Balancing Differential Privacy and Utility: A Relevance-Based Adaptive Private Fine-Tuning Framework for Language Models

Naiyu Wang, Shen Wang, Meng Li, *Senior Member, IEEE*, Longfei Wu, *Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, Zhitao Guan, *Member, IEEE*, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—Differential privacy (DP) has been proven to be an effective universal solution for privacy protection in language models. Nevertheless, the introduction of DP incurs significant computational overhead. One promising approach to this challenge is to integrate Parameter Efficient Fine-Tuning (PEFT) with DP, leveraging the memory-efficient characteristics of PEFT to reduce the substantial memory consumption of DP. Given that fine-tuning aims to quickly adapt pretrained models to downstream tasks, it is crucial to balance privacy protection with model utility to avoid excessive performance compromise. In this paper, we propose a Relevance-based Adaptive Private Fine-Tuning (Rap-FT) framework, the first approach designed to mitigate model utility loss caused by DP perturbations in the PEFT context, and to achieve a balance between differential privacy and model utility. Specifically, we introduce an enhanced layer-wise relevance propagation process to analyze the relevance of trainable parameters, which can be adapted to the three major categories of PEFT methods. Based on the relevance map generated, we partition the parameter space dimensionally, and develop an adaptive gradient perturbation strategy that adjusts the noise addition to mitigate the adverse impacts of perturbations. Extensive experimental evaluations are conducted to demonstrate that our Rap-FT framework can improve the utility of the fine-tuned model compared to the baseline differentially private fine-tuning methods, while maintaining a comparable level of privacy protection.**

***Index Terms*—Differential privacy, language models, parameter efficient fine-tuning, layer-wise relevance.**

Received 18 June 2024; revised 4 November 2024; accepted 24 November 2024. Date of publication 12 December 2024; date of current version 26 December 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62372173, Grant 62372149, and Grant U23A20303; and in part by the Fundamental Research Funds for the Central Universities under Grant 2023JC003. The associate editor coordinating the review of this article and approving it for publication was Prof. Chia-Mu Yu. *(Corresponding author: Zhitao Guan.)*

Naiyu Wang, Shen Wang, and Zhitao Guan are with the School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China (e-mail: Shininess_y@163.com; wangshen@ncepu.edu.cn; guan@ncepu.edu.cn).

Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, the School of Computer Science and Information Engineering, Anhui Province Key Laboratory of Industry Safety and Emergency Technology, and the Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230601, China (e-mail: mengli@hfut.edu.cn).

Longfei Wu is with Department of Mathematics Computer Science, Fayetteville State University, Fayetteville, NC 28301 USA (e-mail: lwu@uncfsu.edu).

Zijian Zhang and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhangzijian@bit.edu.cn; liehuangz@bit.edu.cn).

Digital Object Identifier 10.1109/TIFS.2024.3516579

I. Introduction

TRANSFORMER-BASED language models have demonstrated enormous potential in handling tasks across multiple scenarios and disciplines with their powerful natural language understanding and creative writing abilities [1]. Fine-tuning methods can align language models with vertical domains by training on downstream task data, further enhancing their universality [2], [3]. However, alongside groundbreaking transformations, concerns about privacy and security arise. Research has shown that sensitive information can be successfully extracted from language models through inference attacks [4], [5] and privacy extraction attacks [6], [7], [8].

Following its accomplishments in traditional deep learning, Differential Privacy (DP) has become the predominant approach for addressing privacy concerns within the realm of language models [8], [9]. Lukas et al. [6] evaluated the defense efficacy of existing privacy-preserving methods in language models against three black-box inference attacks. Their findings show that differential privacy can significantly mitigate the leakage of Personally Identifiable Information (PII) when fine-tuning language models. However, there is a key challenge in the transfer process from a deep learning model to a large language model, mainly due to the computational burden involved in training large transformer-based models integrated with DP [9], [10], [11]. To solve this problem, Li et al. [9] proposed a high-performance DP fine-tuning process for large language models along with a memory-saving technique called ghost clipping to alleviate the significant memory overhead associated with fine-tuning large models. Du et al. [10] introduced DP-forward, a DP fine-tuning method that directly perturbs embedding matrices, significantly reducing both time and memory costs.

While many studies focus on dealing the huge memory overheads for applying differential privacy to large models, Yu et al. [11] presented an alternative approach, which leverages Parameter Efficient Fine-Tuning (PEFT) to mitigate DP-induced memory overhead. PEFT addresses the challenge of the immense computational burden for Full Fine-Tuning by updating only a small set of parameters to achieve alignment. Furthermore, they proposed a parameter-efficient private fine-tuning framework based on DP. Since the DP noise is added to only a small number of parameters, it effectively alleviates the challenges posed by memory overhead. However, considering the goal of fine-tuning is to achieve a high performance domain-specific model by training on downstream data, if the introduction of privacy protection methods leads to significant performance degradation, it contradicts the original objective.

In the traditional deep learning domain, there are several mainstream approaches that have attempted to minimize the degradation in model utility caused by DP noise while preserving differential privacy: One type of approaches aims to enhance gradient perturbation accuracy by reducing redundant noise. This is achieved by splitting the gradients dimensionally and adding adaptive noise based on the sensitivity of each gradient component [12], [13]. Another type of approaches introduces interpretability techniques, such as Layer-wise Relevance propagation (LRP) [14], to differentiate the importance of various neurons or samples. These approaches add minor perturbations to the components contributing more significantly and vice versa [15], [16]. However, the above methods are only applicable to traditional deep neural networks where DP noise is added to all parameters. Existing private fine-tuning methods based on DP add an equal amount of noise to each trainable parameter [11]. Therefore, **a critical challenge in the context of PEFT is to minimize the adverse effects of DP-induced noise on model utility while simultaneously adhering to strict differential privacy constraints-a trade-off that necessitates effective strategies to counterbalance the utility impacts of privacy perturbation.** To the best of our knowledge, there is currently no relevant research on this matter.

Challenges: 1) In a PEFT-oriented framework that preserves differential privacy, perturbation is solely added to trainable parameters, which constitute a small portion of the model. Therefore, the first challenge is to interpret the trainable parameters in PEFT to differentiate their importance. 2) On this basis, in order to achieve efficient noise perturbation, we need to add minor perturbations to relatively important parameters during fine-tuning. The second challenge is to design an adaptive perturbation mechanism while satisfying DP. 3) Given the diverse types of PEFT methods, the location, size, and structure of trainable parameters vary significantly. Therefore, the final challenge lies in designing a generalized approach that balances privacy and utility, adaptable to various PEFT methods while addressing the aforementioned challenges.

To tackle these challenges, we introduce an LRP-based interpretability method to differentiate the importance of various trainable parameters. Then, we split the gradients of training samples dimensionally and add adaptive noise to the strongly and weakly significant gradient components, thereby reducing the performance degradation caused by perturbation. Furthermore, we draw inspiration from a taxonomy of the PEFT technique [3], which categorizes various PEFT methods into three types: additive, selective, and reparameterization-based. By analyzing the characteristic of each type, we devise a methodology that integrates relevance analysis with the trainable parameters of different PEFT methods. This makes our approach capable of conducting relevance analysis on the relevant trainable parameters associated with any given type of PEFT methods.

Contributions:

1) We propose the Relevance-based Adaptive Private Fine-Tuning framework (Rap-FT) to balance differential privacy and utility. This framework is adaptable to three major categories of PEFT methods: Additive, Selective, and Reparameterization-based parameter-efficient fine-tuning. To the best of our knowledge, this is the first work to minimize model utility degradation due to DP perturbations while satisfying differential privacy constraints in DP-integrated PEFT.
2) We propose a layer-wise relevance propagation (LRP)-based interpretability method for PEFT of language models to generate a relevance map of the trainable parameters, marking the first attempt to explain PEFT. Furthermore, we extend the application of relevance propagation to generative tasks.
3) We develop an adaptive private fine-tuning strategy, which splits gradients dimensionally and adds more noise to gradients with less impact on model utility according to the relevance map. This strategy effectively reduces redundant noise and mitigates the negative effects of perturbations.

II. Related Work

A. Differentially Private Fine-Tuning of Language Models

Due to its outstanding performance in private deep learning, differential privacy has become the preferred technology for privacy preservation in large language model training and fine-tuning processes [10]. It is also proved that DP can notably reduce the language models' memorization of name entities [17]. To address the challenge of performance drops and high computational overhead in the fine-tuning process, Li et al. [9] highlighted the importance of hyperparameters for DP optimization and proposed a memory-saving technique. This work enabled large transformers to be fine-tuned privately with nearly identical memory costs as non-private methods.

Yu et al. [11] employed the parameter-efficient fine-tuning technique and proposed a differential privacy-based private fine-tuning meta-framework, which can achieve model utility comparable to that of non-private models for many fine-tuning tasks. It is worth mentioning that benefiting from the computational and memory efficiency of the PEFT technique, the meta framework can also mitigate the high costs of computation and memory to some extent. Lukas et al. [6] verified the privacy-preserving effectiveness of existing privacy-preserving fine-tuning approaches. For differential privacy-based approaches, results show that existing differential privacy techniques can protect PII sequences from disclosure to a certain extent, but they may limit the utility of the model. Consequently, there is a necessity to seek improvements that can achieve a more favorable balance between the model's utility and privacy protection [6], [18].

B. Adaptive Differentially Private Deep Learning

We classify the existing differential privacy-based privacy-preservation approaches in the traditional deep learning domain into three categories: reducing model size, employing adaptive noise perturbation, and employing interpretability method-based perturbation.

1) Reducing Model Size: The first category of approaches is based on the intuition that the privacy-utility trade-off worsens as the model size increases [19]. The schemes in this category often incorporate pruning strategies [20]. However, in the PEFT method, the model is already fine-tuned with a small number of parameters, and further pruning might adversely affect the model's utility.

2) Adaptive Noise Perturbation: This category of approaches aims to enhance the accuracy of gradient perturbation to mitigate the accuracy degradation introduced by DP noise. Xiang et al. [12] suggested that perturbations on different parameters have varying impacts on training accuracy. They formulate the perturbation as an optimization problem to find a probability distribution of the random noise that minimizes the total perturbation cost. By adding noise from different distributions to each gradient component, the optimization throughout the training process becomes computationally prohibitive. Xu et al. [13] proposed an adaptive and computationally efficient differentially private deep learning approach, ADADP, which adds gaussian noise with lower variance to gradient components with smaller sensitivity. This approach leverages the observation that the gradient components have heterogeneous sensitivity to training samples. However, estimating the current gradient using historical gradients may introduce additional memory overhead [21].

3) Interpretability Method-Based Perturbation: Gong et al. [15] introduced LRP-based relevance analysis and designed an adaptive differential privacy-preserving learning framework. The result of relevance analysis indicates the contribution of each neuron to the model output and serves as a basis for privacy budget allocation, thus adding different levels of noise to the gradients. In contrast, Chen et al. [16] proposed to compute the sum of feature relevance for each sample using the LRP-method and presented a dynamic privacy budget allocation based on the sample dimension, which helps to inject proper noise into the gradients. However, the relevance analysis process of the above work can not be applicable to PEFT where DP noise is only added to trainable parameters. In this paper, we propose a relevance propagation based on the PEFT of language models, and combine the advantages of the Adaptive noise perturbation and Interpretability method-based perturbation. The result of the relevance analysis is applied to divide the parameters dimensionally, thereby meeting the requirements of the variable trainable parameters in PEFT methods, enabling efficient noise perturbation, and avoiding the high computational cost of per-dimension analysis.

III. Preliminaries

A. Differential Privacy and Its Application in Deep Neural Networks

Definition 1 (ε, δ-Differential Privacy [22]): A randomized mechanism $\mathcal{M}: \mathcal{D} \rightarrow \mathcal{R}$ is (ϵ,δ)-Differential Privacy if for any adjacent inputs $D, D' \in \mathcal{D}$ and any subset of output $R \in \mathcal{R}$, it holds that:

$$Pr(\mathcal{M}(D) \in R) \leq e^{\epsilon} Pr(\mathcal{M}(D') \in R) + \delta \tag{1}$$

1) Gaussian Mechanism.: let $f: D \rightarrow \mathcal{R}$ be a randomized mechanism with domain $\mathcal{D}$ and range $\mathcal{R}$. With L_2 sensitivity of $S_f = \max_{D\sim D'} \|f(D) - f(D')\|_2$, the mechanism f satisfies (ϵ,δ)-differential privacy if $\mathcal{M}(D) \triangleq f(D) + \mathcal{N}(0, S_f^2 \cdot \sigma^2)$, where $\sigma \geq \frac{S_f}{\epsilon}\sqrt{2ln(1.25/\delta)}$.

2) DP Stochastic Gradient Descent (DPSGD).: DPSGD [23] is a widely used algorithm that applies differential privacy to deep learning in a plug-and-play form, making it highly compatible for direct insertion into SGD. The key points of DPSGD include a) clipping per-sample gradients to limit the contribution of each sample, b) adding Gaussian noise to further obscure the aforementioned contributions. Moreover, the optimization method introduces the concept of moment to track privacy loss and achieve a precise estimate of the privacy loss at a manageable computational cost.

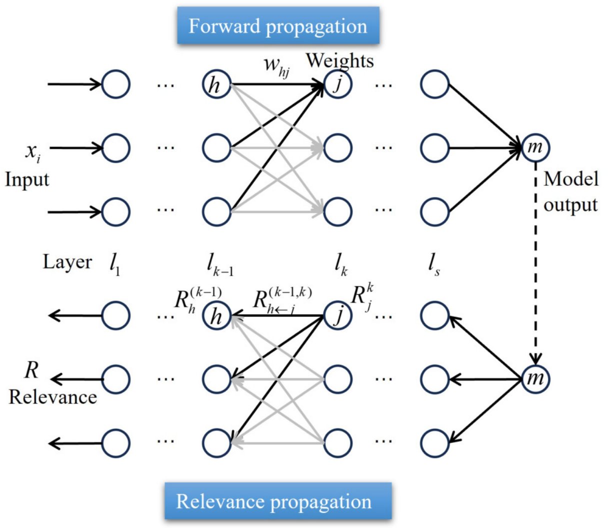


Fig. 1. The procedure of relevance propagation. The upper part of the figure shows the forward propagation process for computing the model's output, while the lower part illustrates the relevance back propagation process based on the output. The black lines represent the operation from neurons in l_{k-1} to the j-th neuron in l_k, which corresponds to our example in equations (2)–(4). We have rendered the other lines in gray. $R_j^{(k)}(x_i)$ represents the relevance of j-th neuron in the k-th layer, and $R_{h\leftarrow j}^{(k-1,k)}$ represents the relevance of the j-th neuron in the k-th layer to the connected h-th neuron in the $(k-1)$-th layer.

B. Layer-Wise Relevance Propagation and Its Application in Transformers

Layer-wise Relevance Propagation (LRP) is a popular interpretability method well-suited for models formulated as neural networks. Built upon the Deep Taylor Decomposition (DTD) [14], it can decompose the model output into pixel relevance layer by layer, reflecting the contribution of each input feature to the output.

Definition 2 (Relevance Propagation): As shown in Fig. 1, consider the input $x_{i\theta} \in x_i \subseteq \mathbb{R}^d$ passing through a model $f(w)$ containing s hidden layers $l_1, l_2, \ldots l_s$ to get the output $f_{x_i}(w)$, where $R_j^{(k)}(x_i)$ represents the relevance of j-th neuron in the *k-th layer to the model output $f_{x_i}(w)$, and $R_{h\leftarrow j}^{(k-1,k)}$ represents the relevance of j-th neuron in the k-th layer to the connected h-th neuron in the $(k-1)$-th layer. This relevance can be propagated between layers:*

$$R_h^{(k-1)}(x_i) = \sum_{j\in l_k} R_{h\leftarrow j}^{(k-1,k)}(x_i) \tag{2}$$

To propagate the relevance backward, the relevance between the model output m and the neurons of the last layer l_s is calculated as follows:

$$R_j^{(s)}(x_i) = f_{x_i}(w) \cdot \left(\alpha \frac{\mathcal{Z}_{jm}^+}{\mathcal{Z}_m^+} + \beta \frac{\mathcal{Z}_{jm}^-}{\mathcal{Z}_m^-} \right) \tag{3}$$

The calculations of $\mathcal{Z}$ values in the above equation are described as follows: let v_j represent the value of neuron j in layer l_s, w_{hj} represent the weight between neuron j and neuron h, and b_j denote the biased term, then the $\mathcal{Z}$ values are calculated as $\mathcal{Z}_{hj} = v_j w_{hj}$, $\mathcal{Z}_j = \sum_h \mathcal{Z}_{hj} + b_j$, respectively. To avoid the relevance from taking unbounded values due to small weights, we treat the negative and positive pre-activations separately. Let $\mathcal{Z}_j^+ = \sum_h \mathcal{Z}_{hj}^+ + b_j^+$, $\mathcal{Z}_j^- = \sum_h \mathcal{Z}_{hj}^- + b_j^-$, where "+" and "-" denote the positive and negative parts of the $\mathcal{Z}_{hj}$ and b_j. Then, the decomposition process of relevance can be described as:

$$R_{h\leftarrow j}^{(s-1,s)}(x_i) = R_j^{(s)}(x_i) \cdot \left(\alpha \frac{\mathcal{Z}_{hj}^+}{\mathcal{Z}_j^+} + \beta \frac{\mathcal{Z}_{hj}^-}{\mathcal{Z}_j^-} \right) \tag{4}$$

where $\alpha + \beta = 1$.

By backpropagating the relevance from one layer to another, we can attain the relevance between the model output and the input features. During the propagation, a rule must be followed: the sum of the relevance of the neurons in each layer should be consistent:

$$f_{x_i}(w) = \sum_{j\in l_s} R_j^{(s)}(x_i) = \cdots = \sum_{x_{i\theta}\in x_i} R_{x_{i\theta}}(x_i) \tag{5}$$

where $R_{x_{i\theta}}(x_i)$ denotes the relevance between the input attribute x_i and the model output.

C. Parameter Efficient Fine-Tuning (PEFT)

To efficiently train on downstream task data, parameter efficient fine-tuning (PEFT) updates only a small subset of parameters to achieve similar or sometimes even better effect compared to full parameters fine-tuning. Existing PEFT methods can be classified into three types, listed in descending order of their prevalence: additive methods, selective methods and reparameterization-based methods [3]. Additive methods, such as Adapters [2], Soft prompts [24], [25], and (IA)^3 [26], add extra parameters to the original pre-trained model and update only these new parts during the fine-tuning process. Selective methods, like BitFit [27], fine-tune only a few parameters in the pre-trained model, sometimes even ignoring the specific structure. Reparameterization-based methods are considered the most popular PEFT method approach, such as LoRA [28]. These methods reparameterize the original parameters by a specific strategy, e.g. employ a simple low-rank matrix to reparameterize the original large model, leveraging the low-rank characteristic of neural networks to significantly reduce the number of parameters that need to be updated.

IV. Problem Statement

A. Problem Formulation

Suppose $f(W_{PT}; x)$ is a pre-trained model where W_{PT} is the pre-trained parameters with N layers, and x refers to the input samples. The private datasets used in the fine-tuning process are $D_{FT} = \{x_1, x_2, \cdots, x_K\}$. The trainable parameters in parameter-efficient fine-tuning are denoted by θ where $dim(\theta) \ll dim(W_{PT})$. The relationship between θ and W_{PT} depends on the specific type of PEFT method being used. For Additive methods, θ represents the additional parameters augmented to the existing pre-trained model, distinct from W_{PT}. For Selective methods, θ represents the selected parameters from the existing pre-trained model, included in W_{PT}. For Reparameterization-based methods, θ represents parameters integrated into the original model after reparameterization transformation of a partial pre-trained model W_{PT}.

During the fine-tuning process, differential privacy noise is added only to the trainable parameters. The trainable parameters are initialized as θ_0 and our private fine-tuning model can be described as $f(W_{PT}, \theta; x)$. Our objective is to minimize the loss function $\mathcal{L}(\theta) = \frac{1}{|D_{FT}|}\sum_i \mathcal{L}(W_{PT}, \theta; x)$.

B. Threat Model and Design Objectives

Threat Model: Suppose that the fine-tuning datasets D_{FT} contain sensitive information. We adopt a *Black-box* access threat model [10], [29], [30], where the attackers have access only to the output of the model and cannot obtain the model's inner weights or structure. The attacker attempts to infer the sensitive embedding information about the target sequences by querying the target model's inference results. In addition, the attacker has access to publicly available prior knowledge and can collect a limited auxiliary dataset drawn from the same distribution as D_{FT}.

Design Objectives: 1) **Privacy.** Our first objective is to protect the sensitive information contained in the fine-tuning datasets D_{FT} against black-box inference attacks and to minimize the risk of generating tokens that reveal private information, even if attackers possess background knowledge or perform multiple access attempts. Additionally, we aim to integrate a noise perturbation mechanism that satisfies differential privacy constraints, thereby ensuring individual-level protection. 2) **Utility.** Aligned with our privacy goals, we aim to mitigate potential losses in model utility caused by differential privacy perturbations. We want to ensure that our model, fine-tuned with the adaptive noises achieves higher utility than the model fine-tuned with standard noise perturbation while maintaining a comparable level of privacy protection across various fine-tuning tasks.

V. Our Approach

A. Overview

The overall framework of Rap-FT is shown in Fig. 2. There are two modules in the Rap-FT Framework, namely relevance analysis of the trainable parameters in PEFT and adaptive gradient perturbation.

The first module is to identify the significant parameter dimensions in the gradient space. We employ a layer-wise relevance propagation method that is well-suited for

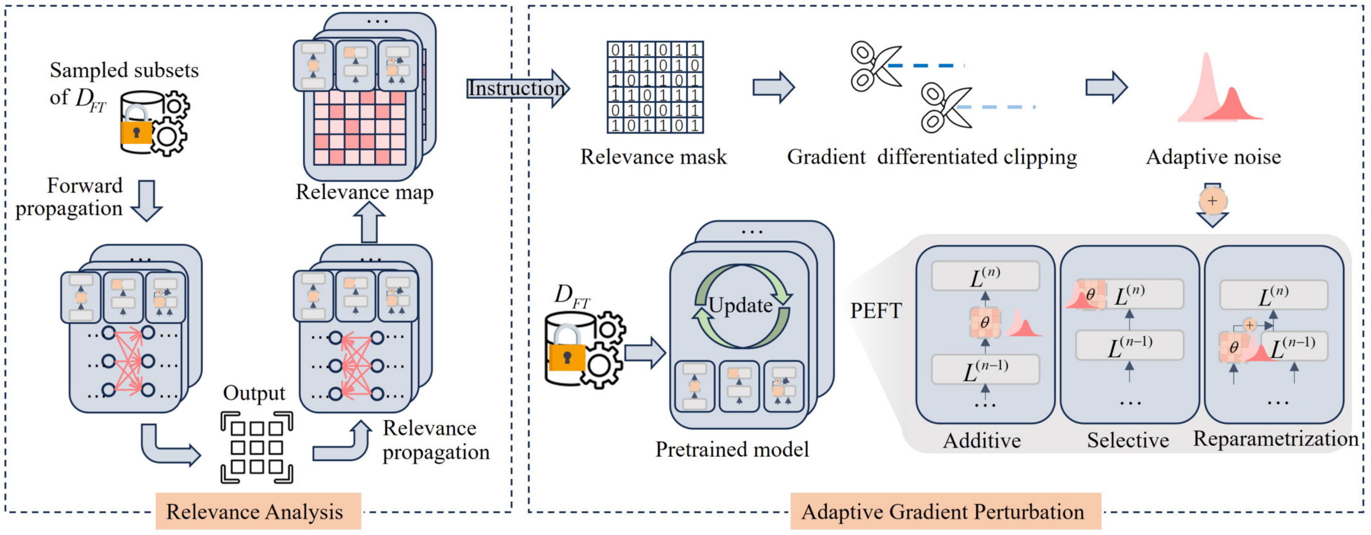


Fig. 2. Rap-FT Framework. The whole framework consists of two modules. The Relevance Analysis module generates relevance map needed for dimension splitting of the gradient space in Adaptive Gradient Perturbation module. $L^{(n)}$ denote the n-th layer.

transformer-based models [31] to analyze the relevance between the fine-tuning datasets and the model output. This process takes into account the structure of trainable parameters across different PEFT methods. Specifically, we first extend the relevance analysis to the generative task and reverse the direction of forward propagation to compute relevance based on different types of PEFT network structures, generating a relevance map of the trainable parameters. This process enables us to identify and mark parameters with high relevance as more important, and vice versa.

In the second module, our proposed adaptive gradient perturbation strategy is used to adaptively add noise to gradients. Firstly, a relevance mask is generated based on the normalized relevance map to distinguish between dimensions of trainable parameters with varying levels of importance. Then, using a masking approach, we split the gradients generated by trainable parameters into two subsets: strongly significant gradients and weakly significant gradients, enabling differentiated processing in subsequent steps. Gradient perturbations are handled separately for each subset. In each iteration, the gradient subsets across different dimensions are divided based on the mask. We apply differentiated gradient clipping to the two subsets, and generate adaptive gaussian noise according to the different noise intensity of each subset, with gradients of lower relevance undergoing more perturbation. This adaptive gradient perturbation process repeats in a loop until the optimal fine-tuned model is obtained.

B. Relevance Analysis of the Trainable Parameters in PEFT

To bias the additive noise process towards less important trainable parameters in PEFT, we adopt and enhance the interpretability method [31] in three ways: 1) extending the method to encompass generative tasks and developing a relevance initialization approach tailored for this purpose, 2) optimizing the relevance propagation process based on different PEFT methods, and 3) generating a more focused relevance map by considering the importance of trainable parameters in PEFT methods.

Existing perturbation strategies that incorporate LRP-based interpretability methods are primarily designed for classification tasks [15], [16], where the output can be transformed into a one-hot matrix indicating the target class and directly used as input for relevance analysis. To extend relevance analysis to generative tasks, the naive method feeds the model's softmax outputs directly into the relevance analysis function. However, the large vocabulary size makes this process susceptible to the curse of dimensionality. To address this, we propose an alternative approach to extract key information based on decoding strategies to initialize the model output, which serves as input for relevance analysis, providing an efficient option for reducing memory overhead without compromising analytical accuracy:

1) *Top-k Initialization:* For each token, select a threshold k and retain only the Top-k probabilities from the word list, and set all others to zero.
2) *Top-p Initialization:* For each token, retain words in the descending list until their cumulative probability reaches a threshold p (e.g., 0.95), setting the rest to zero.
3) *Beam Initialization:* Following the sentence order, retain the top tokens at each step, then calculate the cumulative probabilities and select the top b beams with the highest values. Unselected tokens are set to 0.

These methods ensure that the initialized relevance results avoid huge parameter counts for generative model output, and focus on the more important tokens in the model output.

Based on the relevance initialization, we incorporate the interpretability method [31] and further develop a relevance analysis process tailored to the three major categories of PEFT methods. We provide instantiations for each of the three categories. As far as we know, this is the first exploration of interpretability methods on PEFT.

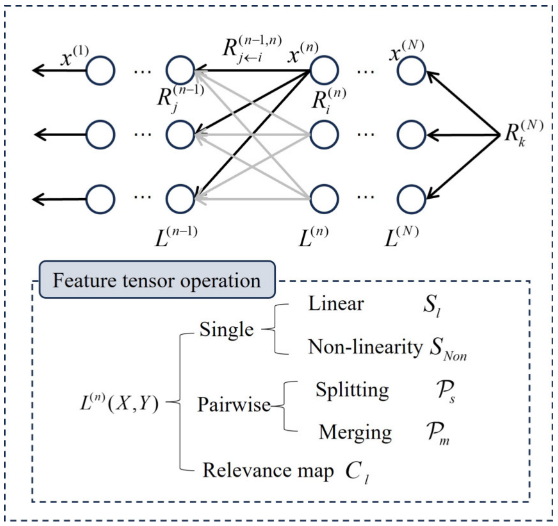


Fig. 3. The procedure of relevance analysis. The top part of the figure displays the relevance analysis procedure at the neuron level, with the specific types of feature tensor operations shown below it. $R_k^{(N)}$ is the result of the relevance initialization with input x_k and $L^{(n)}(X,Y)$ refers to the operation of the input feature map and weights for layer $L^{(n)}$,.

As shown in the top of Fig. 3, suppose the pre-trained model has N layers, and let $x^{(n)}$, where $n \in [1, \cdots, N]$ denotes the input of $L^{(n)}$. Without loss of generality, let $L^{(n)}(X,Y)$ denote the operation of layer $L^{(n)}$ on two tensors X, Y. We can divide the layers in a transformer-based network into two types based on their operations in relevance computation: single feature tensor operations and pairwise feature tensor operations.

Let $R_k^{(N)}$ be the result of the relevance initialization with input x_k, and we can abbreviate it as $R^{(N)}$. For single feature tensor operations, the two tensors X and Y in operation $L^{(n)}(X,Y)$ refer to the input feature map and weights for layer $L^{(n)}$, respectively. These operations can be further categorized into linear layers and non-linearity activation layers.

In linear layers $L^{(n)}$, according to the Deep Taylor Decomposition, the relevance propagation can be written in the form of the chain rule:

$$\begin{aligned} R_j^{(n-1)} &= \mathcal{S}_l\left(X, Y, R^{(n)}\right) \\ &= \sum_j X_j \frac{\partial L_i^{(n-1)}(X,Y)}{\partial X_j} \frac{R_i^{(n)}}{L_i^{(n-1)}(X,Y)} \end{aligned} \tag{6}$$

where $L_i^{(n-1)}(X,Y) = \sum_i x_j w_{ji} + b_j$, i represents the index of elements in $R^{(n)}$, and j represents the index of elements in $R^{(n-1)}$.

The non-linearity activations involve two scenarios regarding whether they output both positive and negative values. For instance, ReLU only outputs positive values, and it introduces non-linearity by returning zero for negative input values and passing through positive input values unchanged. For other non-linearity methods that output both positive and negative values, like GeLU, a subset of indices $q = \{(i,j) \mid x_j w_{ji} > 0\}$ is extracted to propagate the relevance. The two scenarios are presented as follows:

$$\begin{aligned} R_j^{(n-1)} &= \mathcal{S}_{\text{Non}}\left(x, w, R^{(n)}\right) \\ &= \begin{cases} \sum_i \frac{x_j^+ w_{ji}^+}{\sum_{j'} x_{j'}^+ w_{j'i}^+} R_i^{(n)} & \text{Relu} \\ \sum_{\{i|(i,j)\in q\}} \frac{x_j w_{ji}}{\sum_{\{j'((j',i)\in q\}} x_{j'}, w_{j'i}} R_i^{(n)}, & \text{Others} \end{cases} \end{aligned} \tag{7}$$

For pairwise feature tensor operations, such as skip connections and matrix multiplications, both tensors X and Y in operation $L^{(n)}(X,Y)$ represent feature tensors. There are two types of operations: relevance splitting and merging. And the relevance splitting for X, Y can be calculated as follows:

$$\begin{aligned} R_j^{X^{(n-1)}}, R_q^{Y^{(n-1)}} &= \mathcal{P}_s\left(X, Y, R^{(n)}\right) \\ &= \begin{cases} R_j^{X^{(n-1)}} = \mathcal{S}\left(X, Y, R^{(n)}\right) \\ R_q^{Y^{(n-1)}} = \mathcal{S}\left(Y, X, R^{(n)}\right) \end{cases} \end{aligned} \tag{8}$$

Recalling the rule that the sum of relevance in each layer should be consistent, it holds that:

$$f(w) = \sum_{j \in l_s} R_j^{(N)} = \cdots = \sum_{x_{i\theta} \in x^{(1)}} R_{x_{i\theta}^{(1)}} \tag{9}$$

In pairwise feature tensor operation of transformers, the conservation rule should be:

$$\sum_j R_j^{X^{(n-1)}} + \sum_q R_q^{Y^{(n-1)}} = \sum_i R_i^{(n)} \tag{10}$$

Subsequently, it is necessary to add normalization to $R_j^{X^{(n-1)}}$ and $R_q^{Y^{(n-1)}}$ to address any inconsistence with the sum of relevance that may arise from pairwise feature tensor operations:

$$\bar{R}_j^{X^{(n-1)}} = R_j^{X^{(n-1)}} \frac{\left|\sum_j R_j^{X^{(n-1)}}\right|}{\left|\sum R\right|} \cdot \frac{\sum_i R^{(n)}}{\sum_j R_j^{X^{(n-1)}}} \tag{11a}$$

$$\bar{R}_q^{Y^{(n-1)}} = R_q^{Y^{(n-1)}} \frac{\left|\sum_q R_q^{Y^{(n-1)}}\right|}{\left|\sum R\right|} \cdot \frac{\sum_i R^{(n)}}{\sum_q R_q^{Y^{(n-1)}}} \tag{11b}$$

where $|\sum R| = |\sum_j R_j^{X^{(n-1)}}| + |\sum_q R_q^{Y^{(n-1)}}|$. For classification tasks, we have $\sum_i R^{(n)} = 1$, and for generative tasks, $\sum_i R^{(n)}$ depends on the sum of the results of relevance initialization.

When propagation involves merging two matrix relevance, it is calculated as:

$$\begin{aligned} R^{(n-2)} &= \mathcal{P}_m\left(X, \left(\bar{R}'^{X^{(n-1)}}_j, \bar{R}'^{Y^{(n-1)}}_q\right)\right) \\ &= X_j \left[\left.\frac{\partial X}{\partial X}\right|_{\frac{R'^{X^{(n-1)}}_j}{X_j}} + \left.\frac{\partial X}{\partial X}\right|_{\frac{R'^{Y^{X_j}}_{r\,Y(n-1)}}{X_j}} \right] \end{aligned} \tag{12}$$

where $\bar{R}'^{X^{(n-1)}}_j, \bar{R}'^{Y^{(n-1)}}_q$ refer to the results after normalization and propagation through different network structures, and $\left.\frac{\partial X}{\partial X}\right|_{\frac{\bar{R}_j^{X^{(n-1)}}}{X_j}}$ refers to Jacobi product.

Furthermore, we integrate the relevance analysis with the trainable parameters to make it adaptable to any PEFT method In other words, while reverse-propagating the relevance of the entire model structure in the previous step, we simultaneously extract the relevance associated with the trainable parameters

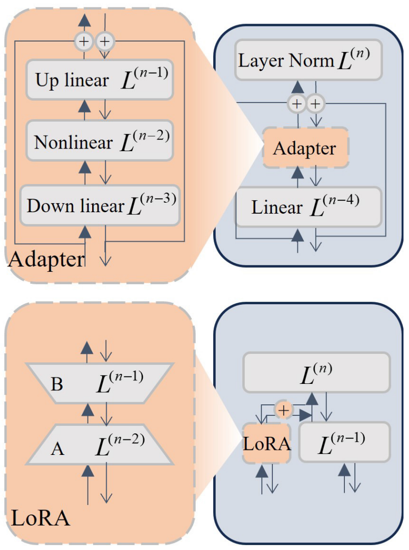


Fig. 4. Relevance propagation of Adapter and LoRA. The triangular arrows indicate the direction of forward propagation, while the regular arrows represent the direction of relevance propagation. $L^{(n)}$ denotes the n-th layer.

of PEFT. Therefore, we compute the relevance matrices of the trainable parameters and integrate them throughout the network structure to form a relevance map.

Let $R^{(n),\theta}$ denote the relevance of trainable layer $L^{(n)}$. Considering that in single feature tensor operations, Y refers to the weights for layers, so we perform a similar computation on Y:

$$R_j^{(n-1),\theta} = \mathcal{C}_l\left(X, Y, R^{(n)}\right) = \mathcal{S}_l\left(Y, X, R^{(n)}\right) = \sum_j Y_j \frac{\partial L_i^{(n-1)}(X,Y)}{\partial Y_j} \frac{R_i^{(n)}}{L_i^{(n-1)}(X,Y)} \tag{13}$$

To account for the variations in relevance among different transformer blocks in the network, we document the relevance of the trainable parameters in the entire model structure, and concatenate them to generate a relevance map.

Next, we demonstrate how the proposed relevance analysis is instantiated in three types of PEFT methods.

Additive Instantiation: We start with the Additive methods, in which extra parameters or layers are introduced into the existing pre-trained model. Consequently, it is necessary to add these extra parameters into the model architecture and perform sequential relevance propagation when analyzing the relevance of the trainable parameters. Taking adapters [2] as an example, as shown in the upper part of Fig. 4, we compute the relevance based on the operations of different layers within the adapter layer:

$$R_{up}^{(n)}, R_{extra1}^{(n)} = \mathcal{P}_s(X, Y, R_a^{(n)}) \tag{14a}$$
$$R_{up}^{(n-1)} = \mathcal{S}_l(X, Y, R_{up}^{(n)}) \tag{14b}$$
$$R_{Non}^{(n-2)} = \mathcal{S}_{Non}(x, w, R_{up}^{(n-1)}) \tag{14c}$$
$$R_{down}^{(n-3)} = \mathcal{S}_l(X, Y, R_{Non}^{(n-2)}) \tag{14d}$$
$$R^{(n-4)} = \mathcal{P}_m(X, (R_{down}^{(n-3)}, R_{extra1}^{(n)})) \tag{14e}$$

Furthermore, to construct the relevance map, we compute the relevance for the adapter layers and concatenate them according to the order of forward propagation in the model after the analysis:

$$R_{up}^{(n-1),\theta} = \mathcal{C}_l(X, Y, R_{up}^{(n)}) \tag{15a}$$
$$R_{down}^{(n-3),\theta} = \mathcal{C}_l(X, Y, R_{Non}^{(n-2)}) \tag{15b}$$

Next, we use the relevance of the adapter parameters in each transformer block in the generation of the trainable parameter relevance map $\mathcal{R}_{Adapter} = \{R_{down}^{(b),\theta}, R_{up}^{(b),\theta}, b \in [1, B]\}$, where b denotes the index of transformer blocks.

Selective Instantiation: The instantiation of Selective PEFT methods involves minimal changes to the model architecture. Since the trainable parameters are part of the original network, we only need to perform sequential relevance propagation according to the original network structure and compute the relevance of the selected parameters. Taking BitFit as an example, we retain the relevance of the bias in the network and concatenate them to generate a relevance map. Let W_b denote the bias of layer $L^{(n)}$, the relevance can be computed as:

$$R_{bias}^{(n-1),\theta} = \mathcal{C}_l(X, Y_b, R^{(n)}) \tag{16}$$

Subsequently, the relevance of all biases is used to construct a relevance map $\mathcal{R}_{BitFit} = \{R_{bias}^{\theta}\}$. It's worth noting that biases may exist not only within transformer blocks.

Reparameterization-based Instantiation: Taking the commonly used method LoRA as an example [28], as depicted in the lower part of Fig. 4, the relevance propagation of trainable parameters is as follows:

$$R_B^{(n-1)} = \mathcal{S}_l(X, Y, R_{lora}^{(n)}) \tag{17a}$$
$$R_A^{(n-2)} = \mathcal{S}_l(X, Y, R_B^{(n-1)}) \tag{17b}$$
$$R_B^{(n-1),\theta} = \mathcal{C}_l(X, Y, R_{lora}^{(n)}) \tag{17c}$$
$$R_A^{(n-2),\theta} = \mathcal{C}_l(X, Y, R_B^{(n-1)}) \tag{17d}$$

Likewise, we document the relevance of the weights B and A, and assemble them to form a trainable parameter relevance map $\mathcal{R}_{Lora} = \{R_B^{(b),\theta}, R_A^{(b),\theta}, b \in [1, B]\}$.

Finally, we compute the relevance based on the category that a given PEFT method belongs to [3], which represents the contribution of the input features to the model output.

C. Adaptive Gradient Perturbation

The core idea of our adaptive perturbation strategy is to differentially add noise rather than uniformly applying it across all dimensions. In our approach, higher levels of Gaussian noise are applied to dimensions that are less significant to model utility (i.e., weakly significant gradients), while lower levels of noise are applied to dimensions that are more critical to utility (i.e., strongly significant gradients), thereby enhancing model performance while maintaining overall privacy protection. Based on the relevance map of the trainable parameters, we generate a relevance mask to split the gradients into two subsets, differentiating between dimensions of varying importance. Then, we perform adaptive gradient clipping and generate noise with different intensities for the two subsets. Notably, our proposed method adjusts the standard deviation σ

Algorithm 1 Adaptive Gradient Perturbation Strategy

```
Input: Fine-tuning datasets $D_{FT} = \{x_1, x_2, \cdots, x_K\}$, learning rate $\eta_t$, noise scale $\sigma$, relevance map $\mathcal{R}_{PEFT}$, batch size $B$, irrelevance threshold $\beta$, initialized trainable parameters $\theta_0$.
Output: The optimal model parameter $f(W_{PT}, \theta^*; x)$
1:  Compute the average relevance of the trainable parameters:
2:  $\mathcal{R}_{PEFT} = \frac{1}{|\mathcal{D}_s|}\sum_{x_k \in \mathcal{D}_s} \mathcal{R}_{PEFT}(x_k)$
3:  Min-max normalization:
4:  $\mathcal{R}_l^j = \frac{\mathcal{R}_l^j - \mathcal{R}_l^{min}}{\mathcal{R}_l^{max} - \mathcal{R}_l^{min}}$
5:  Compute threshold index value: $I_r = \lceil \beta \times |\mathcal{R}_{PEFT}| \rceil$.
6:  Rank the relevance map $\mathcal{R}'_{PEFT}$.
7:  Find the relevance threshold: $\mathcal{R}_{ir} = \mathcal{R}'_{PEFT}[I_r]$.
8:  Generate relevance mask $\mathcal{M}_R$:
9:  $\mathcal{M}_R = \begin{cases} 1 & \text{if } \mathcal{R}_{PEFT} \le \mathcal{R}_{ir} \\ 0 & \text{if } \mathcal{R}_{PEFT} > \mathcal{R}_{ir} \end{cases}$
10: for $t \in [T]$ do
11:     Randomly select a batch of samples $X_t$ from $D_{FT}$.
12:     for $x_k \in X_t$ do
13:         Compute $g_t(x_k) \leftarrow \nabla_{\theta_t} \mathcal{L}(W_{PT}, \theta_t; x_k)$
14:         Clip gradient:
15:         $g_w = \mathcal{M}_R \times g_t(x_k)$, $g_s = (1 - \mathcal{M}_R) \times g_t(x_k)$
16:         $\bar{g}(x_k) \leftarrow \begin{cases} g_w / max(1, \frac{\|g_w\|_2}{C \cdot (|g_w|/|g|)^2}) \\ g_s / max(1, \frac{\|g_s\|_2}{C \cdot (|g_s|/|g|)^2}) \end{cases}$
17:     end for
18:     Add noise:
19:         $\tilde{g}_t = \frac{1}{B}(\sum_{x_k \in D} \bar{g}_t(x_k) + \mathcal{N})$
20:         where $\mathcal{N} = \begin{cases} \mathcal{M}_R \times \mathcal{N}(0, \sigma_w^2 C^2 I) \\ (1 - \mathcal{M}_R) \times \mathcal{N}(0, \sigma_s^2 C^2 I) \end{cases}$
21:     $\sigma_w = \sigma \sqrt{\text{rac|gw||g|}}$, $\sigma_s = \sigma \sqrt{\text{rac|gs||g|}}$
22:     $\theta_{t+1} = \theta_t - \eta_t \tilde{g}_t$
23: end for
24:     return $f(W_{PT}, \theta^*; x)$
```

of Gaussian noise at a macro level, which directly controls the intensity of noise added to each part. Meanwhile, it indirectly manages the privacy budget allocated for each subset. The more noise added, the smaller the privacy budget allocated to that subset, resulting in a higher level of privacy protection. We aim to apply a higher level of privacy protection to subsets that contribute less to overall utility.

As shown in Algorithm 1, $\mathcal{R}_{PEFT}$ is the general symbol for the relevance map of the trainable parameters across different PEFT methods, consisting of layer-wise relevance values propagated from the model output. For the fine-tuning datasets $D_{FT} = \{x_1, x_2, \cdots, x_K\}$, $\mathcal{R}_{PEFT}(x_k)$ denotes the relevance map generated for input x_k, Specifically, it represents a collection of layer-wise relevance for trainable parameters propagated from the model output. We compute relevance on a randomly sampled subset of the fine-tuning dataset $\mathcal{D}_s = \{x_1, x_2, \cdots, x_{K_s}\}$, obtaining an averaged relevance map:

$$\mathcal{R}_{PEFT} = \frac{1}{|\mathcal{D}_s|} \sum_{x_k \in \mathcal{D}_s} \mathcal{R}_{PEFT}(x_k) \tag{18}$$

To ensure that relevance $\mathcal{R}_{PEFT} \in [0, 1]$, we apply min-max normalization for each layer in $\mathcal{R}_{PEFT}$: $\mathcal{R}_l^j = \frac{\mathcal{R}_l^j - \mathcal{R}_l^{min}}{\mathcal{R}_l^{max} - \mathcal{R}_l^{min}}$ where j is the j-th feature in the l-th layer.

Then, we rank the relevance for each layer and set an irrelevance ratio $\beta \in (0.5, 1]$ to divide the ranked relevance map $\mathcal{R}'_{PEFT}$ into two subsets dimensionally. After calculating the threshold index value $I_r = \lceil \beta \times |\mathcal{R}_{PEFT}| \rceil$, we can find the relevance threshold $\mathcal{R}_{ir} = \mathcal{R}'_{PEFT}[I_r]$ from the ranked relevance. Next, we generate a relevance mask to determine how to add perturbations to the gradient of each dimension subset:

$$\mathcal{M}_R = \begin{cases} 1 & \text{if } \mathcal{R}_{PEFT} \le \mathcal{R}_{ir} \\ 0 & \text{if } \mathcal{R}_{PEFT} > \mathcal{R}_{ir} \end{cases} \tag{19}$$

During the private fine-tuning process, we randomly select a batch from D_{FT} in each iteration and compute the per-sample gradient. Then, we divide the gradient into two subsets dimensionally within each iteration t:

$$\begin{cases} g_w = \mathcal{M}_R \times g_t(x_k) \\ g_s = (1 - \mathcal{M}_R) \times g_t(x_k) \end{cases} \tag{20}$$

Based on this, instead of using a single L_2 norm C for gradient clipping, we apply differentiated clipping to each gradient subset. Using the ratios of the two subsets to the total gradient, $\frac{|g_w|}{|g|}$ and $\frac{|g_s|}{|g|}$, we derive the L_2 norm upper bounds for the two parts, $C_w = C \cdot (|g_w|/|g|)^2$ and $C_s = C \cdot (|g_s|/|g|)^2$, respectively. Next, we perform differentiated clipping on the gradient subsets by applying these bounds:

$$\bar{g}(x_k) \leftarrow \begin{cases} g_w / max\left(1, \dfrac{\|g_w\|_2}{C_w}\right) \\ g_s / max\left(1, \dfrac{\|g_s\|_2}{C_s}\right) \end{cases} \tag{21}$$

Thus, the proposed differentiated clipping not only bypasses the huge computational overhead of dimension-by-dimension clipping in [13], but also enables an adaptive noise distribution that varies according to different gradient dimensions.

After getting the clipped gradient for each sample, we generate the adaptive Gaussian noise as $\mathcal{N}(0, \sigma_w^2 C^2 I)$ for weakly significant gradients and $\mathcal{N}(0, \sigma_s^2 C^2 I)$ for strongly significant gradients, where:

$$\begin{cases} \sigma_w = \sigma \sqrt{\text{rac|gw||g|}} \\ \sigma_s = \sigma \sqrt{\text{rac|gs||g|}} \end{cases} \tag{22}$$

Given $\beta \in (0.5, 1]$, it can be guaranteed that $|g_w| > |g_s|$, ensuring $\sigma_w > \sigma_s$. This indicates that more intense noise is added to the weakly significant gradients. In this way, we can indirectly control the allocation of the privacy budget ϵ through noise intensity, imposing stricter privacy protections on less important dimensions. Consequently, we mask the clipped gradients to add adaptive noise.

$$\tilde{g}_t = \frac{1}{B}\left(\sum_{x_k \in D} \bar{g}_t(x_k) + \mathcal{N}\right) \tag{23}$$

where $\mathcal{N} = \begin{cases} \mathcal{M}_R \times \mathcal{N}(0, \sigma_w^2 C^2 I) \\ (1 - \mathcal{M}_R) \times \mathcal{N}(0, \sigma_s^2 C^2 I) \end{cases}$.

Specifically, $\mathcal{N}(0, \sigma_w^2 C^2 I)$ noise is added to weakly significant gradients, introducing more perturbations, while $\mathcal{N}(0, \sigma_s^2 C^2 I)$ noise is added to strongly significant gradients, minimizing redundant noise to mitigate its impact on model utility. Finally, the adaptive perturbed gradients are used to update the trainable parameters, and this iterative process continues until the desired fine-tuned model for the vertical domain is achieved.

VI. Privacy Analysis

The privacy proof of our proposed Rap-FT is given below.

Theorem 1: For $f : N^x \rightarrow R^m$ and neighboring databases D, D', with L_2 sensitivity defined as $S_f = \max_{D \sim D'} \|f(D) - f(D')\|_2$, we have $S_f \leq 1$. Given that the mechanism $\mathcal{M}(D) = f(D) + Z$, where $Z \sim \mathcal{N}(0, S_f \cdot \sigma^2)$ satisfies (ϵ, δ)-differential privacy, the mechanism $\mathcal{M}'(D) = f'(D) + Z'$,where $f' : N^x \rightarrow R^m$ and $Z' = (z_1', \cdots, z_m')^T$, $\forall j' \in m_A, z'_{j'} \sim \mathcal{N}(0, \sigma_A^2), \forall j'' \in m_B, z'_{j''} \sim \mathcal{N}(0, \sigma_B^2)(m_A \cup m_B = m, m_A \cap m_B = \emptyset)$, is also (ϵ, δ)-Differentially Private (DP) if the followings hold:

$$\sigma_j = \begin{cases} \sigma \sqrt{|m_A||m|} & \text{if } j \in m_A \\ \sigma \sqrt{|m_B||m|} & \text{if } j \in m_B \end{cases} \tag{24}$$

$\forall j' \in m_A, \|v_{j'}\| \leq S_f \cdot (|m_A|/|m|)^2, \forall j'' \in m_B, \|v_{j''}\| \leq S_f \cdot (|m_B|/|m|)^2$.

Proof: Firstly, according to the Gaussian mechanism [13], [32], [33], we know that for neighboring datasets D, D', a mechanism $\mathcal{M}$ is (ϵ, δ) -DP if and only if the privacy loss condition holds:

$$P_r\left(l_{M,D,D'} \geq \epsilon\right) - e^{\epsilon} P_r\left(l_{M,D,D'} \leq -\epsilon\right) < \delta \tag{25}$$

where $l_{M,D,D'} = \frac{P_r(\mathcal{M}(D)=o)}{P_r(\mathcal{M}(D')=o)}$.

We extend the privacy loss to functions in R^m. Consider the worst case where $(v_1, \cdots, v_m) = f(D) - f(D')$, let $(x_1, \cdots, x_m) = o - f(D)$, then we have $(x_1 + v_1, \cdots, x_m + v_m) = o - f(D')$. In this context, we have the privacy loss:

$$\begin{aligned}\frac{Pr\left(\mathcal{M}\left(D\right)=z\right)}{Pr\left(\mathcal{M}\left(D'\right)=z\right)} &= \left| ln \frac{\prod_{j=1}^{m} \exp\left(-\frac{1}{2\sigma_j^2}\|x-\mu\|^2\right)}{\prod_{j=1}^{m} \exp\left(-\frac{1}{2\sigma_j^2}\|x+v-\mu\|^2\right)} \right| \\ &= \left| ln \frac{\exp\left(\sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\|x-\mu\|^2\right)}{\exp\left(\sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\|x+v-\mu\|^2\right)} \right| \\ &= \left| ln \exp \sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\left[\|x-\mu\|^2 - \|x+v-\mu\|^2\right] \right| \\ &= \left| \sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\left[\|x\|^2 - \|x+v\|^2\right] \right| \\ &= \left| \sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\left[\left(x_j^2 - \left(x_j+v_j\right)^2\right)\right] \right| \\ &= \left| \sum_{j=1}^{m} -\frac{1}{2\sigma_j^2}\left[\left(2x_jv_j + v_j^2\right)\right] \right| \\ &= \left| \sum_{j=1}^{m} \frac{x_jv_j}{\sigma_j^2} + \sum_{j=1}^{m} \frac{v_j^2}{2\sigma_j^2} \right| \end{aligned} \tag{26}$$

When x is chosen from Z' which follows two Gaussian distributions such that $\forall j' \in [m_A], z_{j'}' \sim \mathcal{N}(0, \sigma_A^2)$ and $\forall j' \in [m_B], z_{j''}' \sim \mathcal{N}(0, \sigma_B^2)$, we have privacy loss $l_{M,D,D'} = \sum_{j=1}^{m} \frac{x_jv_j}{\sigma_j^2} + \sum_{j=1}^{m} \frac{v_j^2}{2\sigma_j^2}$, which also follows a Gaussian distribution $l_{M,D,D'} \sim \mathcal{N}(\sum_{j=1}^{m_A} \frac{v_j^2}{2\sigma_A^2} + \sum_{j=1}^{m_B} \frac{v_j^2}{2\sigma_B^2}, \sum_{j=1}^{m_A} \frac{v_j^2}{\sigma_A^2} + \sum_{j=1}^{m_B} \frac{v_j^2}{\sigma_B{}^2})$. To facilitate subsequence proof, let H denote $\sum_{j=1}^{m_A} \frac{v_j^2}{\sigma_A^2} + \sum_{j=1}^{m_B} \frac{v_j^2}{\sigma_B^2}$, the privacy proof can be rewritten as $l_{M,D,D'} \sim \mathcal{N}(\frac{H}{2}, H)$ ■

Lemma 1: $P_r(l_{M,D,D'} \geq \epsilon) - e^{\epsilon} P_r(l_{M,D,D'} \leq -\epsilon)$ is monotonically increasing in H.

Proof of Lemma 1: Let us break it down term by term:

$$P_r\left(l_{M,D,D'} \geq \epsilon\right) = \int_{\epsilon}^{\infty} \frac{1}{\sqrt{2\pi H^*}} e^{-\frac{(x-H*/2)^2}{2H^{*2}}} dx \tag{27}$$

According to the standardization process of the Gaussian distribution, when $x = \epsilon$, we have $\frac{x-\mu}{\sigma} = \frac{x-\frac{H^*}{2}}{\sqrt{H^*}} = \frac{\epsilon-\frac{H^*}{2}}{\sqrt{H^*}}$. Thus, the first term can be rewritten as $P_r(l_{M,D,D'} \geq \epsilon) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{A(H)} e^{-\frac{x}{2}} dx$, where $A(H) = \frac{\sqrt{H}}{2} - \frac{\epsilon}{\sqrt{H}}$. Furthermore, we can deduce the derivative of $A(H)$ with respect to H: $A'(H) = \frac{1}{4\sqrt{H}} + \frac{\epsilon}{2H^{3/2}}$. Likewise, the second term can be rewritten as $P_r(l_{M,D,D'} \leq -\epsilon) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{B(H)} e^{-\frac{x}{2}} dx$, where $B(H) = -\frac{\sqrt{H}}{2} - \frac{\epsilon}{\sqrt{H}}$, and the derivative with respect to H can be written as: $B'(H) = -\frac{1}{4\sqrt{H}} + \frac{\epsilon}{2H^{3/2}}$. Then, we have:

$$\begin{aligned} &\frac{d\left[P_r\left(l_{M,D,D'} \geq \epsilon\right) - e^{\epsilon} P_r\left(l_{M,D,D'} \leq -\epsilon\right)\right]}{dH} \\ &= \frac{1}{\sqrt{2\pi}}\left[e^{\frac{-(A(H))^2}{2}}\left(A'(H)\right) - e^{\epsilon} e^{\frac{-(B(H))^2}{2}}\left(B'(H)\right)\right] \\ &= \frac{1}{\sqrt{2\pi}}\left[\frac{1}{4\sqrt{H}}\left(e^{\frac{-(A(H))^2}{2}} + e^{\epsilon-\frac{-(B(H))^2}{2}}\right)\right. \\ &\quad \left. + \frac{\epsilon}{2H^{3/2}}\left(e^{\frac{-(A(H))^2}{2}} - e^{\epsilon-\frac{-(B(H))^2}{2}}\right)\right] \end{aligned} \tag{28}$$

First, it is evident that the first term within the parentheses $\frac{1}{4\sqrt{H}}(e^{\frac{-(A(H))^2}{2}} + e^{\epsilon-\frac{-(B(H))^2}{2}}) \geq 0$. Since $(B(H))^2 = (A(H))^2 + 2\epsilon$, it can be deduced that the second term within the parentheses $\frac{\epsilon}{2H^{3/2}}(e^{\frac{-(A(H))^2}{2}} - e^{\epsilon-\frac{-(A(H))^2+2\epsilon}{2}}) = \frac{\epsilon}{2H^{3/2}} e^{\frac{-(A(H))^2}{2}}(1 - e^{\epsilon-\epsilon}) = 0$. This leads to the conclusion that $\frac{d[P_r(l_{M,D,D'} \geq \epsilon) - e^{\epsilon} P_r(l_{M,D,D'} \leq -\epsilon)]}{dH} \geq 0$, which means that $P_r(l_{M,D,D'} \geq \epsilon) - e^{\epsilon} P_r(l_{M,D,D'} \leq -\epsilon)$ is monotonically increasing in H. Thus, the proof is complete.

With the proof of Lemma 1 completed, we now return to the main theorem. If a function $f(\cdot)$ with $S_f \leq 1$ satisfies (ϵ, δ)-differential privacy with σ, the privacy loss variable $l^*_{M,D,D'} \sim \mathcal{N}(\frac{H^*}{2}, H^*)$, where $H^* = \sum_{j=1}^{m} \frac{(v_j^*)^2}{\sigma^2} = \frac{1}{\sigma^2}$. Since $P_r(l_{M,D,D'} \geq \epsilon) - e^{\epsilon} P_r(l_{M,D,D'} \leq -\epsilon)$ is monotonically increasing in H, equation (25) will hold if $H \leq H^*$, which means $\sum_{j=1}^{m_A} \frac{(v_j^*)^2}{\sigma_A^2} + \sum_{j=1}^{m_B} \frac{(v_j^*)^2}{\sigma_B^2} \leq \frac{1}{\sigma^2}$. In other words, if the above inequality holds, we can conclude that the m-dimensional multivariate Gaussian perturbation $Z' = (z_1', \cdots, z_m')^T$ in Theorem 1 also enables $f : N^x \rightarrow R^m$ to satisfy (ϵ, δ)-differential privacy.

At this point, it suffices to verify that the derived inequality is satisfied. Given that equal quantity of noise is added to the dimensions within the subset m_A or m_B ($|m| = |m_A| + |m_B|$), we can draw conclusions $H_A^* = \sum_{j=1}^{m_A} \frac{(v_j^*)^2}{\sigma_A^2} = \frac{S_f^A}{\sigma_A^2}$ and

$H_B^* = \sum_{j=1}^{m_B} \frac{(v_j^*)^2}{\sigma_B^2} = \frac{S_f^B}{\sigma_B^2}$ can also be drawn for the two subsets. When noise is added to the entire gradient space, based on $\|v\| \leq S_f \leq 1$, we can further derive $H^* = \frac{(v_j^*)^2}{\sigma^2} = \frac{1}{\sigma^2}$. Combining this with the conditions $\forall j' \in m_A, \|v_{j'}\| \leq S_f \cdot (|m_A|/|m|)^2, \forall j'' \in m_B, \|v_{j''}\| \leq S_f \cdot (|m_B|/|m|)^2$, we can express the left side of the inequality as:

$$\begin{aligned}
\sum_{j=1}^{m_A} \frac{v_j^2}{\sigma_A^2} + \sum_{j=1}^{m_B} \frac{v_j^2}{\sigma_B^2} &= \frac{S_f^A}{\sigma_A^2} + \frac{S_f^B}{\sigma_B^2} \\
&= \frac{S_f \cdot (|m_A|/|m|)^2}{\sigma^2 \cdot (|m_A|/|m|)} + \frac{S_f \cdot (|m_B|/|m|)^2}{\sigma^2 \cdot (|m_B|/|m|)} \\
&= \frac{S_f \cdot (|m_A|/|m|)}{\sigma^2} + \frac{S_f \cdot (|m_B|/|m|)}{\sigma^2} \\
&= \frac{S_f}{\sigma^2} \leq \frac{1}{\sigma^2} \qquad (29)
\end{aligned}$$

Based on Lemma 1, $f' : N^x \to R^m$ satisfies (ϵ, δ)-differential privacy. Proof is complete. ■

VII. Experimental Evaluation

A. Experimental Settings

1) Datasets: To demonstrate the applicability of our proposed Rap-FT, we employ one general natural language generation task (E2E) [34] and one common private fine-tuning task (Echr) [35] for evaluation: 1) E2E dataset comprises 42,000 training samples, 4,600 validation samples, and 4,600 test samples related to the restaurant domain. It is structured in a "template-like" format and can be mapped to natural language. 2) Echr is an English legal judgment prediction dataset composed of 11,500 cases from the European Court of Human Rights. It contains full descriptions of defendants' personal information, making it suitable for privacy evaluation. For our experiments, we randomly sample 42,112 training samples and 4,672 validation samples.

2) Setup: Our experiments are executed on Nvidia A800 GPU. We choose a representative method from each of the three major categories of PEFT methods: Adapter [2] for Additive PEFT methods, BitFit [27] for Selective PEFT methods and LoRA [28] for reparameterization-based PEFT [3]. For Adapter, we set an adapter size of 64. For LoRA, we choose the bottleneck rank $r = 4, \alpha = 32$ and fine-tune W_q, W_v matrices in the attention layers. We fine-tune GPT-2-Small (117M parameters) with the AdamW optimizer with a weight decay of 1e-2. For the E2E dataset, the learning rate is fixed at 0.0004. For the Echr dataset, we employ an adaptive learning rate starting from 0.0004. We utilize Opacus to add DP perturbation to the gradients during private fine-tuning [36].

3) Baselines: The baselines include: 1) a parameter-efficient fine-tuning (PEFT) approach without privacy, termed Non-private, 2) a differential privacy fine-tuning method where DP-SGD is applied to the new parameters [11] in PEFT context.

4) Metrics: We evaluate model utility using perplexity (PPL) on an unseen test set of sentences from the private fine-tuning model. Lower perplexity indicates high model utility on the dataset. To evaluate the effectiveness of privacy protection, we conduct tests with privacy attacks and measure the performance of privacy preservation. We adopt the definition of Personally Identifiable Information (PII) to identify sensitive information in the fine-tuning datasets [6], [37], which can be a direct or quasi-identifier used for re-identifying individuals. We use the state-of-the-art Named Entity Recognition (NER) flair [38] to tag the PII instances in the datasets. Two privacy attacks proposed in [6] and a perplexity-based Membership inference attack are employed for our evaluation of privacy protection, as they have shown significant potential for PII leakage. When computing the metrics, we exclude the leaked PIIs from the pre-trained models. Our target PII class is "PERSON", representing individuals' names.

1) Extraction Attack (EA): Attackers aim to extract PII from the target language model. We measure the Precision and Recall to reflect the effectiveness of the attack, where Precision measures the proportion of correctly retrieved PIIs and Recall measures the confidence of attackers in the generated PIIs present in the training dataset. In our implementation, the model generates sequences of 256 tokens from an empty prompt using top-k sampling with k = 40.
2) Reconstruction Attack (RA): The adversary's goal is to reconstruct a masked piece of PII in the target sentence. We use Top-1 accuracy to reflect the correctness of predicting a target PII for a sequence. In our implementation, we select 2000 target sequences to attack and report the accuracy. For each target sequence, we sample 64 sequences to generate candidates and determine the final attack result through the computation of perplexity.
3) **Membership inference attack (MIA):** Attackers aim to determine whether a specific target sample was used to train the target model by exploiting differences in the model's behavior on seen versus unseen data. Following previous work [39], we employ perplexity as the prediction score, as member samples typically exhibit lower perplexity compared to non-member samples. In our implementation, we randomly sample 25000 sequences from both training (member) and test (non-member) sets and compute their perplexities using the target model. We evaluate the attack effectiveness using ROC-AUC (Area Under the Receiver Operating Characteristic Curve) score, which measures the model's vulnerability to membership inference - a higher AUC indicates greater susceptibility to the attack.

B. Model Utility and Overhead Comparisons

C. Utility Comparison Across Different PEFT Methods

We first compare the performance of Rap-FT with the baseline models under a consistent privacy level on two datasets on three different PEFT methods, Adapter, BitFit, and LoRA. When BitFit is used for fine-tuning, we initialize the bias of the pretrained model to prevent overfitting. In DP-FT, we set the gradient clipping parameter to 1.0, the noise scale(standard deviation) σ to 0.6, and δ to $1e-5$ (smaller than the inverse of the dataset size [40]). These regulate the noise intensity and ultimately impacts the privacy budgets consumed. For E2E, the

TABLE I
PERFORMANCE COMPARISON ACROSS DIFFERENT PEFT METHODS

Method	**E2E**			**Echr**		
	Adapter	**Bitfit**	**LoRA**	**Adapter**	**Bitfit**	**LoRA**
Non-Private	3.32	3.69	3.49	14.5	16.04	14.60
DP-FT	4.48	4.32	4.54	18.49	17.86	18.15
Rap-FT	4.3	4.36	4.39	17.92	17.51	17.73

TABLE II
MODEL UTILITY UNDER DIFFERENT PRIVACY BUDGETS

Privacy Budget	**E2E**			**Echr**		
ϵ	3.0	5.0	7.0	3.0	5.0	7.0
Non-private		3.49*			14.60*	
DP-FT	4.62	4.53	4.47	18.32	18.1	18.03
Rap-FT	4.45	4.33	4.30	17.79	17.53	17.42

* Non-private values are independent of the privacy budget ϵ.

TABLE III
OVERHEAD COMPARISON ON E2E

Overhead	Train Time per Epoch(min)	Memory per Example(GB/GPU)
Non-private	18.63	1.04
DP-FT	19.06	1.04
Rap-FT	25.44	0.98

privacy budgets consumed across Adapter, BitFit, and LoRA fine-tuning scenarios are 4.31,4.25,4.31 respectively, and for Echr, the privacy budgets consumed are 3.05, 2.88, 3.05. We set the irrelevance ratio $\beta = 0.7$, Top-p initialization ($p = 0.9$) and keep all other hyperparameters the same as the baselines.

TABLE I presents the perplexity and loss across different PEFT methods and datasets, the results demonstrate the applicability of our proposed Rap-FT. Overall, Rap-FT exhibits a noticeable advantage over DP-FT across all three PEFT methods, achieving lower perplexity levels with the same privacy level. From the **perspective of tasks**, both Rap-FT and DP-FT yield higher perplexity on Echr (legal judgment prediction) compared to E2E (natural language generation), indicating the model's greater uncertainty in Echr predictions and making it a more challenging task. In this context, Rap-FT exhibits a more significant improvement in model utility for harder tasks. From the **perspective of PEFT methods**, we find that Rap-FT's impact is weaker on selective PEFT methods. We speculate that this is due to biases being spread out over various parts of the model structure, which are relatively dispersed and small in size. Thus, Rap-FT is suited for PEFT methods with a high concentration of trainable parameters.

1) Utility Comparison Under Different Privacy Budgets: We select LoRA method and conduct experiments based on a predefined privacy budget to evaluate the performance of each framework under varying privacy budget. The results in TABLE II demonstrate that, compared to the DP-FT, the proposed method is effective in enhancing model utility across a range of privacy budgets. Moreover, as the privacy budget increases, the improvement in model utility shows a magnifying trend. By adaptively adding noise, the model's performance is made to align more closely with that of the non-private baseline.

TABLE IV
PERFORMANCE OF RAP-FT WITH VARIOUS RELEVANCE INITIALIZATIONS

Utility				Overhead on E2E	
Perplexity	k/p/b	E2E	Echr	Memory* (GB)	Time per batch(s)
Naive	/	4.38	17.67	2.36	18.16
Top-k	2	4.41	17.68	1.21	22.08
	5	4.39	17.73		
	10	4.38	17.74		
Top-p	0.85	**4.37**	17.68	2.09	35.77
	0.9	4.40	**17.51**		
	0.95	4.38	17.71		
Beam	2	4.38	17.73	1.21	24.40
	5	4.38	17.72		
	10	**4.37**	17.7		

* Memory overhead refers to the cost of layerwise relevance propagation through a single layer of the model structure after initialization, using the first layer as an example. It differs across various values of k/p/b, though these differences are only evident beyond the second decimal place.

2) Overhead Comparison: TABLE III shows time and memory overhead for each method on the E2E. As shown in the table, our proposed Rap-FT method requires more time per epoch compared to other methods due to the need to perform separate clipping and noise addition on different gradient subsets. However, processing data in subsets can reduce peak memory usage, allowing flexible memory allocation and lowering GPU overhead. Overall, our approach trades some time for reduced memory usage, while also improving the model's performance on specific generative tasks.

D. Performance Analysis of Our Framework

1) Performance of Rap-FT With Various Relevance Initializations: Using the LoRA fine-tuning method, we design exploratory experiments to investigate the effectiveness of our proposed relevance initialization method on two datasets, as shown in TABLE IV. In the relevance decomposition process, positive-to-negative ratio of the layer output (α in Section III-B) is set to 0.5. We select three values for the hyperparameters of each relevance initialization method and use the naive method (Section V-B) as a baseline. Our findings indicate that the results of the decoding-based initialization method and the naive method are closely aligned. This suggests that the proposed relevance initialization is capable of representing the global model output matrix for relevance computation, eliminating the need to retrieve the entire matrix. We also validate the memory and time overhead of the relevance analysis process. Although preprocessing the matrix increases the relevance computation time per batch compared to the naive method, it concurrently reduces memory overhead. Among the three decoding-based methods, Top-p and Beam initialization additionally demonstrate a performance advantage in certain scenarios compared to the naive method.

2) Performance of Rap-FT With Various Irrelevance Ratios: We evaluate Rap-FT's performance against the irrelevance ratio β using the LoRA fine-tuning method. Fig. 5 shows that, for both tasks, perplexity (PPL) initially decreases and then increases as β rises, with more complex variations on the Echr dataset. This is because as the irrelevance ratio

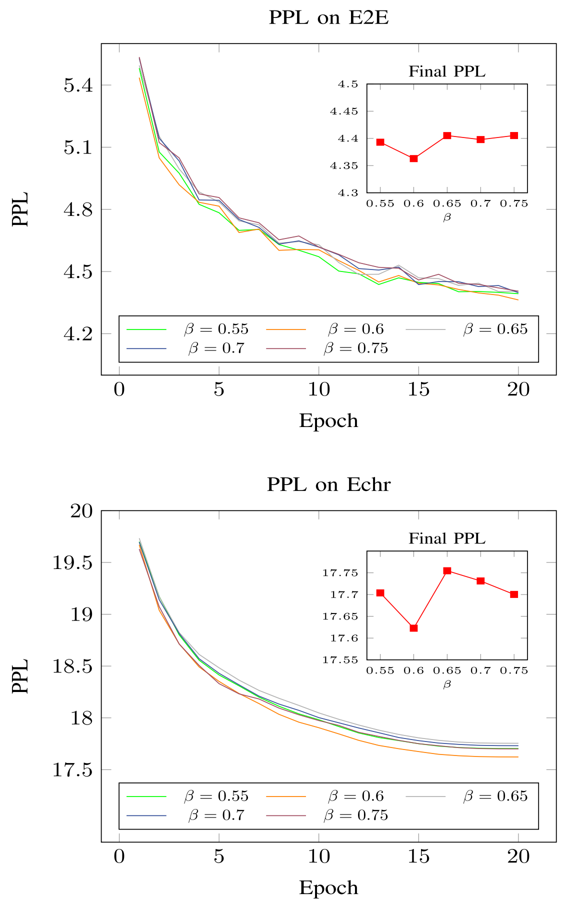


Fig. 5. Perplexity of Rap-FT with various irrelevance ratios.

TABLE V
RESULTS OF PRIVACY ATTACKS ACROSS VARIOUS PEFT METHODS

EA	Non-private		DP-FT		Rap-FT	
	Precision	Recall	Precision	Recall	Precision	Recall
Adapter	46.87%	1.42%	35.27%	0.91%	33.03%	0.62%
BitFit	55.35%	1.24%	49.45%	0.83%	44.91%	0.89%
LoRA	44.21%	1.37%	31.52%	0.81%	33.26%	0.45%
RA	Accuracy		Accuracy		Accuracy	
Adapter	0.35%		0.15%		0.1%	
BitFit	0.45%		0.2%		0.25%	
LoRA	0.25%		0.2%		0.2%	
MIA	ROC-AUC		ROC-AUC		ROC-AUC	
Adapter	0.5042		0.5026		0.5025	
BitFit	0.5034		0.5032		0.5030	
LoRA	0.507		0.5020		0.5021	

rises, the size of the weakly significant gradient $|g_w|$ grows, meaning that the proportion of less significant perturbations increases. Meanwhile, the intensity of Gaussian noise added to the weakly significant gradient σ_w also rises with β, leading to an overall increase in perturbation. Accordingly, the optimal value for model utility improvement is approximately 0.6 in our cases, however, adaptive adjustments can be made based on the required privacy and user preferences for other tasks.

E. Defense Against Privacy Attacks

We evaluate the effectiveness of privacy protection against privacy attacks using the Echr dataset since it contains private

TABLE VI
RESULTS OF MEMBERSHIP INFERENCE ATTACK WITH DIFFERENT PRIACY BUDGETS

Privacy Budget	ROC-AUC		
	Non-private	DP-FT	Rap-FT
$\epsilon = 3.0$	0.507	0.5022	0.5028
$\epsilon = 5.0$		0.5027	0.5026
$\epsilon = 7.0$		0.5023	0.5025

information of individuals on court. Specifically, we compare the defense performance of Rap-FT with the baseline methods against various privacy attacks. As summarized in TABLE V, the baseline without DP protection shows the poorest results against all three types of privacy attacks. The difference in success rates between Rap-FT and DP-FT is minimal, suggesting that Rap-FT does not significantly compromise privacy preservation. Specifically, for the extraction attack, we run five trials and average the results, which indicates Rap-FT can provide comparable defense performance to DP-FT. When attacking models fine-tuned with BitFit, Rap-FT shows a relatively high Recall but a lower Precision rate compared to DP-FT. When attacking models fine-tuned with LoRA, our Rap-FT exhibits relatively large fluctuations in Precision, resulting in a higher overall score, while Recall is significantly superior to DP-FT. For models fine-tuned with Adapter, Rap-FT slightly outperforms DP-FT on both metrics. Regarding the Reconstruction attack, the low attack success rate indicates that DP can effectively mitigate privacy leakage [6]. Rap-FT can maintain a low success rate comparable to both baseline methods in all scenarios. For the Membership inference attack, the ROC-AUC of Rap-FT is comparable to DP-FT in three cases. This indicates that the Rap-FT's ability to resist membership inference attacks is very close to baseline methods. These results demonstrate that our Rap-FT can achieve a comparable level of privacy protection in most scenarios compared to the baselines, with only minor occasional degradation in privacy protection performance compared to the baseline. However, considering its improvement in utility, Rap-FT clearly strikes a good balance between privacy and utility.

Furthermore, we evaluate the performance of Rap-FT and the baselines with different privacy budgets under the MIA. The results in Table VI show that all models fine-tuned with DP perform better in resisting membership inference attacks compared to the non-private setting. Additionally, Rap-FT exhibits minimal performance differences from DP-FT, underscoring its robustness over varying privacy budgets.

VIII. CONCLUSION

In this paper, we present the first solution for achieving a balance between privacy protection and model utility when fine-tuning language models with different PEFT methods. Our proposed approach is adaptable to the three major types of PEFT methods. We introduce different relevance propagation schemes associated with trainable parameters, marking the first attempt to explain PEFT. Additionally, we develop generation schemes for the relevance map of trainable parameters in additive, selective, and reparameterization-based PEFT methods. Moreover, we propose an adaptive gradient perturbation strategy during the fine-tuning process, which splits the gradients dimensionally into two subsets, allowing appropriate noise levels to be added while mitigating the adverse effects of perturbations. Through extensive experiments, we evaluate the effectiveness of our proposed Rap-FT in balancing privacy protection and model utility. The results demonstrate that our approach enhances model utility while maintaining privacy across all three types of PEFT methods.

Acknowledgment

The authors are extremely grateful to Wenti Yang and Xuan Li for their valuable inputs throughout the project. They also thank their anonymous reviewers for their insights and feedback.

References

[1] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1872–1897, Sep. 2020.

[2] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.

[3] V. Lialin, V. Deshpande, X. Yao, and A. Rumshisky, "Scaling down to scale up: A guide to parameter-efficient fine-tuning," 2023, *arXiv:2303.15647*.

[4] F. Mireshghallah, A. Uniyal, T. Wang, D. Evans, and T. Berg-Kirkpatrick, "Memorization in NLP fine-tuning methods," 2022, *arXiv:2205.12506*.

[5] N. Kandpal, K. Pillutla, A. Oprea, P. Kairouz, C. A. Choquette-Choo, and Z. Xu, "User inference attacks on large language models," 2023, *arXiv:2310.09266*.

[6] N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz, and S. Zanella-Béguelin, "Analyzing leakage of personally identifiable information in language models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 346–363.

[7] F. Tian, Z. Wu, X. Gui, J. Ni, and X. S. Shen, "Fine-grained query authorization with integrity verification over encrypted spatial data in cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1831–1847, Jul. 2022.

[8] S. Kim, S. Yun, H. Lee, M. Gubri, S. Yoon, and S. J. Oh, "ProPILE: Probing privacy leakage in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–13.

[9] X. Li, F. Tramer, P. Liang, and T. Hashimoto, "Large language models can be strong differentially private learners," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–31. [Online]. Available: https://openreview.net/forum?id=bVuP3ltATMz

[10] M. Du, X. Yue, S. S. M. Chow, T. Wang, C. Huang, and H. Sun, "DP-forward: Fine-tuning and inference on language models with differential privacy in forward pass," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 2665–2679.

[11] D. Yu et al., "Differentially private fine-tuning of language models," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=Q42f0dfjECO

[12] L. Xiang, J. Yang, and B. Li, "Differentially-private deep learning from an optimization perspective," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 559–567.

[13] Z. Xu, S. Shi, A. X. Liu, J. Zhao, and L. Chen, "An adaptive and fast convergent approach to differentially private deep learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2020, pp. 1867–1876.

[14] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0130140.

[15] M. Gong, K. Pan, Y. Xie, A. K. Qin, and Z. Tang, "Preserving differential privacy in deep neural networks with relevance-based adaptive noise imposition," *Neural Netw.*, vol. 125, pp. 131–141, May 2020.

[16] L. Chen, D. Yue, X. Ding, Z. Wang, K. R. Choo, and H. Jin, "Differentially private deep learning with dynamic privacy budget allocation and adaptive optimization," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 4422–4435, 2023.

[17] Z. Zhang, J. Wen, and M. Huang, "ETHICIST: Targeted training data extraction through loss smoothed soft prompting and calibrated confidence estimation," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, Toronto, ONT, Canada, 2023, pp. 12674–12687.

[18] W. Shi, A. Cui, E. Li, R. Jia, and Z. Yu, "Selective differential privacy for language modeling," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2022, pp. 2848–2859.

[19] L. Demelius, R. Kern, and A. Trügler, "Recent advances of differential privacy in centralized deep learning: A systematic survey," 2023, *arXiv:2309.16398*.

[20] L. T. Phong and T. T. Phuong, "Differentially private stochastic gradient descent via compression and memorization," *J. Syst. Archit.*, vol. 135, Feb. 2023, Art. no. 102819.

[21] Y. Hu, D. Li, Z. Tan, X. Li, and J. Wang, "Adaptive clipping bound of deep learning with differential privacy," in *Proc. IEEE 20th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Oct. 2021, pp. 428–435.

[22] C. Dwork, "Differential privacy," in *Proc. Int. Colloq. Automata, Lang., Program. (ICALP)*. Berlin, Germany: Springer, Jul. 2006, pp. 1–12.

[23] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[24] X. Liu et al., "GPT understands, too," *AI Open*, vol. 5, pp. 208–215, Jan. 2024.

[25] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," 2021, *arXiv:2104.08691*.

[26] H. Liu et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 1950–1965.

[27] E. Ben Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," 2021, *arXiv:2106.10199*.

[28] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.

[29] X. Wang, N. Wang, L. Wu, Z. Guan, X. Du, and M. Guizani, "GBMIA: Gradient-based membership inference attack in federated learning," in *Proc. IEEE Int. Conf. Commun.*, May 2023, pp. 5066–5071.

[30] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[31] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 782–791.

[32] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[33] B. Balle and Y.-X. Wang, "Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 394–403.

[34] J. Novikova, O. Dušek, and V. Rieser, "The E2E dataset: New challenges for end-to-end generation," in *Proc. 18th Annu. SIGdial Meeting Discourse Dialogue*, Saarbrücken, Germany, 2017, pp. 201–206. [Online]. Available: https://aclanthology.org/W17-5525

[35] I. Chalkidis, I. Androutsopoulos, and N. Aletras, "Neural legal judgment prediction in English," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 4317–4323. [Online]. Available: https://aclanthology.org/P19-1424

[36] A. Yousefpour et al., "Opacus: User-friendly differential privacy library in PyTorch," 2021, *arXiv:2109.12298*.

[37] I. Pilán, P. Lison, L. Øvrelid, A. Papadopoulou, D. Sánchez, and M. Batet, "The text anonymization benchmark (TAB): A dedicated corpus and evaluation framework for text anonymization," *Comput. Linguistics*, vol. 48, no. 4, pp. 1053–1101, Dec. 2022.

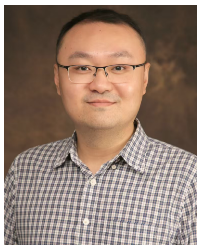
[38] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use framework for state-of-the-art NLP," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 54–59.

[39] N. Carlini et al., "Extracting training data from large language models," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2633–2650.

[40] S. Gopi, Y. T. Lee, and L. Wutschitz, "Numerical composition of differential privacy," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 11631–11642.

Naiyu Wang received the B.Eng. degree in information security from North China Electric Power University, Beijing, China, in 2020, where she is currently pursuing the Ph.D. degree with the School of Control and Computer Engineering. Her research interests include AI security and privacy-preserving and federated learning.

Shen Wang received the B.Eng. degree in internet of things engineering from North China Electric Power University, Beijing, China, in 2022, where he is currently pursuing the master's degree with the School of Control and Computer Engineering. His research focuses on AI Security.



Meng Li (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology (BIT), China, in 2019. He is an Associate Professor and a Personnel Secretary at the School of Computer Science and Information Engineering, Hefei University of Technology (HFUT), China. He is also a Post-Doctoral Researcher at the Department of Mathematics and HIT Center, University of Padua, Italy, where he is with the Security and PRIvacy Through Zeal (SPRITZ) Research Group led by Prof. Mauro Conti (IEEE Fellow). His research interests include security, privacy, applied cryptography, blockchain, TEE, and Internet of Vehicles.

Longfei Wu (Member, IEEE) received the B.E. degree from Beijing University of Posts and Telecommunications in July 2012, and the Ph.D. degree in computer and information sciences from Temple University in July 2017. He is currently an Assistant Professor with the Department of Mathematics and Computer Science, Fayetteville State University. His research interests are the security and privacy of networked systems and modern computing devices, including mobile devices, IoT, implantable medical devices, and wireless networks.

Zijian Zhang (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Beijing Institute of Technology. He was a Visiting Scholar with the Computer Science and Engineering Department, State University of New York at Buffalo, in 2015. He is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.

Zhitao Guan (Member, IEEE) received the B.Eng. and Ph.D. degrees in computer application from Beijing Institute of Technology, China, in 2002 and 2008, respectively. He is currently a Professor with the School of Control and Computer Engineering, North China Electric Power University. His current research focuses on blockchain, AI security, privacy computing. He has authored more than 100 peer-reviewed journal and conference papers in these areas.

Liehuang Zhu (Senior Member, IEEE) received the M.S. degree in computer science from Wuhan University, Wuhan, China, in 2001, and the Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004. He is a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include data security and privacy protection, blockchain applications, and AI security. He has authored more than 500 journal and conference papers in these areas.