. Wang et al. [24] developed Rap-FT, a framework for balancing differential privacy and utility in parameter-efficient fine-tuning (PEFT) for language models. Unlike [18], Rap-FT is tailored for PEFT and incorporates an enhanced

LRP process to analyze the relevance of trainable parameters. This allows for adaptive gradient perturbation that mitigates the adverse impacts of differential privacy noise on model utility, while maintaining strong privacy guarantees. Zhang et al. [25] presented an entropy-based adaptive differential privacy mechanism for deep neural networks. Their method adjusts noise levels based on the information gain of each neuron during backpropagation. Neurons with higher information gain receive less noise, while those with lower information gain receive more noise.

In the field of federated learning, Qian et al. [26] introduced FDP-FL, a flexible privacy budget allocation strategy for federated learning. FDP-FL uses nonlinear noise scaling and adaptive budget distribution to optimize model accuracy while strictly limiting cumulative privacy loss. FDP-FL outperforms fixed-budget schemes, particularly in mitigating gradient leakage risks and improving model accuracy.

In the field of data mining, Li et al. [19] proposed GAPBAS, a genetic algorithm-based privacy budget allocation strategy for differentially private k-means clustering. By formulating privacy budget allocation as a combinatorial optimization problem, GAPBAS utilizes genetic algorithms to determine the optimal budget allocation sequence, significantly enhancing the convergence and usability of the k-means algorithm.

These studies highlight the significant advancements in privacy budget allocation strategies, from simple dynamic adjustments to sophisticated methods leveraging gradient dynamics, feature relevance, information gain, and optimization techniques. Such advancements have greatly reduced the adverse impact of differential privacy on data utility within constrained privacy budgets. However, the above methods mainly focus on parameter or gradient importance by allocating more privacy budget to those that are more important, neglecting learning rate scheduling strategies and the impact of training stages.

2.2 Local differential privacy for LLMs

With the widespread use of LLMs, protecting privacy during fine-tuning has become a critical issue. Traditional differential privacy methods, such as DP-SGD, prevent privacy leakage by adding noise to gradient information during backpropagation. However, DP-SGD is often centralized, requiring a trusted third party to aggregate the gradients, which introduces additional privacy and computational overhead. Additionally, deep learning models are highly sensitive to parameter perturbations, and adding noise directly to the trained model parameters can make the model completely unusable.

To address these issues, Du et al. [20] proposed DP-forward, which perturbs the embedding matrix during the forward pass of an LLM, thereby ensuring strict local differential privacy (LDP). Unlike DP-SGD, DP-forward uses an analytical matrix Gaussian mechanism to generate noise, significantly reducing the computational and storage costs introduced by noise while maintaining privacy.

Building on Du's work, Wang et al. [27] introduced SLDP-FT, a selective local differential privacy protection framework. SLDP-FT applies selective noise to the embedding matrix during the forward pass, which introduces noise only to privacy-sensitive tokens, achieving selective sequence local differential privacy (S-SeqLDP). This approach significantly reduces its impact on model performance, achieving a balance between privacy protection and efficient fine-tuning. However, although these existing studies have acknowledged the curse of dimensionality and made preliminary attempts to address it, their approaches remain largely superficial, lacking both theoretical rigor in solution design and systematic experimental validation.

3 Preliminaries

3.1 Differential privacy

Differential privacy is a mathematical framework designed to protect personal data privacy by providing strong privacy guarantees when analyzing and sharing large datasets.

Definition 1 ((ε,δ)-DP)**.** For any two adjacent datasets D and D' (i.e., they differ by only one data point), and for any possible output result O, if a random algorithm M satisfies the following condition, then the algorithm is said to implement (ε, δ)-DP,

$$\Pr(M(D)\in O)\leqslant \mathrm{e}^{\varepsilon}\Pr(M(D')\in O)+\delta. \tag{1}$$

3.2 Privacy accountant

Privacy accountant is a tool used to track and manage privacy loss when applying DP mechanisms during model training or data processing. It ensures that the noise added to the data or model does not exceed the privacy guarantees defined by the system.

Definition 2 (Privacy curve)**.** For a given DP mechanism M, its privacy curve is defined as follows:

$$\delta(D||D')(\varepsilon) = \sup_{O\in\Omega}\{\Pr(M(D) \in O) - \mathrm{e}^{\varepsilon}\Pr(M(D') \in O)\}. \tag{2}$$

The privacy curve reveals the smallest δ for each ε such that M satisfies (ε, δ)-DP.

Definition 3 (Composition of privacy curves)**.** For any two privacy curves $\delta_1 = \delta(D_1||D'_1)$, $\delta_2 = \delta(D_2||D'_2)$, the composition of the privacy curves is defined as

$$\delta_1 \otimes \delta_2 = \delta((D_1, D_2)||(D'_1, D'_2)). \tag{3}$$

Similarly, let $M_1, M_2, \ldots, M_n$ be differential privacy mechanisms with $\delta_1, \delta_2, \ldots, \delta_n$, respectively. The privacy curve of M_{all}, which is composed by $M_1, M_2, \ldots, M_n$ is given by $\delta_1 \otimes \delta_2 \otimes \cdots \otimes \delta_n$.

Definition 4 (Privacy loss random variables)**.** A privacy loss random variable (PLRV) is a randomized representation of the privacy curve. It transforms the properties of the privacy curve into the properties of random variables, allowing the combination of privacy curves to be achieved through the addition of random variables (or the convolution of distributions). For a given dataset D, its adjacent dataset D' and a differential privacy mechanism M, the PLRV is defined as follows:

$$PLRV_{D,D'}(x) = \log\left(\frac{\Pr[M(D) = x]}{\Pr[M(D') = x]}\right). \tag{4}$$

3.3 Gaussian mechanism

Gaussian mechanism conceals changes between adjacent datasets by adding zero-mean Gaussian noise to the query results, as shown in the following formula:

$$M(x) = f(x) + N(0, \sigma^2), \tag{5}$$

where $f(x)$ is the query or computation function, and $N(0, \sigma^2)$ is Gaussian noise with a mean of 0 and a standard deviation of σ. According to the definition of (ε, δ)-DP, the Gaussian mechanism satisfies

$$\sigma \geqslant \frac{Sensitivity}{\sqrt{2\ln(1.25/\delta)}} \cdot \frac{1}{\varepsilon}. \tag{6}$$

Sensitivity refers to the sensitivity of $f(x)$, quantifying the maximum variation in the output of $f(x)$ when a single record in the dataset is altered.

4 Problem formulation

Our framework is designed to perturb hidden layer embeddings through LDP noise. In the proposed scenario, each user has a sequence seq, consisting of n labels, along with a task-specific label y. Our goal is to safeguard seq from privacy attacks that may arise during the fine-tuning process, which requires repeated access to and training on user data.

To achieve this, we divide the model into three components: the noise perturbation layer, the pre-noise layer $pre(\cdot)$, and the post-noise layer $post(\cdot)$. In this design, the noise perturbation layer refers to a dedicated module that is inserted after one of the intermediate layers between the input embedding layer and the task layer. Based on this position, the pre-noise component $pre(\cdot)$ is defined as an arbitrarily deep forward pass from the input embedding layer up to but excluding the noise perturbation layer, and is held by the user. The post-noise component $post(\cdot)$, comprising the remaining layers starting immediately after the noise perturbation layer and extending to but excluding the task layer, is managed by the service provider. After passing through $pre(\cdot)$, the input sequence seq is converted into hidden embeddings emb and enters the noise perturbation layer. Before uploading these embeddings to the post-noise component $post(\cdot)$, the user applies a differential privacy mechanism M to emb within the noise perturbation layer. Specifically, M ensures that the service provider cannot extract the user's private information from the received emb', which is computed as follows:

$$emb' = M(emb), \quad \text{where } emb = pre(seq). \tag{7}$$

It is important to note that, due to the post-processing property of differential privacy, any operation the service provider performs on emb' will not affect the privacy of the data.

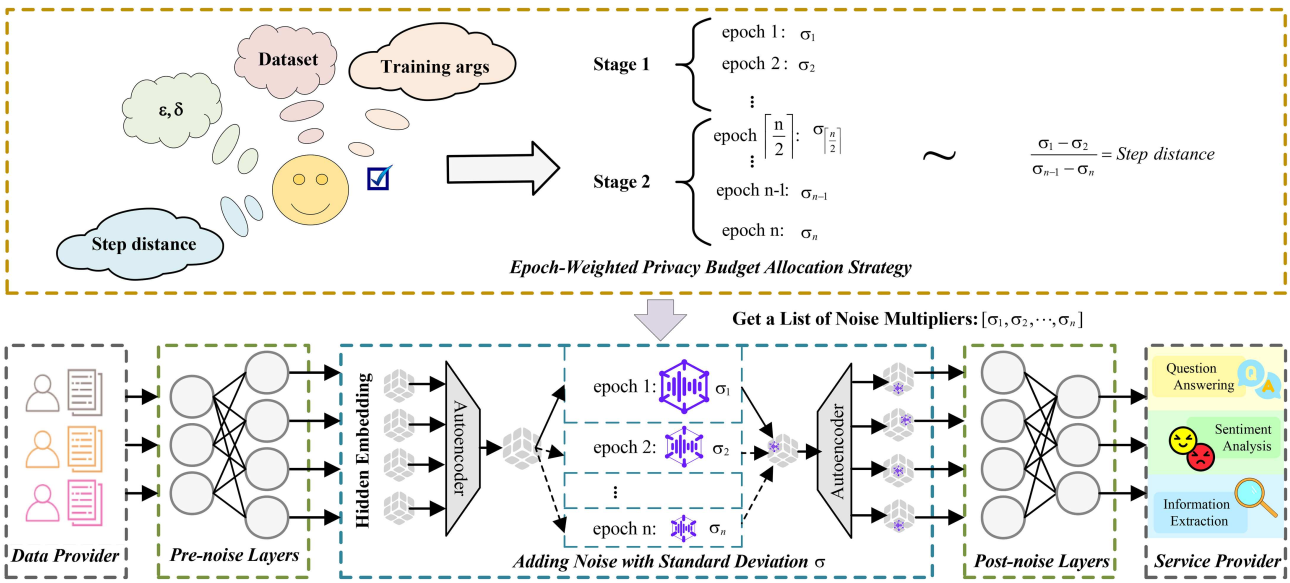


Figure 1 (Color online) The overall framework of EW-FT.

5 Methodology

5.1 Overview

The overall framework of EW-FT is shown in Figure 1. There are mainly two modules in our EW-FT framework.

The first module focuses on privacy budget allocation. We propose the EWPBA algorithm, which accounts for epoch-specific factors when distributing the privacy budget across epochs. As training progresses, EWPBA gradually reduces the noise multiplier. The fine-tuning process is divided into two stages: the initial rapid exploration stage with a larger noise multiplier and the subsequent precision adjustment stage with a smaller noise multiplier. The difference between adjacent epochs in the first stage is an integer multiple of that in the second stage, determined by the step distance. Through this manner, EWPBA reserves more privacy budget during the initial rapid exploration stage, which has relatively lower precision requirements, and reallocates the saved budget to the Precision Adjustment Stage, where more precise adjustments are needed. This strategy ensures a steady improvement in model accuracy.

The second module conducts dimensionality reduction, noise addition, and reconstruction. It adds low-dimensional noise to hidden embeddings generated by the pre-noise layer, to ensure LDP protection while mitigating the curse of dimensionality. Specifically, we utilize a stacked autoencoder to reduce feature dimensionality before adding noise and reconstructing embeddings after noise is added. The encoded outputs are normalized and clipped for sensitivity control. Gaussian noise is then sampled from the Gaussian mechanism and injected into the processed embeddings to implement local differential privacy. Finally, the embeddings are reconstructed. This paradigm—dimensionality reduction, noise addition, and reconstruction—not only enhances data privacy protection but also minimizes the impact on model performance.

5.2 Epoch-weighted privacy budget allocation algorithm

To validate the core assumption of our method that the model's sensitivity to noise varies across different training stages, we conducted preliminary experiments by injecting Gaussian noise of fixed magnitude at various fine-tuning epochs. As shown in Table 1, the impact of identical noise differs significantly depending on the fine-tuning epoch, with the most severe case resulting in an 8.58% accuracy drop in the earlier layer, indicating that the model's sensitivity to perturbation evolves throughout the training process. Based on this observation, we propose the EWPBA algorithm, which allocates noise multipliers based on epoch factors. Unlike conventional budget allocation methods that focus on privacy budget parameters ε and δ, EWPBA employs noise multipliers to directly and dynamically adjust the scale of added noise.

Algorithm 1 presents the pseudocode of our EWPBA algorithm. It takes dataset size $|D|$, privacy parameters ε, δ, batch size B, step distance S and training epoch E as input. The step distance S is set by the user, and controls the ratio of noise multipliers σ between adjacent epochs in the first and second stages of training (module 1). A

Table 1 SST-2 accuracy with different perturbation epochs and layers.

Epoch	Layer					
	Encoder#1	Encoder#3	Encoder#5	Encoder#7	Encoder#9	Encoder#11
1	0.814	0.823	0.8555	0.8865	0.9163	0.9174
2	0.7282	0.7867	0.8176	0.8635	0.8670	0.8922

Algorithm 1 Epoch-weighted privacy budget allocation.

Require: Dataset size $|D|$, privacy parameter ε, δ, batch size B, step distance S, training epoch E;
Ensure: Noise multiplier list σ_{list};
Let $\varepsilon' = \varepsilon - 1$, $P = B/|D|$, $Com = (|D|/B) \cdot E$;
Define $\sigma_{start} = 0.2$, $\sigma_{end} = 50$, and $\varepsilon_{est} = 0$;
while $|\varepsilon_{err} - \varepsilon'| > 0.0001$ **do**
  $\sigma_{mid} = (\sigma_{start} + \sigma_{end})/2$;
  $\varepsilon_{est} = ComposePRV(\sigma_{mid}, P, \delta, Com)$;
  if $\varepsilon_{est} > \varepsilon'$ **then**
    $\sigma_{start} = \sigma_{mid}$;
  else
    $\sigma_{end} = \sigma_{mid}$;
  end if
end while
Get beginning noise factor $\sigma_{beg} = \sigma_{mid}$;
Define $\sigma_1 = 0$, $\sigma_2 = \sigma_{beg}$;
while $|\varepsilon_{est} - \varepsilon'| > 0.0001$ **do**
  $\sigma_{fir} = (\sigma_1 + \sigma_2)/2$;
  $Add\_num = 4 \cdot (\sigma_{beg} - \sigma_{fir})/((E-1) \cdot (S+1))$;
  Create a noise multiplier list $\sigma_{list} = [\sigma_{fir}]$;
  $\sigma_{new} = \sigma_{fir}$;
  for $epoch_{num} = 1$ to $E-1$ **do**
    if $epoch_{num} \leqslant (E/2-1)$ **then**
      $\sigma_{new} = \sigma_{new} + Add\_num$;
    else
      $\sigma_{new} = \sigma_{new} + S \cdot Add\_num$;
    end if
    Append σ_{new} to list σ_{list};
  end for
  Calculate $\varepsilon_{est} = ComposePRV'(\sigma, P, \delta, Com, E)$;
  if $\varepsilon_{est} - \varepsilon > 0$ **then**
    $\sigma_1 = \sigma_{fir}$;
  else
    $\sigma_2 = \sigma_{fir}$;
  end if
end while
Reverse each element in σ_{list};
return σ_{list}.

smaller S corresponds to a larger mean noise multiplier and a smaller final noise multiplier. Next, the sampling probability $P = B/|D|$ and the self-combination number $Com = (|D|/B) \cdot E$ are calculated as parameters for privacy budget estimation.

We employ the PRV Accountant [28] for privacy accounting, which estimates privacy loss by truncating and discretizing PLRVs and can provide tight privacy bounds. The PRV Accountant combines privacy curves through the ComposePRV algorithm. Given a sampling probability P and a noise multiplier σ, the ComposePRV algorithm first constructs a PLRV Y based on the Poisson subsampled Gaussian mechanism. To enable numerical computation, Y is discretized over a symmetric truncated interval $[-L, L]$ with its cumulative distribution function (CDF) and a discretization step size h, producing a set of discrete probability masses $\{q_j\}$ associated with cell-centered grid points $\{j \cdot h\}$. To ensure that the discretized distribution accurately approximates the original, the support is shifted by an offset μ so that the expectation of the resulting discrete variable aligns with that of the truncated continuous PLRV. The resulting discrete PLRV $\widetilde{Y}_0$ is then composed Com times via convolution using the fast Fourier transform (FFT) to obtain the total privacy loss distribution $\widetilde{Y}$. Finally, the privacy curve $\delta(\varepsilon)$ is computed for $\varepsilon \in [0, L]$ using the following formula:

$$\delta_{\overline{Y}}(\varepsilon) = E_{\overline{Y}}[(1 - \mathrm{e}^{\varepsilon - \overline{Y}})_+]. \tag{8}$$

We provide the pseudocode of Algorithm 2 to facilitate deeper understanding. In subsequent applications of this algorithm, we adopt the default values of h and L as specified in the original work. For more computational details and theoretical foundations, please refer to the source [28].

Algorithm 2 ComposePRV: composing privacy curves with the same σ.

Require: Sampling probability P, noise multiplier σ, self-combination number Com, discretization step size h, truncation parameter L;
Ensure: Privacy curve $\delta_{\widetilde{Y}}(\varepsilon)$;
Construct PLRV Y through Poisson subsampled Gaussian mechanism with P and σ;
Get grid number $n = (L - h/2)/h$;
for $j = -n$ to n **do**
    Compute the jth probability mass $q_j = \mathrm{CDF}_Y(jh + h/2) - \mathrm{CDF}_Y(jh - h/2)$;
end for
Normalize $q_j \leftarrow q_j / \sum_{j=-n}^{n} q_j$;
Let $Y^L \leftarrow Y \mid |Y| \leqslant L$;
Compute offset $\mu \leftarrow \mathbb{E}[Y^L] - \sum_{j=-n}^{n} jh \cdot q_j$;
Construct discretized $\widetilde{Y}_0 \leftarrow jh + \mu$ with probability q_j;
Compute $\widetilde{Y} \leftarrow \underbrace{\widetilde{Y}_0 \oplus_L \cdots \oplus_L \widetilde{Y}_0}_{Com \text{ times}}$ using FFT;
for $\varepsilon \in [0, L]$ **do**
    $\delta_{\widetilde{Y}}(\varepsilon) \leftarrow \mathbb{E}_{\widetilde{Y}}[(1 - \mathrm{e}^{\varepsilon - \widetilde{Y}})_+]$;
end for
return $\delta_{\widetilde{Y}}(\cdot)$.

The EWPBA algorithm is divided into two main parts. The first part involves lines 1 to 12, which compute a noise multiplier σ_{beg} under a stricter privacy constraint using the unmodified version of ComposePRV. The second part involves lines 13 to 35, which calculate the noise multiplier list σ_{list} using a central axis noise factor σ_{beg}. To align ComposePRV with our requirements, we modify its internal implementation and refer to the resulting version as ComposePRV$'$ (Algorithm 3). Specifically, the original function, which accepts a single noise multiplier σ, is extended to support a list of noise multipliers $\boldsymbol{\sigma}_{\text{list}} = [\sigma_1, \ldots, \sigma_E]$ as input. For each σ_i in $\boldsymbol{\sigma}_{\text{list}}$, we independently construct the corresponding PLRV Y_i and assign a local composition count $Com_i = Com/E$. Each Y_i is composed of Com_i times, and the resulting distributions are then combined using FFT to obtain the overall privacy loss distribution. Finally, the privacy curve $\delta(\varepsilon)$ is computed from this distribution.

Algorithm 3 ComposePRV$'$: composing privacy curves with a list of σ.

Require: Training epoch E, sampling probability P, noise multiplier list $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_E]$, self-combination number Com, discretization step size h, truncation parameter L;
Ensure: Privacy curve $\delta_{\widetilde{Y}}(\varepsilon)$;
Set $com_i = com/E$;
for $i = 1$ to E **do**
    Construct PLRV Y_i through Poisson subsampled Gaussian mechanism with P and σ_i;
    Get grid number $n = (L - h/2)/h$;
    for $j = -n$ to n **do**
        Compute the j-th probability mass $q_j = \mathrm{CDF}_{Y_i}(jh + h/2) - \mathrm{CDF}_{Y_i}(jh - h/2)$;
    end for
    Normalize $q_j = q_j / \sum_{j=-n}^{n} q_j$;
    Let $Y_i^L = Y_i \mid |Y_i| \leqslant L$;
    Compute offset $\mu_i = \mathbb{E}[Y_i^L] - \sum_{j=-n}^{n} jh \cdot q_j$;
    Construct discretized $\widetilde{Y}_i = jh + \mu_i$ with probability q_j;
    Compute $\widetilde{Y}_i^{(com_i)} = \underbrace{\widetilde{Y}_i \oplus_L \cdots \oplus_L \widetilde{Y}_i}_{com_i \text{ times}}$ using FFT;
end for
Compute $\widetilde{Y} = \widetilde{Y}_1^{(com_1)} \oplus_L \cdots \oplus_L \widetilde{Y}_E^{(com_E)}$ using FFT;
for $\varepsilon \in [0, L]$ **do**
    $\delta_{\widetilde{Y}}(\varepsilon) = \mathbb{E}_{\widetilde{Y}}[(1 - \mathrm{e}^{\varepsilon - \widetilde{Y}})_+]$;
end for
return $\delta_{\widetilde{Y}}(\cdot)$.

A key challenge in privacy budget allocation is avoiding uniform noise multipliers across epochs. Directly applying the original ε to compute the noise multiplier and using it as the central axis for later iteration would lead to a uniform noise multiplier list. This is because noise multiplier values in list σ_{list} will progressively oscillate toward the central value, ultimately converging to the same number (before converging to the same value, the total privacy budget will not be equal to ε). Alternatively, our algorithm first applies $\varepsilon' = \varepsilon - 1$ to calculate the beginning noise factor to avoid this issue.

Part 2 is the core of the EWPBA algorithm. Based on commonly used learning rate schedules, we divide the fine-tuning process into two stages and adjust the privacy protection mechanism for efficient privacy protection and performance optimization. The two stages are described as follows.

Stage 1: Rapid exploration stage. In the early stages of training, a higher learning rate allows the model to quickly explore the loss function space and find a rough minimum. This "coarse-tuning" period aims to rapidly narrow the search space and locate potential solution regions. Since the precision of gradient information is less critical in this stage, more noise can be introduced without significantly affecting the overall model performance. Instead of evenly distributing the privacy budget, allocating a lower budget to this stage can preserve more privacy budget for the later stage without substantially impacting the training process.

Stage 2: Precision adjustment stage. In the later stage of training, a lower learning rate is important for fine-grained parameter adjustments near a local minimum. The goal of this stage is to refine model parameters towards the global optimum. In this precision adjustment stage, excessive noise can greatly degrade accuracy. Therefore, to support fine-grained adjustments, the added noise is reduced to allow for final fine-tuning based on accurate gradient information.

Based on the above two-stage planning, we allocate the first half of training epochs to stage 1 and the second half to stage 2. Then, we design an iteration strategy to assign privacy budgets dynamically across different epochs. This iterative allocation strategy is primarily based on binary search. Initially, we set σ_{fir}, the first element of the noise multiplier list, as the mean value of σ_1 (initialized to 0) and σ_2 (initialized to σ_{beg}). If the estimated privacy budget ε_{est} deviates significantly from the target privacy budget ε, either σ_1 or σ_2 is updated with σ_{fir}. Next, using the step distance S and training epoch E, we calculate the base addition number Add_num, which represents the difference between noise multipliers of adjacent epochs,

$$Add\_num = \frac{4(\sigma_{beg} - \sigma_{fir})}{(E-1)\cdot(S+1)}. \tag{9}$$

During Stage 1, each epoch receives a noise multiplier σ_{new} that is smaller than the previous epoch's by $S \cdot Add\_num$. In Stage 2, the noise multiplier σ_{new} decreases by Add_num per epoch. In practice, we implement this process in reverse by generating the noise multiplier list in the opposite order and then inverting it after computation, achieving the same effect as our design. The difference between noise multipliers of adjacent epochs can be expressed as

$$\sigma_{new}{'} = \begin{cases} \sigma_{new} + \mathrm{Add\_num}, & \mathrm{epoch}_{num} \leqslant \dfrac{E}{2} - 1, \\ \sigma_{new} + S \cdot \mathrm{Add\_num}, & \mathrm{epoch}_{num} > \dfrac{E}{2} - 1. \end{cases} \tag{10}$$

Finally, we generate a noise multiplier list σ_{list} and compute the total privacy budget consumption ε_{est} with ComposePRV$'$. If ε_{est} exceeds ε, we update σ_1 to σ_{fir}; otherwise, we set σ_2 to σ_{fir}. The iteration continues until the difference between ε and ε_{est} falls below a predefined threshold (set to 0.0001).

Through this optimization, we obtain a noise multiplier list that meets the overall privacy budget constraint, where each value specifies the standard deviation of the noise introduced in each epoch.

5.3 Implementation of local differential privacy with reduced model performance loss

Our LDP mechanism is implemented by injecting noise into the hidden embeddings during the forward pass. A key challenge when adding noise to the hidden embeddings is determining the sensitivity of *emb* (a key parameter of differential privacy), since the changes in *emb* are often non-trivial and may even be non-Laplacian continuous. A typical sensitivity control approach is to normalize the function output to a fixed F-norm and clip it by a clipping factor C. This method has proven to be particularly effective when the injection target is in the deeper layers of the model. The clipping formula is shown below:

$$\|pre(\cdot)\|_F = C \ \ or \ \ pre(\cdot)/\max\left(1, \frac{\|pre(\cdot)\|_F}{C}\right). \tag{11}$$

After clipping, the sensitivity of the hidden embeddings becomes $2C$, making it possible for DP implementation.

Furthermore, we apply Gaussian noise perturbation to the hidden embeddings. The choice of differential privacy mechanism M is not a critical issue for our framework, as our focus is not on the implementation of the differential privacy mechanism or tightening DP bounds. Instead, our main concern is how to efficiently allocate privacy budget under a given privacy budget consumption mechanism and budget constraints. Therefore, we use a classical Gaussian mechanism with $\varepsilon \geqslant 0$ and $0 \leqslant \delta \leqslant 1$ to perturb *emb*. The standard deviation sequence of the noise is given by the EWPBA algorithm. After noise injection, the hidden embeddings enter the post noise layer $post(\cdot)$ for further forward propagation.

Table 2 Size of task datasets.

	SST-2	QNLI	IMDB
Training samples	67349	104743	25000
Test samples	1821	5463	25000

Table 3 SST-2 accuracy with different sequence lengths and perturbation layers. Bold values indicate the best result for each l across different perturbation layers.

Layer	l				
	16	32	64	128	256
Output	0.844	0.9071	**0.9209**	0.9174	0.9128
Encoder#11	0.8521	0.9002	**0.9174**	0.9037	0.9094
Encoder#8	0.8349	0.8876	**0.9094**	0.9060	**0.9094**
Encoder#5	0.8085	0.8521	**0.8773**	0.8727	0.8532
Encoder#2	0.7638	0.8303	**0.8337**	0.8085	0.8131

However, applying DP to high-dimensional data often requires injecting large amounts of noise to ensure privacy protection, which often leads to a decline in model performance. This issue, known as the curse of dimensionality, becomes particularly prominent in the earlier layers of the model.

To address this issue, we introduce a stacked autoencoder layer, whose input dimension matches the hidden layer dimension of the BERT pipeline while the output dimension and depth are specified by the user. Notably, we can independently control the noise scale and the abstraction level of the hidden embeddings by controlling the output dimension and autoencoder depth separately. The stacked autoencoder consists of one or more encoder layers and decoder layers. The encoders learn key features of input data and map them into a lower-dimensional space. They are positioned after the pre-noise layer $pre(\cdot)$ and before the normalization and clipping operations. The decoders reconstruct the original input from the low-dimensional representation, and are located after the noise perturbation layer and before the post-noise layer $post(\cdot)$. This improvement not only reduces the complexity but also removes redundant information, further enhancing model robustness to noise.

By incorporating the stacked autoencoder layer, we map the output of the hidden layers to a lower-dimensional space without significantly harming the model's accuracy. This reduces the scale of added noise, thereby improving the model's performance while maintaining privacy protection.

6 Experiments

6.1 Experimental setup

Datasets and models. We conducted experiments on three NLP tasks: (1) SST-2 for binary classification of single-sentence sentiment, (2) IMDB for binary classification of movie review sentiments, and (3) QNLI for binary classification of positive/negative examples. The dataset sizes are shown in Table 2. Following the baseline method [20], we selected BERT-base-uncased as the base checkpoint.

Baseline and parameter settings. We selected DP-forward [20] as our baseline and adopted most of its parameter settings. Specifically, we used example-level settings with the following hyperparameters: number of training epochs $E = 3$, learning rate $\eta = 2\text{e}{-5}$, batch size $B = 32$, and normalization/clipping factor $C = 1$. Other settings (e.g., no weight decay, no learning rate decay) are set to their defaults as in [20, 29]. The privacy parameter δ is fixed at 1e$-$5. Unless otherwise specified, ε is set to 8 and the autoencoder dimension is set to 4.

6.2 Experimental results

Impact of the sequence length l. To determine an appropriate sequence length, we investigated its impact on model performance. We conducted experiments on SST-2 with varying sequence lengths from 16 to 256 and added noise perturbations at different depths of model layers (i.e., Output, Encoder#11, Encoder#8, Encoder#5, Encoder#2). As shown in Table 3, lower sequence lengths lead to a significant decrease in accuracy, and the best results are obtained at a sequence length of 64. Therefore, we set the sequence length l to 64 in subsequent experiments.

Impact of the autoencoder's dimension dim. We explored autoencoders of different sizes and depths. To achieve the most effective autoencoder for dimensionality reduction and reconstruction of BERT hidden layer embeddings, we selected five autoencoders, three of which have the same depth but different sizes (768×4, 768×64,

Table 4 SST-2 accuracy of EW-FT with different autoencoders when perturbing Encoder#5. Bold values indicate the best result for each ε across different *dim*.

dim	ε			
	2	4	8	16
768 × 4	0.8349	0.8658	0.8784	0.8773
768 × 64	**0.8567**	**0.8693**	**0.8807**	0.8784
768 × 256	0.8417	0.8670	0.8521	0.8716
768 × 64 × 4	0.8463	0.8601	0.8773	0.8739
768 × 256 × 64 × 4	0.8486	0.8681	0.8716	**0.8876**

Table 5 SST-2 accuracy of EW-FT with different step distances S when perturbing Encoder#5. Bold values indicate the best result for each ε across different S.

S	ε			
	2	4	8	16
1	0.8555	0.8658	0.8704	**0.8842**
2	**0.8567**	**0.8693**	0.8807	0.8784
3	0.8498	0.8555	0.8693	0.8716
4	0.8521	0.8589	**0.8819**	0.8784

768×256), while the other two have different depths (768×64×4, 768×256×64×4). We tested each autoencoder and applied noise perturbation to Encoder#5 to investigate the impact of autoencoder dimensions on model accuracy under various privacy budgets.

In this set of experiments, the "size" of an autoencoder refers to the reduced dimensionality of the hidden embeddings, while the "depth" denotes the number of layers used to extract higher-level feature representations. Table 4 presents the results of experiments conducted on different autoencoder dimensions. Among those with the same depth but different sizes (768×4, 768×64, 768×256), the 768×64 autoencoder outperforms other configurations in terms of dimensionality reduction and reconstruction quality. We think this is because when compared with smaller autoencoders, the 768 × 64 autoencoder has less distortion to the features, thereby preserving more feature information and yielding better reconstruction outcomes. Compared with larger autoencoders, it achieves more efficient dimensionality reduction and introduces less noise into the hidden embeddings. In other words, it strikes a better balance between dimensionality reduction and feature preservation. For autoencoders with different depths (768 × 4, 768 × 64 × 4, 768 × 256 × 64 × 4), the 768 × 256 × 64 × 4 configuration achieves the best performance in most cases, demonstrating that deeper architectures with hierarchical feature extraction can enhance accuracy to a certain extent. Based on this analysis, we fixed the autoencoder dimension at 768 × 64 for the remaining experiments.

Impact of the step distance S. The step distance S controls the ratio of noise multipliers σ between adjacent epochs in the two training stages. Our experiments have revealed that the step distance S can reflect the overall mean value of σ: a smaller S corresponds to a larger mean σ. Although the final round's σ would also be smaller in this case, the overall increase in σ can lead to suboptimal fine-tuning performance. Therefore, it is necessary and meaningful to study the impact of the step distance S.

In this batch of experiments, we tested four different values for the step distance: 1, 2, 3, and 4. A step distance of 1 represents equal increments, meaning that the difference in σ between adjacent rounds is constant. The experimental results are given in Table 5. As shown in the table, setting the step distance S to 2 yields the best overall performance, followed by $S = 1$, while $S = 3$ and $S = 4$ result in relatively poorer accuracy. This is because $S = 2$ achieves a better balance between early and later epochs, allowing more privacy budget to be reserved for fine-tuning in the later stage of training. In contrast, when S is set to 3 or 4, although less noise is added in the later epochs, ensuring overall privacy requires injecting an excessively large amount of noise during the early epochs. This may lead to severe parameter deviation early in training, ultimately degrading the model performance. Therefore, we fixed the step distance at 2 for all subsequent experiments.

Ablation analysis of each module in the EW-FT framework. To assess the individual contributions of each module within our EW-FT framework, we conducted three sets of experiments: (1) applying only the EWPBA algorithm, excluding the autoencoder; (2) utilizing only the autoencoder, excluding EWPBA; and (3) implementing the complete EW-FT framework with both the EWPBA algorithm and the autoencoder. The performance of each configuration was evaluated across various perturbation positions (Output, Encoder#2, Encoder#5, Encoder#8, Encoder#11) and compared with the baseline method DP-forward, as shown in Table 6. In addition, we provide the accuracy of both the non-private baseline and the classical DP-SGD to intuitively demonstrate the accuracy

Table 6 Ablation experiments on different perturbation layers. Bold values indicate the best result for each layer across different experimental settings used in the ablation experiments. DP-SGD and non-private results are provided for reference and are excluded from the comparison.

Layer	Mechanism					
	DP-forward	Only EWPBA	Only autoencoder	EW-FT	DP-SGD	Non-private
Encoder#2	0.8177	0.8222	0.8268	**0.8314**	0.8759	0.9289
Encoder#5	0.8704	0.8750	0.8796	**0.8842**		
Encoder#8	0.8979	0.9025	**0.9048**	**0.9048**		
Encoder#11	0.9071	**0.9197**	0.9128	0.9186		
Output	0.9151	**0.9232**	0.9025	0.9117		

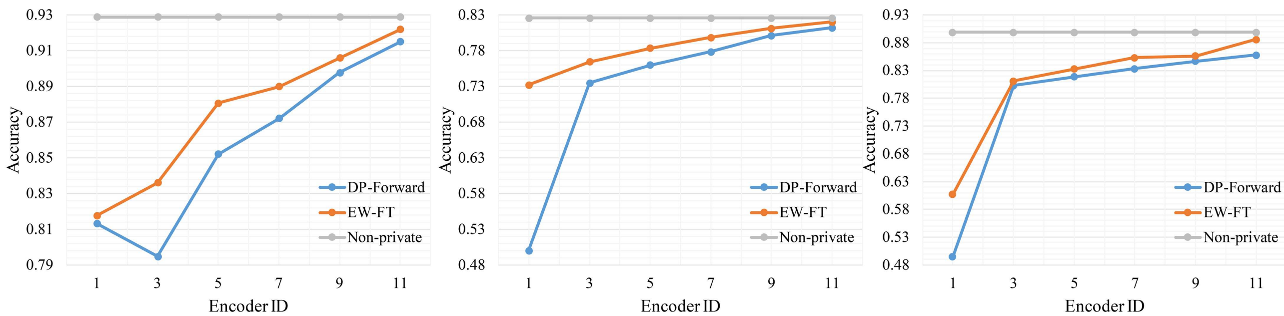
Table 7 Accuracy of fine-tuned BERT with different datasets on Encoder#5. Bold values indicate the best accuracy for each dataset and ε when comparing the baseline method (DP-Forward) and our method (EW-FT). Non-private results are provided for reference and are excluded from the comparison.

Mechanism	SST-2			IMDB			QNLI		
	$\varepsilon = 2$	$\varepsilon = 4$	$\varepsilon = 8$	$\varepsilon = 2$	$\varepsilon = 4$	$\varepsilon = 8$	$\varepsilon = 2$	$\varepsilon = 4$	$\varepsilon = 8$
DP-forward	0.828	0.8417	0.8521	0.7309	0.7450	0.7601	0.7972	0.7979	0.8191
EW-FT	**0.8567**	**0.8693**	**0.8807**	**0.7498**	**0.7673**	**0.7837**	**0.8146**	**0.8210**	**0.8331**
Non-private	0.9289			0.8259			0.8993		

degradation of our method compared with the non-private baseline, as well as the accuracy improvement of our method over the classical CDP approach. The results indicate that the EWPBA algorithm consistently improves performance across different model layers, while the autoencoder performs better in shallower layers. We attribute this to the varying levels of abstraction in hidden layer embeddings: shallow embeddings are more sensitive to noise perturbations compared with deeper embeddings. In shallow layers, incorporating the autoencoder effectively reduces noise dimensionality, thereby enhancing performance. The integration of these two modules provides complementary benefits, leading to overall improved performance.

Performance of EW-FT against DP-forward on all three datasets. We conducted comparative evaluations of EW-FT against DP-forward under different privacy settings across three benchmark datasets. Table 7 shows that under the same privacy parameter configuration, EW-FT consistently outperforms DP-forward in terms of accuracy on all three datasets. This improvement can be attributed to two components: the EWPBA algorithm, which reduces noise in the later training stage to better control parameter updates, and the stacked autoencoder, which reduces the dimensionality of hidden embeddings and thereby lowers the overall noise scale. However, the magnitude of performance improvement brought by EW-FT varies depending on dataset characteristics. Specifically, SST-2 consists of short, sentiment-rich sentences with relatively low semantic complexity, which benefits significantly from allocating a larger privacy budget to later training epochs, thereby making EW-FT's privacy budget reallocation strategy particularly effective. For IMDB, which features longer and more linguistically diverse reviews, EW-FT's stacked autoencoder helps compress high-dimensional hidden embeddings prior to noise injection, preserving semantic information and achieving a moderate performance improvement. In the case of QNLI, a sentence pair classification task that requires fine-grained semantic inference, although EW-FT still yields improvements over DP-forward, the gain is comparatively smaller due to the increased task complexity. On average, EW-FT achieves accuracy gains of 2.8%, 2.2%, and 1.8% over DP-forward on the SST-2, IMDB, and QNLI datasets, respectively. Furthermore, we observe a positive correlation between the privacy budget parameter ε and model accuracy: as ε increases, the accuracy improves accordingly. This trend aligns with expectations, as a larger ε implies a higher privacy budget, resulting in less noise being added during fine-tuning.

In addition, we investigate the impact of adding noise to different encoder layers across the three datasets. The experimental results, depicted in Figure 2, indicate that our EW-FT framework achieves significant accuracy gains over the baseline when noise perturbations are applied to shallower layers (Encoder#1 or Encoder#3). Although the performance advantage reduces when noise is added to deeper layers, our EW-FT consistently outperforms DP-forward. Moreover, the results reveal a clear trend: the earlier the noise is injected into the encoder, the more the model performance deteriorates. This is because injecting noise into early-phase hidden embeddings disrupts semantic representations at an earlier point in the forward pass, thereby amplifying the adverse impact of noise on downstream learning. Notably, the variation in performance across datasets aligns with the pattern observed in previous experiments—SST-2 exhibits the largest overall accuracy gain, followed by IMDB, while QNLI has the smallest improvement.



(a) SST-2 (b) IMDB (c) QNLI

Figure 2 (Color online) Comparison of accuracy when perturbing different encoder layers.

7 Conclusion

In this paper, we propose EW-FT, a novel efficient privacy-preserving framework that enables distributed LDP embedding perturbation during downstream task fine-tuning of pre-trained LLMs. Our framework incorporates an EWPBA algorithm, which considers epoch factors and dynamically assigns Gaussian noise of varying intensities to each training epoch. By prioritizing critical fine-tuning stages, it reserves more privacy budget for later phases, ensuring better model optimization.

Furthermore, we significantly mitigate the "curse of dimensionality" by combining the local differential privacy paradigm with stacked autoencoders and injecting noise into high-dimensional representations of hidden embeddings during data reduction and reconstruction, particularly in shallow layers.

Extensive experiments demonstrate that under fixed privacy budget constraints, our EW-FT framework significantly improves model utility compared with state-of-the-art methods, offering a novel perspective on balancing the privacy-utility trade-off in LLM training. Future work may extend this framework to a broader range of tasks and explore adaptive noise mechanisms tailored to deeper architectures with greater depth and parameter scales to overcome the current challenges in applying differential privacy to larger models.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 62372173).

References

1 Devlin J, Chang M W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, 2019. 4171–4186
2 Ke W J, Shang Z Y, Luo Z Z, et al. Unveiling factuality and injecting knowledge for LLMs via reinforcement learning and data proportion. Sci China Inf Sci, 2024, 67: 209101
3 Li W J, Lyu D X, Wang G, et al. Hardware-oriented algorithms for softmax and layer normalization of large language models. Sci China Inf Sci, 2024, 67: 200404
4 Wei J, Bosma M, Zhao V, et al. Finetuned language models are zero-shot learners. In: Proceedings of International Conference on Learning Representations, 2022
5 Zhang W, Deng Y, Liu B, et al. Sentiment analysis in the era of large language models: a reality check. In: Findings of the Association for Computational Linguistics. Mexico City: Association for Computational Linguistics, 2024. 3881–3906
6 Ross S I, Martinez F, Houde S, et al. The programmer's assistant: conversational interaction with a large language model for software development. In: Proceedings of the 28th International Conference on Intelligent User Interfaces, 2023. 491–514
7 Feng Y, Qiang J, Li Y, et al. Sentence simplification via large language models. ArXiv:2302.11957
8 Shokri R, Stronati M, Song C, et al. Membership inference attacks against machine learning models. In: Proceedings of the 2017 IEEE Symposium on Security and Privacy, 2017. 3–18
9 Carlini N, Tramer F, Wallace E, et al. Extracting training data from large language models. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), 2021. 2633–2650
10 Staab R, Vero M, Balunović M, et al. Beyond memorization: violating privacy via inference with large language models. In: Proceedings of International Conference on Learning Representations, 2024
11 Dwork C, McSherry F, Nissim K, et al. Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd Theory of Cryptography Conference, 2006. 265–284
12 Abadi M, Chu A, Goodfellow I, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016. 308–318
13 Yu D, Zhang H, Chen W, et al. Large scale private learning via low-rank reparametrization. In: Proceedings of International Conference on Machine Learning, 2021. 12208–12218
14 Yu D, Naik S, Backurs A, et al. Differentially private fine-tuning of language models. In: Proceedings of International Conference on Learning Representations, 2022
15 Li X, Tramer F, Liang P, et al. Large language models can be strong differentially private learners. In: Proceedings of International Conference on Learning Representations, 2022
16 Bu Z, Wang Y X, Zha S, et al. Differentially private optimization on large model at small cost. In: Proceedings of International Conference on Machine Learning, 2023. 3192–3218
17 Chase M, Chow S S M. Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, 2009. 121–130
18 Chen L, Yue D, Ding X, et al. Differentially private deep learning with dynamic privacy budget allocation and adaptive optimization. IEEE Trans Inform Forensic Secur, 2023, 18: 4422–4435

19 Li Y, Song X, Tu Y, et al. GAPBAS: genetic algorithm-based privacy budget allocation strategy in differential privacy K-means clustering algorithm. Comput Secur, 2024, 139: 103697

20 Du M, Yue X, Chow S S M, et al. Dp-forward: fine-tuning and inference on language models with differential privacy in forward pass. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023. 2665–2679

21 Hong J, Wang Z, Zhou J. Dynamic privacy budget allocation improves data efficiency of differentially private gradient descent. In: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, 2022. 11–35

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

23 Yu M, Quan T, Peng Q, et al. A model-based collaborative filtering algorithm based on stacked AutoEncoder. Neural Comput Appl, 2022, 34: 2503–2511

24 Wang N, Wang S, Li M, et al. Balancing differential privacy and utility: a relevance-based adaptive private fine-tuning framework for language models. IEEE Trans Inform Forensic Secur, 2025, 20: 207–220

25 Zhang X, Yang F, Guo Y, et al. Adaptive differential privacy mechanism based on entropy theory for preserving deep neural networks. Mathematics, 2023, 11: 330

26 Qian W, Shen Q, Chen X, et al. FDP-FL: differentially private federated learning with flexible privacy budget allocation. Comput J, 2024, 67: 3180–3195

27 Wang T, Zhai L D, Yang T F, et al. Selective privacy-preserving framework for large language models fine-tuning. Inform Sci, 2024, 678: 121000

28 Gopi S, Lee Y T, Wutschitz L. Numerical composition of differential privacy. Adv Neural Inform Process Syst, 2021, 34: 11631–11642

29 Wang A, Singh A, Michael J, et al. GLUE: a multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of International Conference on Learning Representations, 2019