MineGrad: Gradient Inversion Attacks on LoRA Fine-Tuning

Hasin Us Sami*, **Swapneel Sen***, **Başak Güler**
University of California, Riverside, CA
hsami003@ucr.edu, ssen010@ucr.edu, bguler@ece.ucr.edu

Abstract

Parameter-efficient fine-tuning (PEFT), such as low-rank adaptation (LoRA), has recently been adopted in federated learning to reduce communication and computation costs. In this setup, users download a pretrained model from the server prior to fine-tuning, and then fine-tune lightweight LoRA modules locally while keeping the pretrained model frozen, sharing only the gradients of the fine-tuning parameters with the server. Despite its growing popularity, robustness of federated fine-tuning against an adversarial server remains underexplored, where the server maliciously tampers with the training protocol to breach the privacy of users' data. In this work, we investigate gradient inversion attacks on LoRA fine-tuning. We propose an analytical attack that enables a malicious server to recover private user data by leveraging a poisoned pretrained model and fine-tuning parameters. Our design embeds fine-tuning data within the shared gradients, to allow the server to analytically reconstruct user data. Unlike prior works, our attack is applicable to both language and vision tasks, does not rely on computationally expensive (adversarial) pretraining with public datasets or require the number of training tokens to be less than the rank of LoRA modules. Experimental results on both language and vision tasks demonstrate high-fidelity data recovery across multiple baselines, revealing several critical vulnerabilities.

*Equal contribution.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Large-scale pretrained models achieved remarkable success across various tasks. However, their immense size—often billions of parameters Radford et al. (2019); Devlin et al. (2019); Liu et al. (2019); Brown et al. (2020); He et al. (2021); Touvron et al. (2023)—renders full fine-tuning (FFT) computationally and storage intensive. Parameter-efficient fine-tuning (PEFT) mitigates this by updating only a small subset of parameters, either integrated into the pretrained model Cai et al. (2020); Zhao et al. (2024a); Kowsher et al. (2024) or added as external modules Houlsby et al. (2019); Mahabadi et al. (2021); Hu et al. (2022); Hayou et al. (2024).

To reduce communication, computation and memory overhead, PEFT is increasingly adopted in federated learning (FL) Zhang et al. (2023b); Kim et al. (2023); Shysheya et al. (2023); Cho et al. (2024), where users collaboratively train a shared model without revealing local data. While FL's core principle is on-device privacy, shared gradients remain vulnerable. Gradient inversion attacks can reconstruct training data Zhu et al. (2019); Geiping et al. (2020); Yin et al. (2021); Huang et al. (2021); Lu et al. (2022); Deng et al. (2021); Gupta et al. (2022); Fowl et al. (2023), while membership inference Shokri et al. (2016); Nasr et al. (2019); Song and Mittal (2021); Mireshghallah et al. (2022); Nguyen et al. (2023b) and data extraction Carlini et al. (2018, 2021); Tramèr et al. (2022); Panda et al. (2024) attacks expose additional risks. Among these, gradient inversion is especially dangerous since it requires no prior knowledge of target samples.

Despite these concerns, gradient inversion attacks on PEFT remain largely underexplored. Feng and Tramèr (2024) recently introduced a gradient inversion attack by deploying a poisoned pretrained model, focusing on the FFT setup, where the attacker has access to gradients for the full model parameters. Liu et al. (2024) and Wen et al. (2024) explore manipulating the pretrained model for membership inference and data extraction attacks in FFT and PEFT settings. Recent

observations suggest that PEFT methods may offer better privacy protections against inversion attacks by reducing the number of leaked parameters Zhang et al. (2023b). However, this setup does not consider stronger adversaries, such as an adversarial server that can manipulate the pretrained model along with shared PEFT parameters before sending it to users Fowl et al. (2022, 2023); Zhao et al. (2024b); Feng and Tramèr (2024). In such cases, a target user (victim) might inadvertently download a tampered pretrained model and fine-tuning parameters from a compromised server.

This work investigates gradient inversion attacks on low-rank adaptation (LoRA) Hu et al. (2022), where users fine-tune small low-rank matrices while keeping the pretrained model frozen Zhang et al. (2023b); Sun et al. (2024); Bian et al. (2024); Cho et al. (2024). This reduced gradient space complicates inversion. While recent work has explored privacy-aware LoRA Yu et al. (2022); Sun et al. (2024); Li et al. (2024), practical gradient inversion attacks remain underexplored, for instance, whether adversarial servers can invert LoRA gradients for vision transformers (ViT) Dosovitskiy et al. (2021). Our work answers these questions in the affirmative.

Recently, Yao (2024) considered attacks on LoRA-based fine-tuning of a pretrained diffusion model to generate private fine-tuning data. The attack leverages prior training of a neural network encoder using a publicly available dataset from a similar domain to the private fine-tuning data. In contrast, here we do not assume availability of publicly available datasets from the fine-tuning domain - as such data may not be available in practice, especially in privacy-sensitive settings such as FL. Gao et al. (2025) proposes a discrete optimization attack, which requires access to the gradients of the word embedding layer. In LoRA fine-tuning, however, word embedding layer is part of the frozen pretrained model and the server loses access to this gradient, hence the attack cannot be applied to our problem.

Analytical attacks are proposed to recover fine-tuning data from gradients in the context of FFT Feng and Tramèr (2024) or PEFT with adapters Sami et al. (2025) of a pretrained model. These works utilize a carefully crafted pretrained model to facilitate undistorted propagation of target token or patch embeddings through the network up to the trainable linear layer, followed by a non-linear activation function like GELU or ReLU. Subsequently, the weight and bias parameters are precisely designed to ensure that no two tokens or patches activate the same set of neurons. Thus, the corresponding weight and bias gradients can be utilized to accurately recover the original data. In contrast, in LoRA fine-tuning, trainable low-rank matrices are typically not followed by such an activation function, making the aforementioned attacks infeasible.

Another notable work is Petrov et al. (2024), which proposed a gradient inversion attack, DAGER, to recover text data, by leveraging the low-rank structure of weight gradients. This attack can also be applied to LoRA fine-tuning. However, the attack requires the total number of training tokens in the batch to be less than or equal to the rank of LoRA matrices, which is often unrealistic in practice Hu et al. (2022). In addition, DAGER relies on an exhaustive search across all the tokens in the vocabulary in all possible positions, hence it cannot be applied to datasets that do not have a predefined vocabulary such as vision datasets.

To address these challenges, we introduce a novel gradient inversion attack, MineGrad, by leveraging the gradients corresponding to the trainable low-rank matrices. We design a poisoned pretrained model and low-rank matrices to embed the fine-tuning data within the victim's gradients. Using these manipulated gradients, the server can analytically extract the fine-tuning data. Unlike prior methods, our attack can be applied to both language and vision tasks, does not rely on computationally expensive pretraining with a public dataset, and does not require that the number of tokens be less than the rank of the LoRA matrices Petrov et al. (2024). Our code is available at `https://github.com/info-ucr/MineGrad`.

2 RELATED WORKS

Parameter-Efficient Fine Tuning and LoRA. PEFT mechanisms reduce the storage and computation costs of fine-tuning large pretrained models by freezing the model and updating lightweight modules Houlsby et al. (2019); Cai et al. (2020); Mahabadi et al. (2021); Li and Liang (2021); Hu et al. (2022). Among the PEFT techniques, LoRA and its variants are among the most widely-adopted due to their superior performance in resource-limited settings Hu et al. (2022); Zhang et al. (2023a); Wen and Chaudhuri (2024); Hayou et al. (2024). Recent works have further extended the application of LoRA to FL Zhang et al. (2023b); Cho et al. (2024); JianHao et al. (2024).

Gradient inversion attacks. Gradient inversion attacks aim to reconstruct training data from shared gradients. To do so, earlier works retrieve data by solving a distance minimization problem Zhu et al. (2019); Geiping et al. (2020) or training a generative adversarial network Hitaj et al. (2017); Wang et al. (2019); Zhang et al. (2020). More recent works leverage batch normalization statistics Yin et al. (2021); Hatamizadeh et al. (2022), dropout mask optimization Scheliga et al. (2023) or blind source separation Kariyappa et al. (2023) to enhance the attack performance. Beyond attacks to

recover image samples, several works propose mechanisms to recover text or tabular data Deng et al. (2021); Gupta et al. (2022); Balunovic et al. (2022); Vero et al. (2023). Attack performance can further be enhanced by considering stronger adversaries who can tamper with model parameters or architectures Fowl et al. (2022); Chu et al. (2023); Fowl et al. (2023); Feng and Tramèr (2024). A gradient-inversion attack was recently demonstrated for PEFT with adapters Sami et al. (2025). To recover data, this work leverages weight and bias gradients from adapter modules, along with distinct neuron activations within a linear layer. Other lines of work consider membership inference and data extraction attacks Shokri et al. (2016); Nasr et al. (2019); Song and Mittal (2021); Mireshghallah et al. (2022); Nguyen et al. (2023b); Carlini et al. (2021); Liu et al. (2024).

3 PRELIMINARIES AND PROBLEM SETUP

LoRA fine-tuning. In a typical transformer Vaswani et al. (2017); Devlin et al. (2019); Liu et al. (2019), a distinct position encoding vector is added to each token. These token embeddings are then passed through transformer encoders, where each encoder consists of stacked multi-head self-attention (MSA), multi-layer perceptron (MLP) and LayerNorm (LN) layers. For

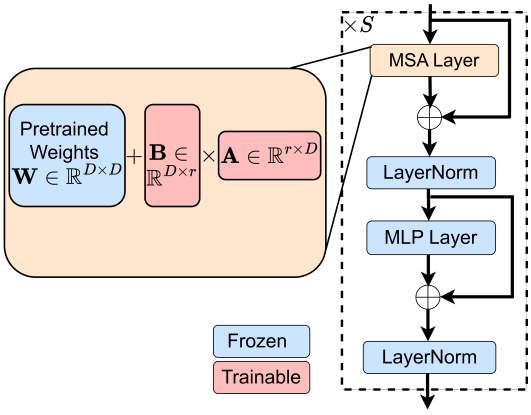


Figure 1: **Transformer encoder with LoRA modules.** Transformer encoder consists of stacked MSA, LayerNorm, MLP layers and residual connections. Low-rank matrices $\mathbf{A}, \mathbf{B}$ (of rank r) are inserted within each MSA layer.

LoRA fine-tuning, low-rank decomposition matrices $\mathbf{A}$, $\mathbf{B}$ (of rank r) are used as additive modules within each MSA layer as illustrated in Fig. 1. The product $\mathbf{BA}$ is added to the pretrained MSA weights during inference. During backpropagation, only the gradients with respect to $\mathbf{A}$ and $\mathbf{B}$ are computed, while keeping the pretrained weights frozen.

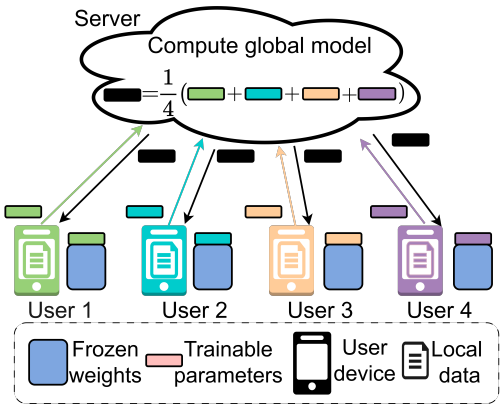


Figure 2: **Federated LoRA with $U = 4$ users.** In each training round, users perform fine-tuning locally and send the LoRA gradients to the server. The server performs aggregation and sends the new global state of the LoRA modules to the users.

Federated LoRA fine-tuning. We consider a centralized FL setting with U users. For LoRA-based fine-tuning, we denote the low-rank matrices of user $u \in [U]$ as $\mathbf{W}_F^u \triangleq \{\mathbf{A}^u, \mathbf{B}^u\}$. Prior to training, the server shares a pretrained model $\mathbf{W}_P$ with the users. In each training round, user $u \in [U]$ performs fine-tuning on its local dataset and sends a local update (gradient) $\Delta\mathbf{W}_F^u$ to the server JianHao et al. (2024). After receiving the local updates, the server computes the current state of the global model $\mathbf{W}_F \triangleq \{\mathbf{A}, \mathbf{B}\}$:

$$\mathbf{W}_F = \mathbf{W}_F + \frac{1}{U}\sum_{u=1}^{U} \Delta\mathbf{W}_F^u \tag{1}$$

The goal is to minimize the global loss function, $\mathcal{L} \triangleq \frac{1}{U}\sum_{u=1}^{U} \mathcal{L}_u(\mathbf{W}_P, \mathbf{W}_F)$, where $\mathcal{L}_u$ denotes the local loss of user u and $\mathbf{W}_P$ denotes the frozen pretrained model.

Threat model. We consider an adversarial server who can tamper with the model parameters shared with users. This includes: 1) disseminating a poisonous pretrained model to users once prior to fine-tuning, and 2) modifying the global fine-tuning parameters sent to the users in each training round after aggregation. As in Nguyen et al. (2023a); Shysheya et al. (2023); Chen et al. (2023), the server performs pretraining with proprietary data or publicly available proxy datasets and then releases the pretrained model to the edge users. Hence, the server can manipulate the pretrained model (once before training) and global fine-tuning parameters (at each round during training). The adversary aims to coerce a target user into producing gradients with respect to the poisonous fine-tuning modules, to reveal its local data. Malicious tampering with the training protocol is inspired by Fowl et al. (2022, 2023); Chu et al. (2023); Zhao et al. (2024b); Feng and Tramèr

(2024). Users often download a pretrained model without any formal verification protocol Feng and Tramèr (2024); Wen et al. (2024); Liu et al. (2024). Training is usually performed in the background e.g., during night time when the device is being charged or remains idle Ramaswamy et al. (2019); Fowl et al. (2023). Therefore, a user may not be observing the training loss on a continual basis. Hence, even though a malicious pretrained model may adversely affect the training performance, a user may fail to detect such suspicious activity under the common practices of FL.

Existing Attacks. The closest to our work is DAGER Petrov et al. (2024), which leverages rank-deficiency of weight gradients to recover the input tokens. Let us denote the total number of input tokens as N, where each token embedding is of dimension D. For a linear layer with stacked input tokens $\mathbf{Y} \in \mathbb{R}^{N\times D}$, weight $\mathbf{W} \in \mathbb{R}^{D\times r}$, and bias $\mathbf{b} \in \mathbb{R}^r$, the output can be written as $\mathbf{Z} \triangleq \mathbf{YW} + (\mathbf{b}|\dots|\mathbf{b})^{\mathrm{T}}$. Accordingly, the gradient of user u can be written as,

$$\frac{\partial \mathcal{L}_u}{\partial \mathbf{W}} \triangleq \mathbf{Y}^{\mathrm{T}} \frac{\partial \mathcal{L}_u}{\partial \mathbf{Z}} \tag{2}$$

The rank of $\frac{\partial \mathcal{L}_u}{\partial \mathbf{W}}$ is at most N if $N \leq min\{D, r\}$. Due to the rank-deficiency of the gradient matrix $\frac{\partial \mathcal{L}_u}{\partial \mathbf{W}}$, its columns form a subspace of $\mathbb{R}^D$ with dimension N. Next, DAGER initiates an exhaustive search algorithm with all possible tokens in the vocabulary in all possible positions. The goal is to find the set of target input embeddings by checking whether they are in the column space of $\frac{\partial \mathcal{L}_u}{\partial \mathbf{W}}$. Overall, the success of the reconstruction algorithm relies on the assumption that the total number of training tokens N satisfies: $N \leq min\{D, r\}$.

Challenges. With LoRA fine-tuning, the attacker has a much reduced space of observable gradients. The attacker only has access to r gradient vectors per low-rank matrix in each MSA layer. As noted in Hu et al. (2022), the value of r typically ranges from 1 to 64. When applying DAGER Petrov et al. (2024) to LoRA fine-tuning, the attack success requires the total number of training tokens to be less than or equal to the rank, r, of the LoRA matrices. This condition is often not satisfied in practice since the rank of the LoRA matrices is typically small as noted above. In addition, DAGER relies on an exhaustive search with all possible words in the vocabulary in all possible positions, which is not feasible for datasets without a predefined vocabulary such as vision datasets.

Contributions. To address these challenges, we propose a novel inversion attack, MineGrad, for LoRA fine-tuning in FL. Our approach involves a carefully crafted design of the pretrained model, which remains fixed during fine-tuning but is utilized during inference. Additionally, we demonstrate an adversarial manipulation of the lightweight LoRA matrices, which are shared with the users in each training round during federated fine-tuning. These two adversarial components work in tandem to reveal fine-tuning data of a target user effectively. To maximize the number of tokens reconstructed, MineGrad mitigates the low-rank issue of a single matrix by utilizing the gradients from multiple LoRA matrices across different encoders. Unlike Petrov et al. (2024), MineGrad can successfully recover data even when the number of training tokens are greater than the rank of LoRA matrices and can be applied to vision datasets.

4 MINEGRAD

For simplicity, we initially consider text classification as the downstream task, and provide the key design principles of MineGrad. Details are deferred to Apps. B and C. For the pretrained model, we first consider encoder-based transformer architectures building on bidirectional self-attention, such as BERT Devlin et al. (2019) and its successor RoBERTa Liu et al. (2019), which has been used to demonstrate the efficacy of LoRA over other PEFT methods Hu et al. (2022). Our framework can also be applied to decoder-based architectures with unidirectional self-attention such as GPT-2 Radford et al. (2019), as we demonstrate in App. E.4. In our experiments, we also extend our results to vision transformers (ViT) for image classification.

Fig. 1 illustrates the transformer encoder with LoRA modules. The typical layers within an encoder, i.e., MSA, LN, and MLP are frozen, while only the LoRA matrices and classification head are trainable. For simplicity, here we consider fine-tuning on a single text sequence, consisting of N tokens. Our framework can also handle a batch of sequences, as shown in Section 5 along with the theoretical analysis in App. D.

Denote the word embedding of token $n \in [N]$ as $\mathbf{x}^{(n)} \in \mathbb{R}^D$, and the class token as $\mathbf{x}^{(0)}$. Embedding $\mathbf{x}^{(n)}$ is added to a position encoding vector $\mathbf{e}^{(n)} \in \mathbb{R}^D$, resulting in a token embedding,

$$\mathbf{y}^{(n)} \triangleq \mathbf{x}^{(n)} + \mathbf{e}^{(n)} \text{ for } n \in \{0, \dots, N\} \tag{3}$$

We now demonstrate a malicious parameterization of the pretrained model and global LoRA modules for the server to recover the private fine-tuning tokens for a set of target positions $\mathcal{T}$, using the gradients shared by a victim user u.

Key intuition. MineGrad is an adversarial design where each column of the low-rank adaptation matrix $\mathbf{B}$ reveals a token at a (different) targeted position via its gradient. Since $\mathbf{B}$ has rank r, at most r tokens can be revealed per encoder. To scale, we employ multiple

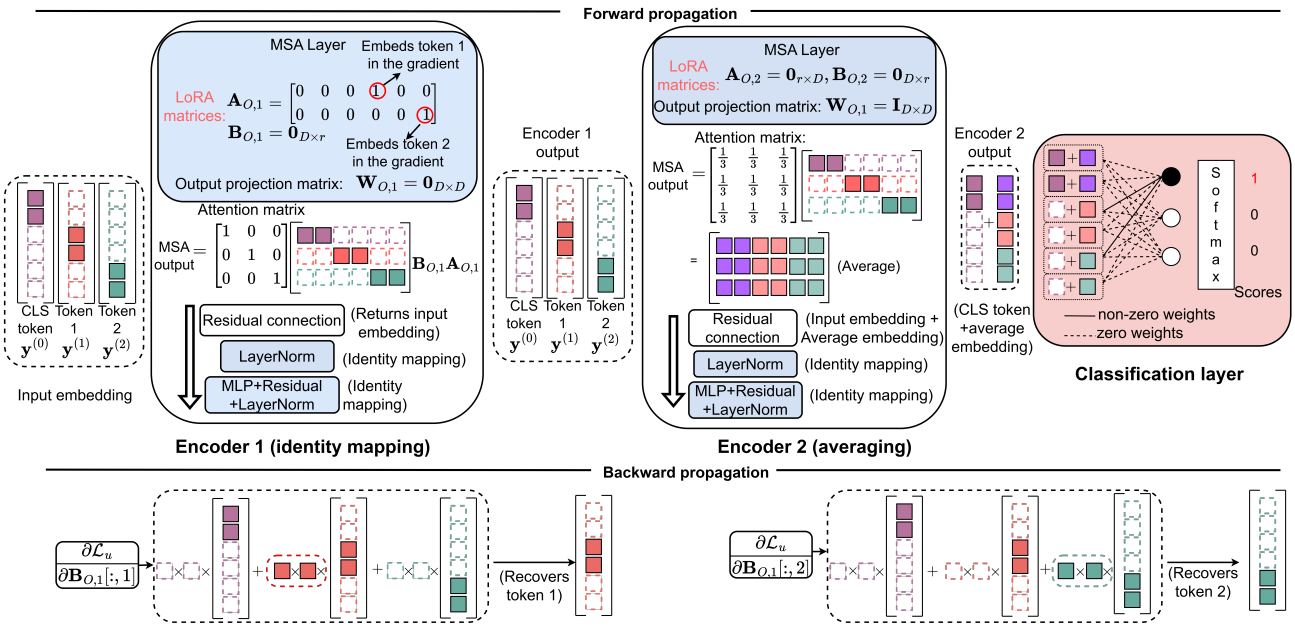


Figure 3: Overview of MineGrad with $D = 6$, $r = 2$, $N = 2$. Each token embedding $\mathbf{y}^{(n)}$ for $n = 0, 1, 2$ contains large values in two distinct positions (colored). The 1st encoder is designed to output identity mapping of the input tokens. The LoRA matrices $\mathbf{A}_{O,1}$, $\mathbf{B}_{O,1}$ in the 1st encoder are designed to recover tokens 1 and 2. The 2nd encoder is designed to produce average of the tokens. In the classification layer, only the embedding for the class token is utilized. Only the 3rd and 5th elements of this embedding are connected to the 1st logit via large weights, leading the 1st logit to produce a softmax score of 1. In the gradient for the 1st column of $\mathbf{B}_{O,1}$, each embedding is weighted by $\mathbf{y}_3^{(i)}\mathbf{y}_4^{(i)}$ for token i, where $\mathbf{y}_d^{(i)}$ is the d^{th} element of $\mathbf{y}^{(i)}$. Token 1 has a larger weight compared to the other tokens, which enables the recovery of token 1. In the gradient for the 2nd column of $\mathbf{B}_{O,1}$, each embedding is weighted by $\mathbf{y}_5^{(i)}\mathbf{y}_6^{(i)}$ for token i, leading to the recovery of token 2.

encoders and carefully structure self-attention and low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ to preserve token embeddings via identity mappings in the forward pass. During back-propagation, each encoder then exposes private tokens from distinct target positions. Our design ensures that for each column n of the fine-tuning matrix $\mathbf{B}$, the local gradient shared by the user contains a weighted average $\sum_{i\in\mathcal{T}} \alpha_{n,i}\mathbf{y}^{(i)}$ of the token embeddings $\mathbf{y}^{(i)}$, where the weight $\alpha_{n,i}$ is much larger for the target token compared to all other tokens. This enables recovery of a distinct token embedding from each gradient. The weighting is controlled by the position encoding vector $\mathbf{e}^{(n)}$. We illustrate this idea in Fig. 3.

Position Encoding. Since position encoding vectors are part of the pretrained model, the attacker has the capability to tamper with these parameters once Feng and Tramèr (2024). Let H be the total number of heads in the MSA layer and $\overline{D}$ be the dimension of each head, i.e., $\overline{D} \triangleq D/H$. The server designs the position encoding vectors as,

$$\mathbf{e}_d^{(n)} \triangleq \begin{cases} c_1 & \text{if } d = 2n+1 \\ -c_1 & \text{if } d = 2n+2 \\ c_2 & \text{if } d = 2n+1+(h-1)\overline{D} \text{ for } 2 \le h \le H \\ -c_2 & \text{if } d = 2n+2+(h-1)\overline{D} \text{ for } 2 \le h \le H \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{e}_d^{(n)}$ is the d^{th} element of $\mathbf{e}^{(n)}$, and c_1 is a large positive value ($\sim 10^2$) such that $c_1 >> c_2$. We want c_2 to have negligible effect on the standard deviation of $\mathbf{e}^{(n)}$ compared to c_1, while ensuring,

$$(\mathbf{e}^{(i)})_h^{\mathrm{T}}(\mathbf{e}^{(i)})_h >> (\mathbf{e}^{(i)})_h^{\mathrm{T}}(\mathbf{e}^{(j)})_h \quad \text{if } i \neq j \tag{5}$$

for all heads $h \in [H]$, where $(\mathbf{e}^{(i)})_h \triangleq [\mathbf{e}^{(i)}_{h\overline{D}+1} \cdots \mathbf{e}^{(i)}_{(h+1)\overline{D}}]$ represents the $\overline{D}$ elements that propagate through head h. As the word embeddings $\mathbf{x}^{(n)}$ typically lie within the range $[-1, 1]$ Wolf et al. (2020), $\mathbf{e}^{(n)}$ is the dominant factor in determining the mean and standard deviation across the elements in $\mathbf{y}^{(n)}$ from (3). From (4), the mean across the elements in $\mathbf{y}^{(n)}$ is,

$$\mu^{(n)} \triangleq \frac{1}{D}\sum_{d=1}^{D}\mathbf{y}_d^{(n)} \approx \frac{1}{D}\sum_{i=1}^{D}\mathbf{e}_d^{(n)} = 0 \triangleq \mu \tag{6}$$

whereas the standard deviation is,

$$\sigma^{(n)} \approx \sqrt{\frac{1}{D}\sum_{d=1}^{D}(\mathbf{e}_d^{(n)} - \mu^{(n)})^2} \triangleq \sigma \tag{7}$$

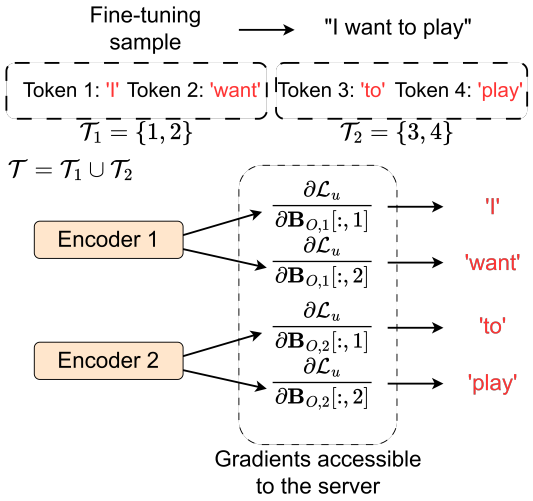


Figure 4: **Token extraction from multiple encoders.** The rank of the LoRA modules is $r = 2$. Gradients from Encoder 1's LoRA modules recover tokens 1 and 2 in the target set $\mathcal{T}_1 = \{1, 2\}$. Gradients from Encoder 2's LoRA modules recover tokens 3 and 4, i.e., $\mathcal{T}_2 = \{3, 4\}$.

As we describe later, this design will be critical in ensuring that each target token can be recovered from the accumulated gradients for the LoRA modules.

Target Encoders (Encoders 1 to $S-1$). Let S be the total number of encoders. We call encoders 1 to $S-1$ the *target encoders*, as the fine-tuning modules of these encoders will produce the gradients that reveal the target tokens. The set of tokens targeted by encoder $t \in [S-1]$ is denoted by $\mathcal{T}_t$. We denote the n^{th} element in set $\mathcal{T}_t$ by $\mathcal{T}_t(n)$. LoRA modules are added to query, key, value, and output projection matrices, denoted by $\mathbf{W}_{Q,t}, \mathbf{W}_{K,t}, \mathbf{W}_{V,t}$ and $\mathbf{W}_{O,t}$ Hu et al. (2022). Our attack recovers tokens from the gradients of LoRA modules added to $\mathbf{W}_{V,t}$ and $\mathbf{W}_{O,t}$. For simplicity, we describe our attack using $\mathbf{W}_{O,t}$. Same principles also apply to $\mathbf{W}_{V,t}$.

Embeddings $\mathbf{y}^{(n)}$ first enter the MSA layer. Our design of query, key, value weights, biases, and LoRA matrices produces an attention matrix $\mathbf{I}_{(N+1)\times(N+1)}$ for all heads (App. B). Then, the self-attention output for head $h \in [H]$ becomes an identity mapping of the input. We set the output projection matrix as $\mathbf{W}_{O,t} = \mathbf{0}_{D\times D}$. After the trainable LoRA modules $\mathbf{A}_{O,t} \in \mathbb{R}^{r\times D}$, $\mathbf{B}_{O,t} \in \mathbb{R}^{D\times r}$ are inserted to $\mathbf{W}_{O,t}$, the MSA output becomes,

$$\begin{aligned}
&\text{MSA}([\mathbf{y}^{(0)}, \ldots, \mathbf{y}^{(N)}]) \\
&\triangleq \left[SA_1(\mathbf{y}^{(0)}, \ldots, \mathbf{y}^{(N)}), \ldots, SA_H(\mathbf{y}^{(0)}, \ldots, \mathbf{y}^{(N)})\right] \\
&\qquad (\mathbf{W}_{O,t} + \mathbf{B}_{O,t}\mathbf{A}_{O,t}) \\
&= \begin{bmatrix}\mathbf{y}^{(0)} & \ldots & \mathbf{y}^{(N)}\end{bmatrix}^{\mathrm{T}} \mathbf{B}_{O,t}\mathbf{A}_{O,t}
\end{aligned} \tag{8}$$

The LoRA modules are then designed as,

$$\mathbf{A}_{O,t}[p,q] \triangleq \begin{cases} 1 & \text{if } p = n \text{ and } q = 2\mathcal{T}_t(n) + 2,\ n \in [r] \\ 0 & \text{otherwise} \end{cases}$$
$$\mathbf{B}_{O,t} = \mathbf{0}_{D\times r} \tag{9}$$

where $\mathbf{A}_{O,t}[p,q]$ is the element at row p, column q. The design in (9) ensures that for any token i,

$$\frac{\partial \mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(i)}}{\partial \mathbf{B}_{O,t}[:,n]} = \begin{cases} \mathbf{y}^{(i)} & \text{if } d = 2\mathcal{T}_t(n) + 2 \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{10}$$

where $\mathbf{A}_{O,t}^{\mathrm{T}}[d,:]$ is the d^{th} row of $\mathbf{A}_{O,t}^{\mathrm{T}}$, and $\mathbf{B}_{O,t}^{\mathrm{T}}[:,n]$ is the n^{th} column of $\mathbf{B}_{O,t}$. As shown later, this enables the attacker to recover the n^{th} token in $\mathcal{T}_t$ using the gradient with respect to $\mathbf{B}_{O,t}^{\mathrm{T}}[:,n]$. Since $|\mathcal{T}_t| \leq r$, from a single encoder, at most r tokens can be recovered. From (9), $\mathbf{B}_{O,t}\mathbf{A}_{O,t} = \mathbf{0}_{D\times D}$, hence,

$$\begin{bmatrix}\mathbf{y}^{(0)} & \cdots & \mathbf{y}^{(N)}\end{bmatrix}^{\mathrm{T}} \mathbf{B}_{O,t}\mathbf{A}_{O,t} = \mathbf{0} \tag{11}$$

and the output of the residual connection becomes,

$$\mathbf{u}^{(n)} \triangleq \mathbf{y}^{(n)} + \mathbf{A}_{O,t}^{\mathrm{T}}\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(n)} = \mathbf{y}^{(n)} \tag{12}$$

for $n \in \{0, \ldots, N\}$, which enters the next LN layer. The mean across the elements in $\mathbf{u}^{(n)}$ is $\mu_2^{(n)} \approx \mu$ from (6) and standard deviation is $\sigma_2^{(n)} \approx \sigma$ from (7). We set the weight parameters of this LN layer to σ and bias parameters to 0. Then, the LN output becomes,

$$\mathbf{p}^{(n)} \triangleq \frac{\mathbf{u}^{(n)} - \mu_2^{(n)}}{\sigma_2^{(n)}} \odot \sigma \times \mathbf{1}_D + 0 \times \mathbf{1}_D \approx \mathbf{y}^{(n)} \tag{13}$$

where $\mathbf{1}_D$ is a D-dimensional vector containing all 1s. Next, MLP and LN layers within this encoder are designed to propogate embeddings from (13) undistorted towards the next encoder (App. B.1).

Last Encoder (Encoder S). Our goal in this stage is to ensure that the class token embedding contains the average of all token embeddings. As only the class token is used in the final layer for loss prediction, this is critical for the final gradient to carry information from all target tokens. Hence, the parameters within the MSA layer are designed to produce an attention matrix equal to $\frac{1}{N+1}\mathbf{1}_{(N+1)\times(N+1)}$. This leads the MSA layer to output the average of all input token embeddings. Next, the parameters within the MLP and LN layers

are designed to propagate the average of the token embeddings added to the class token (through the residual connection). We denote this final embedding as $\mathbf{a}^{(0)} \approx \mathbf{p}^{(0)} + \frac{1}{N+1}\sum_{i=0}^{N}\mathbf{p}^{(i)}$. The detailed designs are delegated to App. B.2. Then,

$$\frac{\partial \mathbf{a}_d^{(0)}}{\partial \mathbf{B}_{O,t}[:,n]} \approx -\frac{1}{N+1}\frac{1}{\sigma^2 D}\sum_{i=0}^{N}\mathbf{y}_d^{(i)}\mathbf{y}_{2\mathcal{T}_t(n)+2}^{(i)}\mathbf{y}^{(i)} \approx \begin{cases} \frac{c_1^2}{(N+1)\sigma^2 D}\mathbf{y}^{(\mathcal{T}_t(n))}, & \text{if } d = 2\mathcal{T}_t(n)+1 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{14}$$

as demonstrated in App. C. Equation (14) follows from the design of the position encoding vector in (3) and (4), which ensures a large value for the multiplicative factor $-\mathbf{y}_{2\mathcal{T}_t(n)+1}^{(i)}\mathbf{y}_{2\mathcal{T}_t(n)+2}^{(i)}$ for the target token $i = \mathcal{T}_t(n)$. The detailed analysis is provided in App. C.

Classification Layer. In the final layer, the class token embedding $\mathbf{a}^{(0)}$ is used to produce the logits for each class. This is a linear layer with a weight matrix $\mathbf{W}_{CLS} \in \mathbb{R}^{D\times C}$, where C is the number of classes. The bias parameters are set to 0. We set the weight matrix,

$$\mathbf{W}_{CLS}[i,j] \triangleq \begin{cases} c_3 & \text{if } i = 2n'+1, n' \in \mathcal{T}, j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where c_3 is a large constant ($\sim 10^2$). Denote the output logit for class i as q_i. Then,

$$q_1 = \sum_{n'\in\mathcal{T}} \mathbf{a}_{2n'+1}^{(0)}\mathbf{W}_{CLS}[2n'+1,1] \tag{16}$$

Let $z_i \triangleq e^{q_i}/\sum_{c=1}^{C} e^{q_c}$ denote the predicted score for class i after softmax. The weights in (15) ensure that the score for class 1 is $z_1 \approx 1$ and 0 for all other classes, which allows the server to estimate the partial derivative of the loss function with respect to the logits. The loss function of user u is given by $\mathcal{L}_u = -\sum_{i=1}^{C} l_i \log(z_i)$, where $l_i = 1$ if the fine-tuning sample belongs to class i and 0 otherwise. Then, the gradient with respect to the n^{th} column of the fine-tuning matrix $\mathbf{B}_{O,t}$ in encoder t can be written as,

$$\frac{\partial \mathcal{L}_u}{\partial \mathbf{B}_{O,t}[:,n]} = \sum_{n'\in\mathcal{T}} \frac{\partial \mathcal{L}_u}{\partial q_1}\frac{\partial q_1}{\partial \mathbf{a}_{2n'+1}^{(0)}}\frac{\partial \mathbf{a}_{2n'+1}^{(0)}}{\partial \mathbf{B}_{O,t}[:,n]} \approx (z_1 - l_1)c_3\frac{c_1^2}{(N+1)\sigma^2 D}\mathbf{y}^{(\mathcal{T}_t(n))} \tag{17}$$

where (17) follows from (14), (15), and (16). Note that the gradient in (17) is zero if the fine-tuning sample belongs to class 1 (i.e., $l_1 = 1$), and non-zero otherwise (when $l_1 = 0$). For the former, the server can predict the true class of the fine-tuning sample, and set the weights of a different class to c_3 in (15) in the next round, to obtain a non-zero gradient. Finally, the server can recover the target token embedding $\mathbf{y}^{(\mathcal{T}_t(n))}$ by dividing (17) by a factor $c_3 c_1^2/((N+1)\sigma^2 D)$, and recover the target word embedding $\mathbf{x}^{(\mathcal{T}_t(n))}$ by subtracting the position encoding vector, $\mathbf{x}^{(\mathcal{T}_t(n))} = \mathbf{y}^{(\mathcal{T}_t(n))} - \mathbf{e}^{(\mathcal{T}_t(n))}$, for $t \in [S-1], n \in \mathcal{T}_t$. Fig. 4 illustrates token reconstruction from multiple encoders.

Note that our work adopts the standard adversarial server model in FL Fowl et al. (2023); Sami et al. (2025), which attempts to recover user data by tampering with model parameters, without regard for preserving model accuracy. FL typically runs in the background without active monitoring, making malicious behavior less likely to be detected Ramaswamy et al. (2019); Fowl et al. (2023). Since our attack is one-shot and reconstructs data in a single training round (e.g., at an early stage when accuracy is low), even if a user later notices degraded performance, the server may already have recovered the sensitive data. Investigating stealthier attacks that preserve training performance such as Feng and Tramèr (2024) is an interesting future direction.

5 EXPERIMENTS

Our experiments seek to answer the following questions:

- How does MineGrad perform in recovering tokens from text sequences?
- How does MineGrad perform with different ranks of LoRA matrices?
- How can recovery rate be increased by leveraging LoRA matrices from multiple encoders?
- How is the reconstruction performance affected by increasing the sequence length and batch size?
- How does MineGrad perform in recovering images?

Setup. We initially consider federated fine-tuning for text classification with 100 users, each training locally and sending the LoRA gradients to the server. From each gradient, the server reconstructs the fine-tuning data using MineGrad. Experiments are run on a 24 core AMD Ryzen with NVIDIA RTX4000.

Datasets and model architecture. We demonstrate the results with multiple datasets, which includes Yahoo Answers Topics, Yelp Review, AG's News, DBPedia and TREC-6 Zhang et al. (2015); Li and Roth (2002). For the pretrained model, we consider BERT-base, BERT-large Devlin et al. (2019), RoBERTa-base and RoBERTa-large Liu et al. (2019).

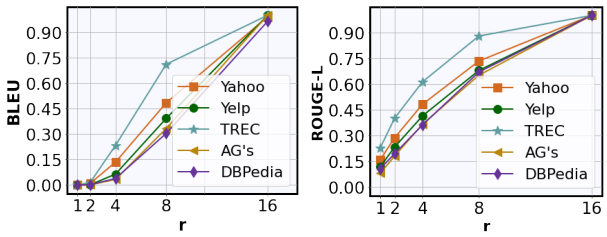
Performance metrics. To evaluate the performance of MineGrad, we measure BLEU and ROUGE-L scores

Table 1: Average BLEU and ROUGE-L scores over 100 samples for different datasets and models.

Dataset	RoBERTa-base		RoBERTa-large		BERT-base		BERT-large	
	BLEU	ROUGE-L	BLEU	ROUGE-L	BLEU	ROUGE-L	BLEU	ROUGE-L
AG's	1.0	1.0	1.0	1.0	1.0	1.0	0.82	0.94
DBPedia	0.99	1.0	0.99	1.0	1.0	1.0	0.79	0.92
Yahoo	1.0	1.0	1.0	1.0	1.0	1.0	0.81	0.94
Yelp	0.98	1.0	0.98	1.0	0.98	1.0	0.73	0.91
TREC	1.0	1.0	1.0	1.0	0.98	1.0	0.82	0.94

Table 2: Recovered tokens from different datasets (rank $r = 4$, RoBERTa-base).

Dataset		Fine-tuning sample
AG's	Ground-truth	Microsoft Sends Digital Business Cards New InterConnect 2004 software automatically updates contact info.
	Recovered	'Microsoft', 'S', 'ends', 'Digital', 'Business', 'Cards', 'New', 'Inter', 'Connect', '2004', 'software', 'automatically', 'updates', 'contact', 'info','.'
DBPedia	Ground-truth	Palacete de Belomonte is a historic palace in Porto Portugal.
	Recovered	'Pal', 'ac', 'ete', 'de', 'Bel', 'omon', 'te', 'is', 'a', 'historic', 'palace', 'in', 'Port', 'o', 'Portugal', '.'
Yahoo	Ground-truth	Heavy water what is the role that heavy water plays in the nuclear explosion process?
	Recovered	'Heavy', 'water', 'what', 'is', 'the', 'role', 'that', 'heavy', 'water', 'plays', 'in', 'the', 'nuclear', 'explosion', 'process', '?'
Yelp	Ground-truth	If you like eastern north carolina BBQ, you'll be in heaven.
	Recovered	'If', 'you', 'like', 'eastern', 'north', 'car', 'olina', 'BBQ', ',', 'you', ' 'll ', 'be', 'in', 'heaven', '.'
TREC	Ground-truth	What American League baseball team 's worst finish between 1926 and 1964 was fourth?
	Recovered	'What', 'American', 'League', 'baseball', 'team', ' ' ', 's', 'worst', 'finish', 'between', '1926', 'and', '1964', 'was', 'fourth', '?'

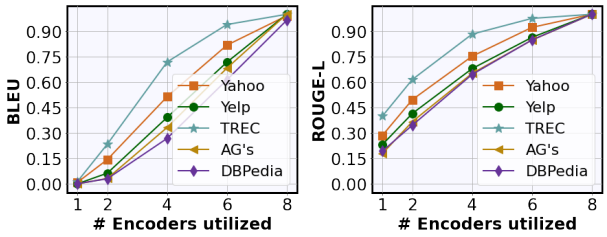


(a) BLEU score (b) ROUGE-L score

Figure 5: Reconstruction quality with varying r.

Papineni et al. (2002); Lin (2004) between recovered and ground-truth texts. Higher score implies better reconstruction.

Hyperparameters. We use $c_1 = 10^2$, $c_2 = 3$ and $c_3 = 10^2$ in equations (4), (15). Unless stated otherwise, we show the results with RoBERTa-base, rank $r = 4$, and sequence length 16.



(a) BLEU score (b) ROUGE-L score

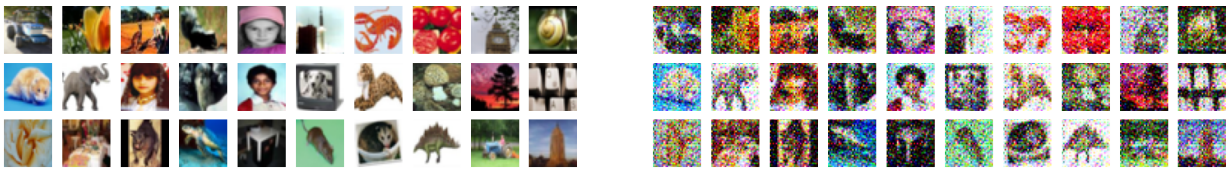
Figure 6: Utilizing multiple encoders ($r = 2$).

Results. In Table 1, we report the average scores across 100 samples for different datasets and architectures under the same hyperparameters. As we observe, for all the models and datasets, MineGrad achieves ROUGE-L and BLEU scores equal to (or close to) 1, implying perfect (or near perfect) reconstruction. In Table 2, we illustrate examples of tokens recovered from the shared gradients. From the gradients corresponding to each encoder, we recover 4 tokens (as $r = 4$). To recover all 16 tokens, we use 4 encoders.

Reconstruction rate vs. rank of LoRA matrices. In Fig. 5, we study the impact of rank r on reconstruction for a single encoder. We observe that reconstruction quality enhances when r increases. This suggests adopting a small r could be suffiicent to preserve privacy. However, in Fig. 6 we show that this is not the case. By deploying multiple encoders, we can achieve the same BLEU and ROUGE-L scores for rank $r = 2$ as for $r = 16$ with a single encoder.

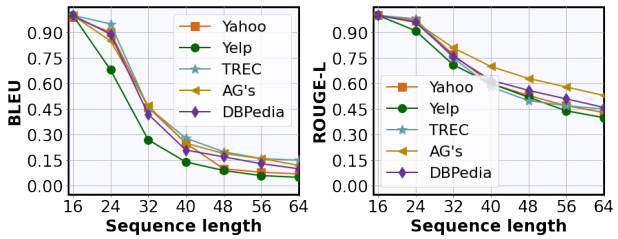
Reconstruction rate vs. sequence length. Fig. 8 shows our results for increasing sequence length with RoBERTa-base, using the same rank and number of encoders as Tables 1–2. As expected, reconstruction performance degrades as the sequence length increases. Performance can be further enhanced by tampering with pretrained word embeddings (App. E.1).

Comparison with DAGER Petrov et al. (2024). In Fig. 9, we compare the performance of MineGrad and DAGER. We observe that the reconstruction performance of DAGER heavily degrades when the rank is lower than the sequence length 16. In contrast, MineGrad retains the attack success even with a rank as small as 4.



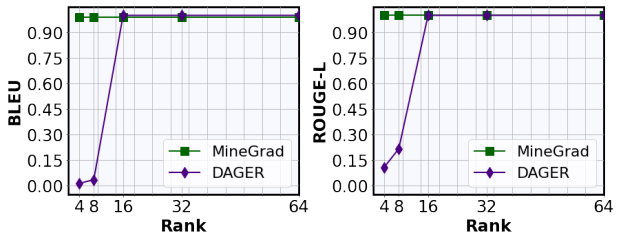
(a) Ground-truth (b) Recovered

Figure 7: Image reconstruction ($r = 4$) for CIFAR-100.



(a) BLEU score (b) ROUGE-L score

Figure 8: Reconstruction with varying sequence length.



(a) BLEU score (b) ROUGE-L score

Figure 9: Reconstruction performance for DAGER vs. MineGrad (Yahoo Answers dataset).

Defense mechanisms. In App. E.2, we present the attack performance against popular defenses such as secure aggregation Bonawitz et al. (2017), pruning Lin et al. (2018), and noise Abadi et al. (2016).

Reconstruction for a batch of sequences. We next discuss how our attack performs for larger batch sizes (i.e., multiple sequences). Under this setup, we recover an average of multiple word embeddings from the gradient each associated with a particular sequence in the batch. From this average, the target word embeddings can be retrieved via a similarity-based search method across the vocabulary. Details are provided in App. D. In Table 3, we report the recovery performance. As we observe, even for a batch size of 64, up to 52.2% of the tokens can be recovered.

Results for image recovery. We next study how MineGrad performs on vision transformers (ViT) to recover image samples Dosovitskiy et al. (2021). In Table 4, we report the mean and standard deviation for LPIPS scores Zhang et al. (2018) (between recovered and original images) across 100 images from CIFAR-10 and CIFAR-100 datasets Krizhevsky and Hinton (2009). In Fig. 7, we provide the results for some sample images. We observe that the recovered images are close to original images.

Table 3: Percentage of tokens recovered under different batch sizes across multiple datasets.

Batch size	**8**	**16**	**32**	**64**
Yahoo	99.7	89.4	64.9	45.9
Yelp	99.5	88.0	65.3	44.2
AGnews	99.8	91.1	66.3	46.8
Trec	97.8	85.8	65.9	52.2
DBpedia	99.8	89.9	60.2	41.6

Table 4: LPIPS score across 100 images.

	CIFAR-10	**CIFAR-100**
LPIPS	0.20 ± 0.04	0.21 ± 0.06

Decoder-based architectures. In App. E.4 we further provide our results with GPT-2 Radford et al. (2019).

Ablation study. In App. E.3, we provide an ablation study by varying the hyperparameters c_1, c_2 and additional examples of recovered images/text sequences in Apps. E.5 and E.6.

App. E.7 further demonstrates attack performance when LayerNorm and word embeddings are trainable. In App. E.8, we consider the performance in the presence of adaptive optimizers, such as Adam and AdaGrad. Apps. E.9 and E.10 demonstrate the performance under regularization, including Label Smoothing and Dropout.

6 CONCLUSION

We study how an adversary can recover sensitive fine-tuning data through malicious tampering with the pretrained model and fine-tuning LoRA modules. In addition to new privacy threats, we also demonstrate several new design principles that can be utilized in future studies. We show the efficacy of our attack across multiple models and datasets. Our results highlight the need for verifiable defense mechanisms for PEFT.

ACKNOWLEDGEMENT

This work was supported in part by the NSF CAREER Award CCF-2144927, UCR OASIS Fellowship, and OUSD (R&E)/RT&L and was accomplished under Cooperative Agreement Number W911NF-20-2-0267. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR, ARL and OUSD(R&E)/RT&L or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. (2018). The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (Neurips)*, pages 5977–5987.

Balunovic, M., Dimitrov, D. I., Jovanovic, N., and Vechev, M. T. (2022). LAMP: extracting text from gradients with language model priors. In *Advances in Neural Information Processing Systems (Neurips)*.

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. (2020). Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1253–1269.

Bian, J., Wang, L., Zhang, L., and Xu, J. (2024). Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement. *Arxiv*.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Cai, H., Gan, C., Zhu, L., and Han, S. (2020). Tinytl: Reduce memory, not parameters for efficient on-device learning. In *Neural Information Processing Systems*.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. X. (2018). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. (2021). Extracting training data from large language models. In *30th USENIX Security Symposium*, pages 2633–2650.

Chen, H., Tu, C., Li, Z., Shen, H., and Chao, W. (2023). On the importance and applicability of pre-training for federated learning. In *The Eleventh International Conference on Learning Representations, ICLR*.

Cho, Y. J., Liu, L., Xu, Z., Fahrezi, A., and Joshi, G. (2024). Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 12903–12913. Association for Computational Linguistics.

Chu, H., Geiping, J., Fowl, L. H., Goldblum, M., and Goldstein, T. (2023). Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *The Eleventh International Conference on Learning Representations, ICLR*.

Deng, J., Wang, Y., Li, J., Wang, C., Shang, C., Liu, H., Rajasekaran, S., and Ding, C. (2021). TAG: gradient attack on transformer-based language models. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 3600–3610. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). 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, NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16

words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR*.

Feng, S. and Tramèr, F. (2024). Privacy backdoors: Stealing data with corrupted pretrained models. In *Forty-first International Conference on Machine Learning, ICML*.

Fowl, L. H., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. (2022). Robbing the fed: Directly obtaining private data in federated learning with modified models. In *The Tenth International Conference on Learning Representations, ICLR*.

Fowl, L. H., Geiping, J., Reich, S., Wen, Y., Czaja, W., Goldblum, M., and Goldstein, T. (2023). Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *The Eleventh International Conference on Learning Representations, ICLR*.

Gao, Y., Xie, Y., Deng, H., and Zhu, Z. (2025). Gradient inversion attack in federated learning: Exposing text data through discrete optimization. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2582–2591.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gupta, S., Huang, Y., Zhong, Z., Gao, T., Li, K., and Chen, D. (2022). Recovering private text in federated learning of language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hatamizadeh, A., Yin, H., Roth, H., Li, W., Kautz, J., Xu, D., and Molchanov, P. (2022). Gradvit: Gradient inversion of vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10011–10020. IEEE.

Hayou, S., Ghosh, N., and Yu, B. (2024). Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning, ICML*.

He, P., Liu, X., Gao, J., and Chen, W. (2021). Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Hitaj, B., Ateniese, G., and Pérez-Cruz, F. (2017). Deep models under the gan: Information leakage from collaborative deep learning. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR*.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. (2021). Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems (Neurips)*, pages 7232–7241.

JianHao, Z., Lv, C., Wang, X., Wu, M., Liu, W., Li, T., Ling, Z., Zhang, C., Zheng, X., and Huang, X. (2024). Promoting data and model privacy in federated learning through quantized LoRA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10501–10512. Association for Computational Linguistics.

Kariyappa, S., Guo, C., Maeng, K., Xiong, W., Suh, G. E., Qureshi, M. K., and Lee, H. S. (2023). Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In *International Conference on Machine Learning, ICML*.

Kim, Y., Kim, J., Mok, W., Park, J., and Lee, S. (2023). Client-customized adaptation for parameter-efficient federated learning. In *Findings of the Association for Computational Linguistics: ACL*, pages 1159–1172.

Kowsher, M., Esmaeilbeig, T., Yu, C., Soltanalian, M., and Yousefi, N. (2024). Rocoft: Efficient finetuning of large language models with row-column updates. *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Li, B., Hu, Y., Nie, X., Han, C., Jiang, X., Guo, T., and Liu, L. (2023). Dropkey for vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22700–22709.

Li, X. and Roth, D. (2002). Learning question classifiers. In *International Conference on Computational Linguistics*.

Li, X., Zmigrod, R., Ma, Z., Liu, X., and Zhu, X. (2024). Fine-tuning language models with differential privacy through adaptive noise allocation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 8368–8375. Association for Computational Linguistics.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 4582–4597.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. (2018). Deep gradient compression: Reducing the communication bandwidth for distributed training. In *6th International Conference on Learning Representations, ICLR*.

Liu, R., Wang, T., Cao, Y., and Xiong, L. (2024). Precurious: How innocent pre-trained language models turn into privacy traps. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 3511–3524. ACM.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *Arxiv*.

Lu, J., Zhang, X. S., Zhao, T., He, X., and Cheng, J. (2022). APRIL: finding the achilles' heel on privacy for vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10041–10050. IEEE.

Mahabadi, R. K., Henderson, J., and Ruder, S. (2021). Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1022–1035.

Mireshghallah, F., Goyal, K., Uniyal, A., Berg-Kirkpatrick, T., and Shokri, R. (2022). Quantifying privacy risks of masked language models using membership inference attacks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 8332–8347. Association for Computational Linguistics.

Nasr, M., Shokri, R., and Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *IEEE Symposium on Security and Privacy (SP)*, pages 739–753.

Nguyen, J., Wang, J., Malik, K., Sanjabi, M., and Rabbat, M. G. (2023a). Where to begin? on the impact of pre-training and initialization in federated learning. In *The Eleventh International Conference on Learning Representations, ICLR*.

Nguyen, T. D. T., Lai, P., Tran, K., Phan, N., and Thai, M. T. (2023b). Active membership inference attack under local differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pages 5714–5730.

Panda, A., Choquette-Choo, C. A., Zhang, Z., Yang, Y., and Mittal, P. (2024). Teach llms to phish: Stealing private information from language models. In *The Twelfth International Conference on Learning Representations, ICLR*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Pasquini, D., Francati, D., and Ateniese, G. (2022). Eluding secure aggregation in federated learning via model inconsistency. In *Conference on Computer and Communications Security, CCS*, pages 2429–2443. ACM.

Petrov, I., Dimitrov, D. I., Baader, M., Müller, M. N., and Vechev, M. T. (2024). DAGER: exact gradient inversion for large language models. In *Advances in Neural Information Processing Systems (Neurips)*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Ramaswamy, S., Mathews, R., Rao, K., and Beaufays, F. (2019). Federated learning for emoji prediction in a mobile keyboard. *Arxiv*.

Sami, H. U., Sen, S., Roy-Chowdhury, A. K., Krishnamurthy, S. V., and Guler, B. (2025). Gradient inversion attacks on parameter-efficient fine-tuning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10224–10234.

Scheliga, D., Maeder, P., and Seeland, M. (2023). Dropout is NOT all you need to prevent gradient leakage. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, pages 9733–9741.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2016). Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.

Shysheya, A., Bronskill, J., Patacchiola, M., Nowozin, S., and Turner, R. E. (2023). Fit: Parameter efficient few-shot transfer learning for personalized and federated image classification. In *The Eleventh International Conference on Learning Representations, ICLR*.

Song, L. and Mittal, P. (2021). Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium*, pages 2615–2632. USENIX Association.

Sun, Y., Li, Z., Li, Y., and Ding, B. (2024). Improving lora in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations, ICLR*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *Arxiv*.

Tramèr, F., Shokri, R., Joaquin, A. S., Le, H., Jagielski, M., Hong, S., and Carlini, N. (2022). Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 2779–2792. ACM.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (Neurips)*, pages 5998–6008.

Vero, M., Balunovic, M., Dimitrov, D. I., and Vechev, M. T. (2023). Tableak: Tabular data leakage in federated learning. In *International Conference on Machine Learning, ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 35051–35083.

Wang, Y., Lin, L., and Chen, J. (2022). Communication-efficient adaptive federated learning. In *International conference on machine learning*, pages 22802–22838. PMLR.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H. (2019). Beyond inferring class representatives: User-level privacy leakage from federated learning. In *2019 IEEE Conference on Computer Communications, INFOCOM*, pages 2512–2520.

Wen, Y. and Chaudhuri, S. (2024). Batched low-rank adaptation of foundation models. In *The Twelfth International Conference on Learning Representations, ICLR*.

Wen, Y., Marchyok, L., Hong, S., Geiping, J., Goldstein, T., and Carlini, N. (2024). Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. *Arxiv*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics.

Wu, X., Huang, F., Hu, Z., and Huang, H. (2023). Faster adaptive federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10379–10387.

Yao, D. (2024). Risks when sharing lora fine-tuned diffusion model weights. *Arxiv*.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. (2021). See through gradients: Image batch recovery via gradinversion. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*.

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., Kulkarni, J., Lee, Y. T., Manoel, A., Wutschitz, L., Yekhanin, S., and Zhang, H. (2022). Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR*.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. (2023a). Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR*.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 586–595.

Zhang, X., Zhao, J. J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Neural Information Processing Systems*.

Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., and Song, D. (2020). The secret revealer: Generative model-inversion attacks against deep neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 250–258.

Zhang, Z., Yang, Y., Dai, Y., Wang, Q., Yu, Y., Qu, L., and Xu, Z. (2023b). Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL*, pages 9963–9977.

Zhao, B., Tu, H., Wei, C., Mei, J., and Xie, C. (2024a). Tuning layernorm in attention: Towards efficient multi-modal LLM finetuning. In *The Twelfth International Conference on Learning Representations, ICLR*.

Zhao, J., Sharma, A., Elkordy, A., Ezzeldin, Y. H., Avestimehr, S., and Bagchi, S. (2024b). Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*.

Zhu, L., Liu, Z., and Han, S. (2019). Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX

A BROADER IMPACT

We demonstrate how fine-tuning samples can be recovered from the LoRA gradients in an FL setup. In practice, an adversarial server can get access to private fine-tuning data of a victim user through malicious tampering with the fine-tuning protocol. Pretrained models can be downloaded from a compromised source without a stringent verification protocol. In FL, users implicitly trust the models shared by the server. The server can leverage this scenario and send a poisonous pretrained model to the users. In addition, users rely on the server for receiving the updated fine-tuning parameters in each training round. By deploying a poisonous design of the pretrained model and fine-tuning modules, the server can successfully captivate the fine-tuning data inside a user's local gradient. Without adopting further defense mechanisms, such fine-tuning does not ensure privacy. Therefore, it is important to further adopt formal mechanisms to guarantee authenticity of the models received from the server. By demonstrating the recoverability of sensitive fine-tuning samples with lightweight analytical attacks, without access to heavy computational resources, we hope to motivate new incentives towards employing robust defense mechanisms with provable guarantees in practice.

B DETAILS OF THE ATTACK DESIGN

In this section, we describe the malicious design details of the encoders.

B.1 Design of Target Encoders

MSA layer. In the MSA layer, we have the embeddings $\mathbf{y}^{(n)}$ as inputs. For head $h \in [H]$, we denote the query, key and value weight matrices of a target encoder $t \in [S-1]$ as $\mathbf{W}^h_{Q,t}, \mathbf{W}^h_{K,t}$ and $\mathbf{W}^h_{V,t}$, and biases as $\mathbf{b}^h_{Q,t}, \mathbf{b}^h_{K,t}$ and $\mathbf{b}^h_{V,t}$, respectively, which are designed as,

$$\mathbf{W}^h_{Q,t} = \mathbf{W}^h_{K,t} = \mathbf{W}^h_{V,t} = \mathbf{I}_{\overline{D}\times\overline{D}} \tag{18}$$

$$\mathbf{b}^h_{Q,t} = \mathbf{b}^h_{K,t} = \mathbf{b}^h_{V,t} = \mathbf{0} \tag{19}$$

The LoRA matrices added to query, key and value weights are set to $\mathbf{0}$. Then, query, key and value for head h are computed as,

$$\begin{aligned}
\mathbf{Q}^h_t &\triangleq \begin{bmatrix}\mathbf{W}^h_{Q,t}(\mathbf{y}^{(0)})_h & \cdots & \mathbf{W}^h_{Q,t}(\mathbf{y}^{(N)})_h\end{bmatrix}^{\mathrm{T}} \\
\mathbf{K}^h_t &\triangleq \begin{bmatrix}\mathbf{W}^h_{K,t}(\mathbf{y}^{(0)})_h & \cdots & \mathbf{W}^h_{K,t}(\mathbf{y}^{(N)})_h\end{bmatrix}^{\mathrm{T}} \\
\mathbf{V}^h &\triangleq \begin{bmatrix}\mathbf{W}^h_{V,t}(\mathbf{y}^{(0)})_h & \cdots & \mathbf{W}^h_{V,t}(\mathbf{y}^{(N)})_h\end{bmatrix}^{\mathrm{T}}
\end{aligned} \tag{20}$$

Now, following (5), the self-attention output for head h becomes,

$$\begin{aligned}
SA_h([\mathbf{y}^{(0)} \ \cdots \ \mathbf{y}^{(N)}]) \triangleq softmax(\mathbf{Q}^h_t(\mathbf{K}^h_t)^{\mathrm{T}}/\sqrt{\overline{D}})\mathbf{V}^h_t &\cong \mathbf{I}_{(N+1)\times(N+1)}\mathbf{V}^h_t \\
&= \begin{bmatrix}(\mathbf{y}^{(0)})_h & \cdots & (\mathbf{y}^{(N)})_h\end{bmatrix}^{\mathrm{T}}
\end{aligned} \tag{21}$$

Hence, our malicious design of query, key, value weights, biases and LoRA matrices along with the condition from (5) produces an identity mapping of the input embeddings. Finally, as in (8), we have,

$$MSA([\mathbf{y}^{(0)} \ \cdots \ \mathbf{y}^{(N)}]) = \begin{bmatrix}(\mathbf{y}^{(0)}) & \cdots & (\mathbf{y}^{(N)})\end{bmatrix}^{\mathrm{T}} \mathbf{B}_{O,t}\mathbf{A}_{O,t} \tag{22}$$

MLP layer. Embeddings $\mathbf{p}^{(n)}$ from (13) enters the MLP layer. We set all the pretrained weight parameters in the MLP layer to 0 to produce zero output. Then, through the residual connection, input to the next LN layer is equal to $\mathbf{p}^{(n)}$ from (13). By following the same design as LN layer parameters in (13), the output of this LN layer is approximately equal to $\mathbf{p}^{(n)}$.

B.2 Design of the Last Encoder

MSA layer. For the MSA layer in the last encoder (Encoder S), we have embeddings $\mathbf{p}^{(n)}$ as inputs for $n \in \{0, \ldots, N\}$. For head $h \in [H]$, the query, key and value parameters $\mathbf{W}^h_{Q,S}, \mathbf{W}^h_{K,S}$ and $\mathbf{W}^h_{V,S}$ are designed as,

$$\mathbf{W}^h_{Q,S} = \mathbf{W}^h_{K,S} = \mathbf{0}_{\overline{D}\times\overline{D}}, \quad \mathbf{W}^h_{V,S} = \mathbf{I}_{\overline{D}\times\overline{D}} \tag{23}$$

and biases $\mathbf{b}^h_{Q,S}, \mathbf{b}^h_{K,S}, \mathbf{b}^h_{V,S}$ are designed as,

$$\mathbf{b}^h_{Q,S} = \mathbf{b}^h_{Q,S} = \mathbf{b}^h_{V,S} = \mathbf{0} \tag{24}$$

The LoRA modules added to query, key and value weights are set to $\mathbf{0}$. Then, query, key and values for head h are,

$$\mathbf{Q}^h_S \triangleq \left[\mathbf{W}^h_{Q,S}(\mathbf{p}^{(0)})_h \quad \cdots \quad \mathbf{W}^h_{Q,S}(\mathbf{p}^{(N)})_h\right]^{\mathrm{T}} = \mathbf{0} \tag{25}$$

$$\mathbf{K}^h_S \triangleq \left[\mathbf{W}^h_{K,S}(\mathbf{p}^{(0)})_h \quad \cdots \quad \mathbf{W}^h_{K,S}(\mathbf{p}^{(N)})_h\right]^{\mathrm{T}} = \mathbf{0} \tag{26}$$

$$\mathbf{V}^h_S \triangleq \left[\mathbf{W}^h_{V,S}(\mathbf{p}^{(0)})_h \quad \cdots \quad \mathbf{W}^h_{V,S}(\mathbf{p}^{(N)})_h\right]^{\mathrm{T}} = \left[(\mathbf{p}^{(0)})_h \quad \cdots \quad (\mathbf{p}^{(N)})_h\right]^{\mathrm{T}} \tag{27}$$

Hence, the attention matrix becomes a matrix with all elements equal to $\frac{1}{N+1}$,

$$softmax(\mathbf{Q}^h_S(\mathbf{K}^h_S)^{\mathrm{T}}/\sqrt{\overline{D}}) \cong \frac{1}{N+1}\mathbf{1}_{(N+1)\times(N+1)}$$

where $\mathbf{1}_{(N+1)\times(N+1)}$ represents a matrix containing all 1s. The self-attention output for head h is computed as,

$$\begin{aligned} SA_h([\mathbf{p}^{(0)} \quad \cdots \quad \mathbf{p}^{(N)}]) &\triangleq softmax(\mathbf{Q}^h_S(\mathbf{K}^h_S)^{\mathrm{T}}/\sqrt{\overline{D}})\mathbf{V}^h_S \\ &= \left[\tfrac{1}{N+1}\textstyle\sum_{i=0}^N(\mathbf{p}^{(i)})_h \quad \cdots \quad \tfrac{1}{N+1}\sum_{i=0}^N(\mathbf{p}^{(i)})_h\right]^{\mathrm{T}} \end{aligned} \tag{28}$$

We then set $\mathbf{W}_{O,S} = \mathbf{I}_{D\times D}$, and LoRA modules $\mathbf{A}_{O,S}, \mathbf{B}_{O,S}$ to $\mathbf{0}$, after which the MSA output becomes,

$$\begin{aligned} &MSA([\mathbf{p}^{(0)} \quad \cdots \quad \mathbf{p}^{(N)}]) \\ &= \left[\tfrac{1}{N+1}\textstyle\sum_{i=0}^N \mathbf{p}^{(i)} \quad \cdots \quad \tfrac{1}{N+1}\sum_{i=0}^N \mathbf{p}^{(i)}\right]^{\mathrm{T}} \end{aligned} \tag{29}$$

and ensures the average embeddings propagate undistorted. After the residual connection, class token embedding is,

$$\mathbf{v}^{(0)} \triangleq \mathbf{p}^{(0)} + \frac{1}{N+1}\sum_{i=0}^N \mathbf{p}^{(i)} \tag{30}$$

We design the parameters of the subsequent LN layer following the same intuition as the LN layer from the preceding encoders, which produces an identity mapping.

MLP layer. All weight parameters in the MLP layer are set to 0 to produce zero output. After going through the residual connection and the LN layer (similar design intuition as the previous LN layer), the output of the last encoder is,

$$\mathbf{a}^{(n)} \approx \mathbf{v}^{(n)} \text{ for } n \in \{0, \ldots, N\} \tag{31}$$

C GRADIENT COMPUTATION

In this section, we describe how we obtain (14). Since $\mathbf{a}^{(0)} \approx \mathbf{v}^{(0)}$ from (31), we have for $n' \in \mathcal{T}$ (the set of target tokens),

$$\frac{\partial \mathbf{a}^{(0)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} \approx \frac{\partial \mathbf{v}^{(0)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} \tag{32}$$

$$= \frac{\partial(\mathbf{p}^{(0)}_{2n'+1} + \frac{1}{N+1}\sum_{i=0}^{N}\mathbf{p}^{(i)}_{2n'+1})}{\partial \mathbf{B}_{O,t}[:,n]} \tag{33}$$

$$= \frac{\partial \mathbf{p}^{(0)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} + \frac{\partial \frac{1}{N+1}\sum_{i=0}^{N}\mathbf{p}^{(i)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} \tag{34}$$

where (33) follows from (30). From (13), $\mathbf{p}^{(i)}_{2n'+1} = \frac{\mathbf{y}^{(i)}_{2n'+1} + \mathbf{A}^{\mathrm{T}}_{O,t}[2n'+1,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} - \mu^{(i)}_2}{\sigma^{(i)}_2}\sigma$. Hence,

$$\frac{\partial \mathbf{p}^{(i)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} = \frac{\partial \frac{\mathbf{y}^{(i)}_{2n'+1} + \mathbf{A}^{\mathrm{T}}_{O,t}[2n'+1,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} - \mu^{(i)}_2}{\sigma^{(i)}_2}\sigma}{\partial \mathbf{B}_{O,t}[:,n]} \tag{35}$$

Note that the mean of $\mathbf{u}^{(i)}$ in (12) is,

$$\mu^{(i)}_2 = \frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}^{(i)}_d + \mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}) = \frac{1}{D}\sum_{d=1}^{D}\mathbf{y}^{(i)}_d \approx 0 \tag{36}$$

which follows from (6). Now,

$$\frac{\partial \mu^{(i)}_2}{\partial \mathbf{B}_{O,t}[:,n]} = \frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]}\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}^{(i)}_d + \mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}) \tag{37}$$

$$= \frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]}\frac{1}{D}\sum_{d=1}^{D}\mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} = \frac{1}{D}\mathbf{y}^{(i)} \tag{38}$$

where (38) follows from (10). The standard deviation of $\mathbf{u}^{(i)}$ in (12) is,

$$\sigma^{(i)}_2 \triangleq \sqrt{\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}^{(i)}_d + \mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} - \mu^{(i)}_2)^2} \approx \sqrt{\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}^{(i)}_d - 0)^2} \approx \sigma \tag{39}$$

which follows from (7). Next,

$$\frac{\partial \frac{\mu^{(i)}_2}{\sigma^{(i)}_2}}{\partial \mathbf{B}_{O,t}[:,n]} = \frac{(\sigma^{(i)}_2)\frac{\partial \mu^{(i)}_2}{\partial \mathbf{B}_{O,t}[:,n]} - \mu^{(i)}_2\frac{\partial \sigma^{(i)}_2}{\partial \mathbf{B}_{O,t}[:,n]}}{(\sigma^{(i)}_2)^2} \tag{40}$$

$$\approx \frac{1}{\sigma^{(i)}_2}\frac{\partial \mu^{(i)}_2}{\partial \mathbf{B}_{O,t}[:,n]} \tag{41}$$

$$= \frac{1}{\sigma^{(i)}_2}\frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]}\frac{1}{D}\sum_{d=1}^{D}(\mathbf{y}^{(i)}_d + \mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}) \tag{42}$$

$$= \frac{1}{\sigma^{(i)}_2}\frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]}\frac{1}{D}\sum_{d=1}^{D}\mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} \tag{43}$$

$$\approx \frac{1}{\sigma}\frac{1}{D}\mathbf{y}^{(i)} \tag{44}$$

where (41) follows since $\mu^{(i)}_2 \approx 0$ in (36) and (44) follows from (10) and (39). Next,

$$\frac{\partial \frac{\mathbf{y}^{(i)}_{2n'+1}}{\sigma^{(i)}_2}}{\partial \mathbf{B}_{O,t}[:,n]}$$

$$= \mathbf{y}^{(i)}_{2n'+1}\frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]}\frac{1}{\sigma^{(i)}_2} \tag{45}$$

$$= -\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^2} \frac{\partial \sigma_2^{(i)}}{\partial \mathbf{B}_{O,t}[:,n]} \tag{46}$$

$$= -\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^2} \frac{\partial ((\sigma_2^{(i)})^2)^{\frac{1}{2}}}{\partial \mathbf{B}_{O,t}[:,n]} \tag{47}$$

$$= -\frac{1}{2}\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3} \frac{\partial ((\sigma_2^{(i)})^2)}{\partial \mathbf{B}_{O,t}[:,n]} \tag{48}$$

$$= -\frac{1}{2}\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3} \frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]} \frac{1}{D} \sum_{d=1}^{D} (\mathbf{y}_d^{(i)} + \mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(i)} - \mu_2^{(i)})^2 \tag{49}$$

$$\begin{aligned} = -\frac{1}{2}\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3} \frac{1}{D} 2 \sum_{d=1}^{D} (\mathbf{y}_d^{(i)} + \mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(i)} - \mu_2^{(i)}) \\ \times \frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]} (\mathbf{y}_d^{(i)} + \mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(i)} - \mu_2^{(i)}) \end{aligned} \tag{50}$$

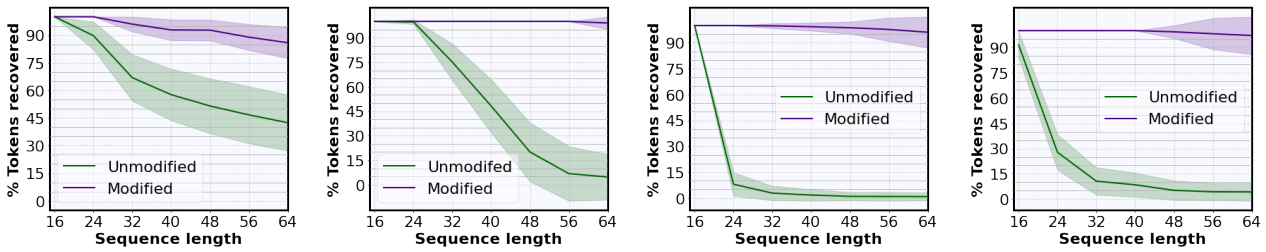
$$\approx -\mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3} \frac{1}{D} \sum_{d=1}^{D} \mathbf{y}_d^{(i)} \frac{\partial}{\partial \mathbf{B}_{O,t}[:,n]} (\mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}}\mathbf{y}^{(i)} - \mu_2^{(i)}) \tag{51}$$

$$\begin{aligned} = \mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3 D^2} \sum_{d=1, d\neq 2\mathcal{T}_t(n)+2}^{D} \mathbf{y}_d^{(i)} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} - \frac{\mathbf{y}^{(i)}_{2n'+1}}{(\sigma_2^{(i)})^3 D} \mathbf{y}^{(i)}_{2\mathcal{T}_t(n)+2} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} \\ + \mathbf{y}^{(i)}_{2n'+1} \frac{1}{(\sigma_2^{(i)})^3} \frac{1}{D^2} \mathbf{y}^{(i)}_{2\mathcal{T}_t(n)+2} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} \end{aligned} \tag{52}$$

$$\begin{aligned} \approx \mathbf{y}^{(i)}_{2n'+1} \frac{1}{\sigma^3 D^2} \sum_{d=1, d\neq 2\mathcal{T}_t(n)+2}^{D} \mathbf{y}_d^{(i)} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} - \frac{\mathbf{y}^{(i)}_{2n'+1}}{\sigma^3 D} \mathbf{y}^{(i)}_{2\mathcal{T}_t(n)+2} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} \\ + \mathbf{y}^{(i)}_{2n'+1} \frac{1}{\sigma^3} \frac{1}{D^2} \mathbf{y}^{(i)}_{2\mathcal{T}_t(n)+2} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} \end{aligned} \tag{53}$$

where (51) follows since $\mu_2^{(i)} \approx 0$ and $\mathbf{A}_{O,t}^{\mathrm{T}}[d,:]\mathbf{B}_{O,t}^{\mathrm{T}} = 0$ for all $d \in [D]$. Equation (52) follows from (10) and (38). Equation (53) follows since $\sigma_2^{(i)} \approx \sigma$ in (39). In (53), compared to second term, the first and third terms have negligible impact since they are multiplied by an additional factor of $\frac{1}{D}$, which is very small for $D = 768$ (base architecture) and $D = 1024$ (large architecture). For the second term, if $i = \mathcal{T}_t(n)$ and $n' = \mathcal{T}_t(n)$, then

$$-\mathbf{y}^{(i)}_{2n'+1} \frac{1}{\sigma^3} \frac{1}{D} \mathbf{y}^{(i)}_{2\mathcal{T}_t(n)+2} \begin{bmatrix} \mathbf{y}_1^{(i)} \\ \mathbf{y}_2^{(i)} \\ \vdots \\ \mathbf{y}_D^{(i)} \end{bmatrix} \approx c_1^2 \frac{1}{\sigma^3} \frac{1}{D} \mathbf{y}^{(i)} \tag{54}$$



(a) RoBERTa-base (b) RoBERTa-large (c) BERT-base (d) BERT-large

Figure 10: Reconstruction quality for unmodified vs. modified word embeddings (Yelp Review dataset).

which follows from the design of position encoding vectors in (4). For all other values of i and n', the term is negligible. Thus,

$$\frac{\partial \frac{\mathbf{y}^{(i)}_{2n'+1}}{\sigma_2^{(i)}}}{\partial \mathbf{B}_{O,t}[:,n]} \approx \begin{cases} c_1^2 \frac{1}{\sigma^3} \frac{1}{D} \mathbf{y}^{(i)} & \text{if } i = \mathcal{T}_t(n), n' = \mathcal{T}_t(n) \\ \text{negligible} & \text{otherwise} \end{cases} \tag{55}$$

Next, as we know from (11), $\mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)} = 0$ for any $d \in [D]$, therefore,

$$\frac{\partial \frac{\mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}}{\sigma_2^{(i)}}}{\partial \mathbf{B}_{O,t}[:,n]} = \frac{1}{\sigma_2^{(i)}} \frac{\partial \mathbf{A}^{\mathrm{T}}_{O,t}[d,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}}{\partial \mathbf{B}_{O,t}[:,n]} = \begin{cases} \frac{\mathbf{y}^{(i)}}{\sigma_2^{(i)}} & \text{if } d = 2\mathcal{T}_t(n)+2 \\ \mathbf{0}_D & \text{otherwise} \end{cases} \tag{56}$$

$$\approx \begin{cases} \frac{\mathbf{y}^{(i)}}{\sigma} & \text{if } d = 2\mathcal{T}_t(n)+2 \\ \mathbf{0}_D & \text{otherwise} \end{cases} \tag{57}$$

Hence, in (35),

$$\frac{\partial \frac{\mathbf{A}^{\mathrm{T}}_{O,t}[2n'+1,:]\mathbf{B}^{\mathrm{T}}_{O,t}\mathbf{y}^{(i)}}{\sigma_2^{(i)}}}{\partial \mathbf{B}_{O,t}[:,n]} = \mathbf{0} \tag{58}$$

Now, from (38) and (55), for $i = \mathcal{T}_t(n)$ and $n' = \mathcal{T}_t(n)$, $\frac{\partial \frac{\mathbf{y}^{(i)}_{2n'+1}}{\sigma_2^{(i)}}}{\partial \mathbf{B}_{O,t}[:,n]} >> \frac{\partial \frac{\mu_2^{(i)}}{\sigma_2^{(i)}}}{\partial \mathbf{B}_{O,t}[:,n]}$ since $\frac{c_1^2}{\sigma^3 D} >> \frac{1}{\sigma D}$ (as observed from our experiments). Therefore, from (55) and (57); (35) can be rewritten as,

$$\frac{\partial \mathbf{p}^{(i)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} \approx \begin{cases} c_1^2 \frac{1}{\sigma^2} \frac{1}{D} \mathbf{y}^{(i)} & \text{if } i = \mathcal{T}_t(n), n' = \mathcal{T}_t(n) \\ \text{negligible} & \text{otherwise} \end{cases} \tag{59}$$

Finally, in (34),

$$\frac{\partial \mathbf{a}^{(0)}_{2n'+1}}{\partial \mathbf{B}_{O,t}[:,n]} \approx \begin{cases} \frac{1}{N+1} c_1^2 \frac{1}{\sigma^2} \frac{1}{D} \mathbf{y}^{(\mathcal{T}_t(n))} & \text{if } i = \mathcal{T}_t(n), n' = \mathcal{T}_t(n) \\ \text{negligible} & \text{otherwise} \end{cases} \tag{60}$$

from which the target token $\mathbf{y}^{(\mathcal{T}_t(n))}$ can be retrieved. Similarly, different tokens can be recovered by leveraging different columns $n \in [r]$ of matrix $\mathbf{B}_{O,t}$ and different encoders $t \in [S-1]$, where r is the rank of LoRA matrices and S is the total number of encoders in the architecture.

D EXTENSION OF MINEGRAD FOR A BATCH OF SAMPLES

Assume there are M samples in the batch. Our attack described in Section 4 essentially recovers an average of M word embeddings in a target position from the gradient with respect to a particular column of LoRA matrix

Table 5: Recovered texts of the two target users from the gradient aggregate with respect to LoRA matrices. The recovered words are highlighted.

Sequence length	User		Fine-tuning sample
16	1	Ground-truth	Will the raise in the minimum wage help you or anyone you know over 19 years
		Recovered	Will the raise in the minimum wage help you or anyone you know over 19 years
	2	Ground-truth	iv been doing alot more excersise and eating approx 1200 cals per day
		Recovered	iv been doing alot more excersise and eating approx 1200 cals per day
32	1	Ground-truth	Will the raise in the minimum wage help you or anyone you know over 19 years
		Recovered	Will the raise in the minimum wage help you or anyone you know over 19 years
	2	Ground-truth	iv been doing alot more excersise and eating approx 1200 cals per day
		Recovered	iv been doing alot more excersise and eating approx 1200 cals per day

B. For simplicity, in the following, we keep the description restricted to a target position. We denote the word embedding (in the target position) for the m^{th} sample in the batch by $\mathbf{x}(m)$ for $m \in [M]$, and the set of these word embeddings is denoted by $\mathcal{M}$ with $|\mathcal{M}| = M$. From the gradient, the attacker can recover,

$$\mathbf{g} \triangleq \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}(m) \tag{61}$$

Now, we design a malicious word embedding layer, where each element is randomly generated from a uniform distribution $\mathcal{U}(-\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}})$ and D is the embedding dimension as defined in Section 4.

Note that the above distribution has a mean 0 and variance $\frac{1}{3D}$. Hence, for any word embedding $\mathbf{v}$,

$$\mathbb{E}[\|\mathbf{v}\|^2] = D \cdot \frac{1}{3D} = \frac{1}{3} \tag{62}$$

Now, for a word embedding $\mathbf{v}$ from the vocabulary where $\mathbf{v} \in \mathcal{M}$, we have

$$\begin{aligned} \mathbb{E}[\mathbf{g}^{\mathrm{T}}\mathbf{v}] &= \frac{1}{M}\mathbb{E}\left[\sum_{m=1}^{M} (\mathbf{x}(m))^{\mathrm{T}}\mathbf{v}\right] \\ &= \frac{1}{M}\mathbb{E}[\|\mathbf{v}\|^2] + \frac{1}{M}\sum_{\mathbf{x}(m)\neq\mathbf{v}} \mathbb{E}[(\mathbf{x}(m))^{\mathrm{T}}\mathbf{v}] \\ &= \frac{1}{3M} + 0 = \frac{1}{3M} \end{aligned} \tag{63}$$

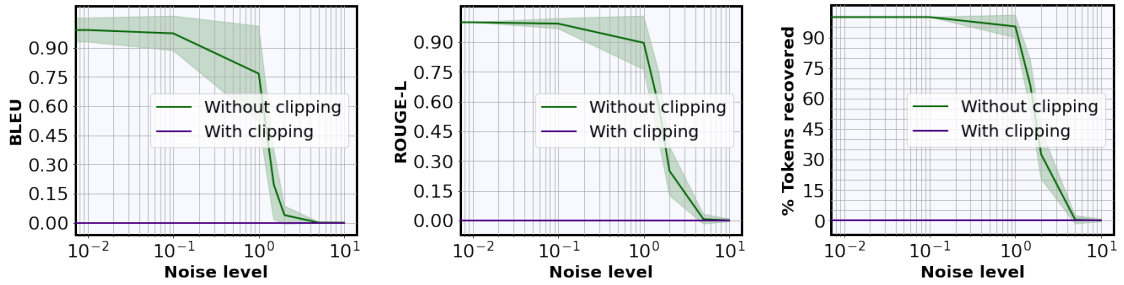
and for a word embedding $\mathbf{v}$, where $\mathbf{v} \notin \mathcal{M}$,

$$\mathbb{E}[\mathbf{g}^{\mathrm{T}}\mathbf{v}] = \frac{1}{M}\mathbb{E}\left[\sum_{m=1}^{M} (\mathbf{x}(m))^{\mathrm{T}}\mathbf{v}\right] = 0 \tag{64}$$

The above properties can be leveraged to find out whether a word embedding from the vocabulary is present in the batch or not. For this, after recovering the average of word embeddings from the gradient, we compute its dot product with the word embeddings in the vocabulary and select the top-M embeddings with highest similarity. We report the results in Table 3, which shows the high success rate of our approach.

Table 6: Average BLEU, ROUGE-L scores and percentage of recovered tokens from gradient aggregate for 100 samples.

Sequence length	Number of users	BLEU	ROUGE-L	% tokens recovered
16	25	0.99	1.0	100
	50	0.99	1.0	100
	75	0.99	1.0	100
	100	0.99	1.0	100
32	25	0.45 ± 0.27	0.78 ± 0.12	80.2 ± 11
	50	0.45 ± 0.27	0.78 ± 0.12	80.2 ± 11
	75	0.45 ± 0.27	0.78 ± 0.12	80.2 ± 11
	100	0.45 ± 0.27	0.78 ± 0.12	80.2 ± 11



(a) BLEU score (b) Rouge-L score (c) Percentage of tokens recovered

Figure 11: Reconstruction performance under gradient noise and clipping (RoBERTa-base model).

E ADDITIONAL EXPERIMENTS

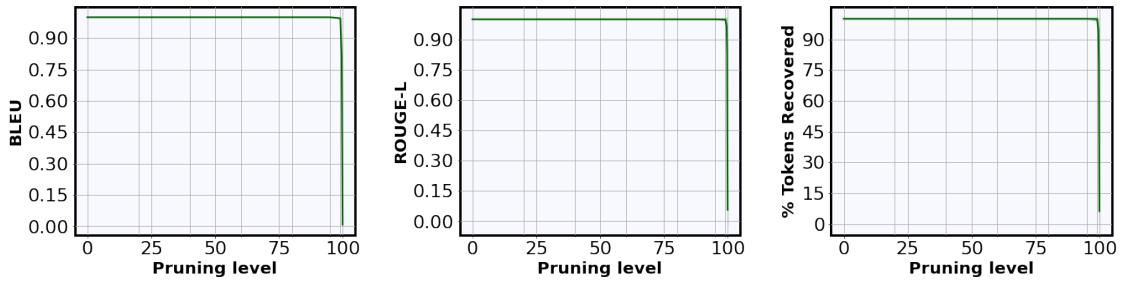
E.1 Modified Word Embedding

In Section 5, the experimental results are shown by using the original word embedding layer from the pretrained models. Since this layer is a part of the pretrained model, an adversarial server has the capability to tamper with this layer as well Feng and Tramèr (2024). The unmodified word embedding layers from pretrained models contain values with a very small standard deviation (close to 0). The server can deploy a malicious word embedding layer with a higher standard deviation, leading to a large difference between different word embeddings. This enables the server to distinguish between different word embeddings and recover corresponding words/sub-words with higher accuracy. In Fig. 10 we demonstrate how an adversarial embedding layer can further enhance the attack performance. For these experiments we consider a rank $r = 4$ and utilize 4 encoders to recover the first 16 tokens. We then demonstrate how the recovery of these 16 tokens is impacted by increasing sequence length. As we observe in Fig. 10, for all architectures, performance is significantly improved by leveraging malicious word embeddings over unmodified word embeddings.

E.2 Robustness Against Defense Mechanisms

In this section, we study the robustness of MineGrad against existing defense mechanisms such as secure aggregation, pruning, gradient noise and clipping. For this, we run the experiments with the RoBERTa-base pretrained model (unmodified word-embedding layer) and Yahoo Answers dataset with rank $r = 4$. We use 4 encoders to recover 16 tokens.

Attack to secure aggregation. We first demonstrate the robustness of MineGrad against secure aggregation protocols. Secure aggregation builds on cryptographic primitives Bonawitz et al. (2017); Bell et al. (2020), where each user sends a masked gradient to the server. Upon receiving the masked local gradients, the server can decode the true gradient aggregate, but cannot access individual local gradients. Recent works Pasquini et al. (2022); Zhao et al. (2024b) show how secure aggregation can be bypassed by leveraging model inconsistency, i.e, the server sends different models to different users. However, prior works do not consider the PEFT setup, which is the focus of our work. We investigate how individual users can be trapped by MineGrad by using model



(a) BLEU score (b) Rouge-L score (c) Percentage of tokens recovered

Figure 12: Reconstruction performance under gradient pruning (RoBERTa-base model).

Table 7: Attack performance for varying c_2 with Yahoo Answers dataset on RoBERTa-base model ($c_1 = 100$).

c_2	3	5	10	20	30
Percentage of tokens recovered	100	100	100	98.6	39.2

inconsistency under secure aggregation. Let us assume for a target encoder t, the server sends malicious LoRA matrices $\mathbf{A}^u_{O,t}$, $\mathbf{B}^u_{O,t}$ designed according to (9) to a target user u, and sends $\mathbf{A}^v_{O,t} = \mathbf{0}$, $\mathbf{B}^v_{O,t} = \mathbf{0}$ to users $v \neq u$. Then user u would produce the target gradient containing information about target tokens whereas users $v \neq u$ would produce zero gradients. To verify this, we target 2 users, without loss of generality, user 1 and user 2. We set the rank as $r = 4$ and leverage encoders $1 - 4$ for user 1, and encoders $5 - 8$ for user 2 to recover 16 tokens. For the pretrained model, we use RoBERTa-base. As we observe in Table 5, even from the gradient aggregate, MineGrad can recover fine-tuning texts of these two users. In Table 6, we report the average scores across 100 different recovered samples from aggregated gradients for two different sequence lengths 16 and 32. We observe that even with the number of total users increasing, the reconstruction accuracy is not impacted.

Table 8: Attack performance for varying c_1 with Yahoo Answers dataset on RoBERTa-base model ($c_2 = 3$).

c_1	5	10	50	100
Percentage of tokens recovered	34.6	98.6	100	100

Robustness against added noise. Next, we investigate how well MineGrad performs under added noise to gradients Abadi et al. (2016). In Fig. 11, we demonstrate the results with added Gaussian noise with varying levels of standard deviation. As we observe, MineGrad successfully retrieves the fine-tuning data with high accuracy even in the presence of noise with standard deviation of up to 1. We further conduct experiments with gradient clipping, where gradients are clipped to 1. As we observe, the attack is unsuccessful under clipping, suggesting clipping could be a better defense.

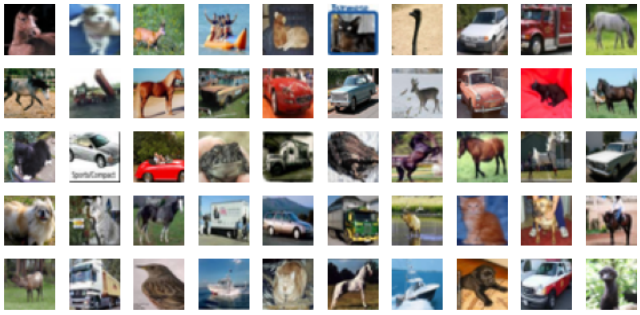
Robustness against pruning. We now demonstrate how MineGrad attack is affected by gradient pruning Lin et al. (2018); Alistarh et al. (2018), where gradient elements with magnitude below a certain percentage are set to 0. As we observe in Fig. 12, even for up to 99% pruning, reconstruction remains unaffected. This can be attributed to the fact that only the target gradients (gradients from LoRA modules of either value weights or output projection weights) are non-zero. Hence, mostly non-target zero gradients are pruned out, while retaining the target gradients.

E.3 Impact of Varying c_1 and c_2

In Table 7, we report the attack performance with varying c_2 while keeping c_1 fixed. As we see, attack success degrades with higher c_2. Moreover, under a fixed c_2, choosing a lower value for c_1 causes an approximation error in Eqs. (6) and (7), resulting in degraded attack performance as evident in Table 8.

Table 9: Mean and standard deviation of the performance metrics across 100 samples with sequence length 16 for different datasets (GPT-2).

Dataset	Metrics		
	ROUGE-L	**BLEU**	**% Tokens Recovered**
AG's News	0.99 ± 0.04	0.96 ± 0.02	99.7 ± 1
DBPedia	0.94 ± 0.15	0.98 ± 0.03	99.2 ± 2
Yahoo Answers	0.996 ± 0.02	0.98 ± 0.08	99.3 ± 2
Yelp Review	0.996 ± 0.02	0.99 ± 0.05	99.6 ± 2
TREC	0.998 ± 0.01	0.994 ± 0.03	99.7 ± 1



(a) Original

(b) Recovered

Figure 13: Reconstruction performance of MineGrad with $r = 4$ on CIFAR-10.

E.4 Decoder-Based Architecture

We now demonstrate the attack performance of MineGrad with GPT-2 Radford et al. (2019) as the pretrained model. Unlike encoder-based architectures like BERT or RoBERTa, GPT-2 follows a unidirectional self-attention mechanism. For classification tasks, either a class token is added to the end of all other tokens, or the last token is used for classification. Same design principles as described in Section 4 can also be utilized for GPT-2. In Table 9, we report the mean and standard deviation across 100 samples for different metrics. As we observe, the tokens are recovered with high success rate.

E.5 Recovered Images from CIFAR-10 Dataset

In Fig. 13, we demonstrate the reconstruction of sample images from CIFAR-10 dataset. As we observe, images are recovered with high fidelity.

Table 10: Recovered tokens for multiple datasets with both modified and unmodified word embeddings (rank $r = 4$). We leverage fine-tuning gradients from 8 encoders to recover 32 tokens.

Dataset	Fine-tuning sentences	
AG's	Ground-truth text	Google gets a bounce, ends its first day up 18 percent Shares of Google leaped $15.34, or 18 percent, to $100.
	Recovered (unmodified)	Google gets a bounce , ends its first day up 18 percent Shares of Google leaped $ 15.34, or 18 percent , to $ 100.
	Recovered (modified)	Google gets a bounce , ends its first day up 18 percent Shares of Google leaped $15 . 34 , or 18 percent , to $100 .
DBPedia	Ground-truth text	Takuma Kanaiwa is a New York City based musician and electrical engineer. His music is known for transcending genres such as free jazz and Japanese folk.
	Recovered (unmodified)	Takuma Kanaiwa is a New York City based musician and electrical engineer . His music is known for transcending genres such as free jazz and Japanese folk .
	Recovered (modified)	Takuma Kanaiwa is a New York City based musician and electrical engineer . His music is known for transcending genres such as free jazz and Japanese folk .
Yahoo	Ground-truth text	In the san francisco bay area, does it make sense to rent or buy ? the prices of rent and the price of buying does not make sense to me
	Recovered (unmodified)	In the san francisco bay area , does it make sense to rent or buy ? the prices of rent and the price of buying does not make sense to me
	Recovered (modified)	In the san francisco bay area , does it make sense to rent or buy ? the prices of rent and the price of buying does not make sense to me
Yelp	Ground-truth text	I try to visit every time I'm in LV because I like the food here but I'm only giving 3 stars because of the controversy around their calorie counts.
	Recovered (unmodified)	I try to visit every time I 'm in LV because I like the food here but I 'm only giving 3 stars because of the controversy around their calorie counts .
	Recovered (modified)	I try to visit every time I'm in LV because I like the food here but I'm only giving 3 stars because of the controversy around their calorie counts .
TREC	Ground-truth text	Who was the author of the book about computer hackers called " The Cuckoo 's Egg : Tracking a Spy Through the Maze of Computer Espionage " ?
	Recovered (unmodified)	Who was the author of the book about computer hackers called " The Cuckoo ' s Egg : Tracking a Spy Through the Maze of Computer Espionage " ?
	Recovered (modified)	Who was the author of the book about computer hackers called " The Cuckoo 's Egg : Tracking a Spy Through the Maze of Computer Espionage " ?

E.6 More Examples of Recovered Sequences

In Table 10, we show reconstruction results for all 32 tokens from sequences with length 32. For this, we leverage gradients from 8 encoders with rank $r = 4$.

E.7 Trainable LayerNorm/Embeddings

Table 11: Average Scores in terms of BLEU and Percentage of tokens recovered (TokRec) over 100 samples with trainable LayerNorm and word-embeddings.

Dataset	RoBERTa-base		RoBERTa-large		BERT-base		BERT-large	
	BLEU	%TokRec	BLEU	%TokRec	BLEU	%TokRec	BLEU	%TokRec
Yahoo	1.0	100	1.0	100	1.0	100	0.81	95
Yelp	0.98	100	0.98	100	0.98	100	0.73	93
DBPedia	0.99	100	0.99	100	1.0	100	0.79	93
TREC	1.0	100	1.0	100	0.98	100	0.82	94
AG's	1.0	100	1.0	100	1.0	100	0.82	94

Our attack remains successful when LayerNorm/word embeddings are trainable. For LayerNorm, (13) guarantees that the target token embeddings reach the next encoder's LoRA modules without distortion as in (8). Hence, target gradients with respect to the LoRA matrix $\mathbf{B}$ remain exactly as in (17) and still reveal the fine-tuning tokens - additional gradients with respect to LayerNorm do not affect this signal. A similar argument also holds for word embeddings. Table 11 demonstrates our results for reconstruction performance, where we report the BLEU Scores, as well as the percentage of tokens recovered for sequence length 16 and $r = 4$.

E.8 Different Optimizers

Table 12: Average scores across 100 samples from Yahoo Answers Dataset with different optimizers (RoBERTa-base).

Seq. Length	Optimizer	Rouge-L	BLEU	% Tokens Recovered
16	SGD	1.00	1.00	100
	Adam	1.00	1.00	100
	AdaGrad	1.00	1.00	100
32	SGD	0.77	0.45	79
	Adam	0.76	0.43	78
	AdaGrad	0.77	0.44	79

Our original framework and derivations consider conventional SGD as the optimizer, as was typically considered also in prior attacks Chu et al. (2023); Fowl et al. (2022, 2023); Petrov et al. (2024); Zhao et al. (2024b), but adaptive optimizers such as Adam or AdaGrad can also be considered as follows. For server side optimizers Wang et al. (2022), note that an adversarial server can bypass such optimizers, so the attack is unchanged. For user-side optimization Wu et al. (2023), since first and second moments of Adam are initialized as zero, after one local step and bias correction, the model update becomes,

$$\hat{\mathbf{g}}_u = \eta \frac{\mathbf{g}_u}{|\mathbf{g}_u| + \epsilon} \tag{65}$$

where $\mathbf{g}_u$ is the gradient of user u, η is the step-size, and ϵ is a small parameter ($\sim 10^{-8}$). Note that $\hat{\mathbf{g}}_u$ preserves the sign pattern of $\mathbf{g}_u$. Since the target gradient in (17) is a scaled token embedding, cosine similarity still recovers the correct embedding and attack remains effective. The same also holds for AdaGrad. Because the attack is one-shot, the adversary only needs the first round. Table 12 reports our results with different optimizers.

E.9 Label Smoothing

We next demonstrate attack performance under label smoothing. In Table 13, we replace each label $i \in [C]$ with,

$$l_i = (1 - \alpha)\delta_i + \frac{\alpha}{C} \tag{66}$$

Table 13: Average scores across 100 samples from Yahoo Answers Dataset with different levels of smoothing (RoBERTa-base).

Seq. Length	α	Rouge-L	BLEU	% Tokens Recovered
16	0.1	1.00	1.00	100
	0.2	1.00	1.00	100
	0.3	1.00	1.00	100
	0.4	1.00	1.00	100
32	0.1	0.76	0.41	78
	0.2	0.74	0.38	77
	0.3	0.74	0.37	76
	0.4	0.73	0.36	76

where δ_i is the Dirac delta, C is the number of classes, and α is the smoothing parameter Szegedy et al. (2016). In (17) one-hot labels result in $z_1 - l_1 = 1$. On the other hand, under smoothing, $l_1 = \alpha/C$, hence $z_1 - l_1 = 1 - \alpha/C$, which stays close to 1 for typical α choices, e.g., 0.1 Szegedy et al. (2016). Thus, embeddings remain almost undistorted, and attack succeeds.

E.10 Dropout Rate

Table 14: Average scores across 100 samples for different dropout rates (RoBERTa-base, sequence length 16).

Dataset	Dropout rate (p)	Rouge-L	BLEU	% Tokens Recovered
Yahoo	0.1	0.91	0.77	94.46
	0.2	0.79	0.52	82.68
	0.4	0.52	0.20	59.35
Yelp	0.1	0.90	0.72	92.58
	0.2	0.76	0.44	80.29
	0.4	0.60	0.21	63.42
DBpedia	0.1	0.90	0.74	92.17
	0.2	0.81	0.54	83.00
	0.4	0.56	0.25	60.50
TREC	0.1	0.91	0.79	95.28
	0.2	0.78	0.48	81.60
	0.4	0.49	0.16	58.03
AG News	0.1	0.91	0.78	94.46
	0.2	0.79	0.50	83.83
	0.4	0.49	0.16	58.63

We evaluate our attack under DropKey Li et al. (2023), which applies dropout to the scaled dot-product scores in the attention layer, by adding a dropout mask before softmax, where each element is sampled as,

$$x = \begin{cases} 0 & \text{with prob. } 1-p \\ -\infty & \text{with prob. } p \end{cases} \tag{67}$$

where the dropout rate is $p \in [0, 1]$. Table 14 reports the reconstruction performance for different dropout rates p. As expected, lower dropout yields higher success.