Uncovering Gradient Inversion Risks in Practical Language Model Training

Xinguo Feng
The University of Queensland
Brisbane, Australia
s.feng@uq.edu.au

Zhongkui Ma
The University of Queensland
Brisbane, Australia
zhongkui.ma@uq.edu.au

Zihan Wang
The University of Queensland
Brisbane, Australia
zihan.wang@uq.edu.au

Eu Joe Chegne
The University of Queensland
Brisbane, Australia
e.chegne@uqconnect.edu.au

Mengyao Ma
The University of Queensland
Brisbane, Australia
mengyao.ma@uq.edu.au

Alsharif Abuadbba
CSIRO's Data61
Sydney, Australia
sharif.abuadbba@data61.csiro.au

Guangdong Bai[*]
The University of Queensland
Brisbane, Australia
g.bai@uq.edu.au

Abstract

The gradient inversion attack has been demonstrated as a significant privacy threat to federated learning (FL), particularly in continuous domains such as vision models. In contrast, it is often considered less effective or highly dependent on impractical training settings when applied to language models, due to the challenges posed by the *discrete* nature of tokens in text data. As a result, its potential privacy threats remain largely underestimated, despite FL being an emerging training method for language models. In this work, we propose a domain-specific gradient inversion attack named GRAB (**gra**dient inversion with *hy**b**rid* optimization). GRAB features two alternating optimization processes to address the challenges caused by practical training settings, including a simultaneous optimization on dropout masks between layers for improved token recovery and a discrete optimization for effective token sequencing. GRAB can recover a significant portion (up to 92.9% recovery rate) of the private training data, outperforming the attack strategy of utilizing discrete optimization with an auxiliary model by notable improvements of up to 28.9% recovery rate in benchmark settings and 48.5% recovery rate in practical settings. GRAB provides a valuable step forward in understanding this privacy threat in the emerging FL training mode of language models.

CCS Concepts

• **Security and privacy**; • **Computing methodologies → Machine learning**;

[*]Corresponding author

Keywords

Federated Learning, Language Models, Gradient Inversion

1 Introduction

The advent of neural network-based language models represents a significant breakthrough in the field of natural language processing (NLP), enabling remarkable performance in various tasks such as named entity recognition [26], sentiment analysis [34], text summarization [37] and text generation [2, 38]. Nevertheless, the requirement of ever-larger models and datasets poses a challenge to model training[1]—the need for large text corpus and data security. Federated learning (FL) [33] steps in this context as a pivotal solution. By facilitating collaborative training over distributed data while ensuring the ownership of individual training data, it enables multiple clients to collaborate to train a model without sharing their raw data with other entities. As a result, the use of FL in language model training has attracted much attention from both academia, *e.g.*, FedFSL [5] and FedBERT [46], and industry, *e.g.*, FedML [11].

Despite the training data being confined locally, recent studies [54, 55] have revealed that an adversary can reconstruct the private training data of a client via an attack known as *gradient inversion.* This attack recovers training data by randomly initializing dummy data and updating it with an optimization process through gradient descent to produce similar model parameter gradients, hence approximating the original training data. The power of gradient inversion is largely due to the efficacy of the optimization techniques, and therefore it shows a high efficacy for the *continuous*

[1]In practice, training a language model typically refers to fine-tuning a published pretrained language model with a local training dataset for a downstream task. Without loss of generality, we use "training" to refer to this practice.

data, such as adjusting the pixel values of the surrogate images to mirror the original images in vision models [13, 24, 30, 51, 54, 55]. However, this strength wanes when it comes to language models, a domain characterized by the *discrete* nature of tokens.

Several efforts have been dedicated to adapting gradient inversion for language models. One line of research [8, 55] explores applying attack strategies originally designed for continuous domains onto token embedding, treating embedding as the "continuous" representation of tokens. This retains the capability of gradient inversion in recovering tokens, but falls short in restoring token order due to the dominance of token embedding over positional embedding. Attack strategies based on discrete optimization [4] seem promising in coping with the token discreteness. However, due to the challenges posed by dropout-induced randomness and long-sequence tokens reordering, they have to strongly rely on controlled experimental settings such as deactivating dropout and providing exact sequence lengths. *The applicability of gradient inversion in practical language model training scenarios* remains largely open. Consequently, the privacy threats posed by gradient inversion against language models have been underestimated.

Our work. We introduce a **gra**dient inversion with a hy**b**rid optimization called Grab, an attack specifically for language model training in practical FL scenarios. Grab's hybrid optimization combines the continuous optimization, which is highly sensitive to initialization [15], with the discrete optimization, which requires the correct recovered tokens to take effect. It schedules them in an iterative paradigm to complement each other, where the output of one serves as an improved initialization for the other in the new iteration. This approach helps avoid local optima and achieves better final recovery. During each iteration, Grab considers the dropout masks between layers and uses gradient descent with dropout mask learning as the continuous optimization to offset the dropout-induced noise, mainly for effective *token recovery*. It employs beam search on the recovered tokens as the discrete optimization for systematic *token sequencing*. This involves the padding token to dynamically adjust the valid sequence lengths, eliminating the need for prior knowledge of the exact sequence lengths in the batch.

During the continuous optimization, Grab employs a dropout mask learning technique to counteract the negative effects caused by dropout [43]. Dropout is a typical regularization technique for overfitting prevention in deep learning. It also introduces noise to the gradient by randomly omitting a subset of neurons in every training round. This poses a major challenge to the traditional continuous gradient inversion, as missing gradients and potential perturbations are caused. Grab's dropout mask learning technique initializes a dummy dropout mask and updates it alongside the dummy data during the optimization process. This measure effectively offsets the dropout-induced noise, which aids the gradient matching and substantially results in an improved token recovery in the presence of dropout.

In the discrete optimization, Grab focuses on reordering the tokens recovered from the continuous optimization. The continuous gradient inversion falls short in restoring token order due to the dominance of token embedding over positional embedding [4, 8, 15, 55]. This limitation becomes particularly pronounced when

no prior knowledge of the exact sequence lengths is provided to the attacker in a practical FL scenario, which introduces numerous potential orderings. To counter this limitation, Grab employs beam search for reordering. Unlike the existing discrete strategy [4] that utilizes an external auxiliary model to guide the search, beam search systematically explores the search space by iterating through each position in the sequence and utilizing the interconnection between adjacent tokens, resulting in the capability of recovering significantly longer sequences. Particularly, our approach involves the padding token in the reordering to dynamically adjust the valid sequence lengths, which requires no prior knowledge of the exact sequence lengths in the batch.

To evaluate the performance of Grab, we conduct both comparative and ablation studies. To evaluate its attack efficacy, we first apply it to three widely-used datasets and compare it with three baseline approaches using various optimization strategies [4, 8, 55] in benchmark settings. The results demonstrate the remarkable recovery ability of Grab. It recovers a large portion (up to 92.9% recovery rate) of the training data, and surpasses the best baseline approach [4] that is based on the discrete optimization with an auxiliary model by notable improvements of up to 28.9% recovery rate. Subsequently, to highlight the practicality of Grab, we introduce the practical settings, including frozen embedding layers and activated dropout. In these settings, Grab remains highly effective, outperforming the best baseline by a remarkable improvement of up to 48.5% recovery rate. This shows its strong practicality. In ablation studies, we evaluate Grab with various training settings, including various model sizes/types (all sizes of BERT [9] and RoBERTa [28] models), various dropout rates [16], more complex task (multi-class classification), large batch sizes (up to 128), and relaxed assumptions. It remains a highly effective and consistent attack performance. Furthermore, we evaluate it against proactive defenses, including gradient noise [49, 55] and gradient pruning [55]. The results reveal its resilience against existing defenses.

Contributions. Our contributions are summarized as follows.

- **A novel gradient inversion in the domain of language models.** We introduce Grab, a novel gradient inversion with hybrid optimization to address domain-specific challenges. It adapts the dropout-mask learning technique to language models for a dropout-aware continuous optimization. It then employs the beam search strategy that involves padding tokens as a systematic discrete optimization, which results in effective data recovery. Grab represents a step forward in understanding the privacy threats posed by the gradient inversion against the emerging FL training mode of language models.

- **Effective attack performance**. Grab demonstrates a high recovery rate of private training data in both benchmark and practical settings. It remains highly effective across various training settings, such as model sizes/types and dropout rates. These findings underscore the threat posed by gradient inversion against language models, refreshing the current understanding of this attack in an emerging domain.

- **Strong practicality and resilience.** To the best of our knowledge, Grab is the first *practical* gradient inversion against language models. It formulates key factors that cause

challenges for inversion attacks, such as activated dropout, frozen embeddings, large batch sizes, and unknown sequence lengths. GRAB achieves high attack efficacy in practical training settings and defense strategies, demonstrating its strong practicality. This establishes a new baseline for gradient inversion attacks for language models, providing a valuable step forward in understanding this previously underestimated threat.

Notations. The notations used throughout this work are introduced. Lower-case letters, such as b, d, i, j, l, m, n, p, s, t, α, and λ, are employed to denote scalar variables or constants. The bold lower-case letters, such as $\boldsymbol{l}$, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{\theta}$, represent vectors or sequences, and bold capitalized letters, such as $\boldsymbol{X}$, $\boldsymbol{B}$, $\boldsymbol{E}$, and $\boldsymbol{\Psi}$, stand for matrices or tensors. We use upper-case calligraphic letters, such as $\mathcal{L}$, $\mathcal{D}$, to denote functions. We use ∇ to denote the gradient operator. For conciseness, the set of consecutive integers from m to n is referred to as $m..n$.

Availability. Our code is publicly available at: https://github.com/UQ-Trust-Lab/GRAB/.

2 Related Work

This section delves into existing attempts of gradient inversion attacks on language models, followed by a discussion on typical training settings and assumptions in existing attacks, and existing defense strategies.

2.1 Existing Attacks

Deep learning models trained in FL context are known to be vulnerable to privacy attacks in the FL process. Besides attacks such as membership inference [31, 36, 44] that aim to infer if a data sample is part of the training data, previous studies show that gradients shared by the clients in FL can be used to recover the private training data with gradient inversion for vision models [13, 24, 30, 51, 54, 55]. These attacks achieve a high recovery rate by utilizing the recovered labels, image priors, and batch normalization statistics. Gradient inversion primarily focuses on continuous data. While image data is continuous (pixel values) and text data is discrete (individual tokens), text data can be mapped to embedding vectors through the embedding layer of language models, such that it can be considered continuous in the embedding space. Recently, more studies show that it is feasible to apply similar methods to recover text data of language models [4, 8, 12, 15, 55].

Solely utilizing continuous optimization. Initially, gradient inversion on language models relies merely on the same techniques for image data, which is highly sensitive to initialization [15]. Zhu *et al.* [55] propose DLG and firstly show that the same attack for vision models can be directly applied to a BERT model [9] and recover part of its training data. Deng *et al.* [8] propose TAG, an improved version of the original attack, which adds the L_1 norm of the gradient distance as a regularization term to improve the token recovery.

Combining with discrete optimization. Despite the partial success of directly utilizing continuous optimization, this approach falls short in restoring token order due to the dominance of token embedding over positional embedding [4, 8, 15, 55]. The employment of discrete techniques enhances the effectiveness of the inversion

Table 1: Comparisons of settings, assumptions, and methods of different approaches.

	Settings		Assumptions		Methods	
	Trainable embeddings	Deactivated dropout	Known Seq. lengths	Known labels	Continuous optimization	Discrete optimization
DLG [55]	●	●	●	●	✓	
TAG [8]	●	●	●	●	✓	
LAMP [4]	●	●	●	●	✓	✓
GRAB	○	○	◐	◐	✓	✓

* ● (or ○) refers to the presence (or absence) of a setting or assumption. ◐ presents the assumption is optional. ✓ presents the involvement of a method. Seq. is the abbreviation for the word "Sequence".

process by reordering the recovered tokens to further minimize the gradient distance. Inspired by the black-box data extraction attack proposed by Carlini *et al.* [7], Gupta *et al.* [15] propose FILM, an attack that utilizes the memorization ability of a GPT-2 model [40] as a probability distribution to reconstruct the private training text sequences with the highest overall probability. This technique is primarily for Causal Language Models (CLMs). Most related to our work for Masked Language Models (MLMs), Balunovic *et al.* [4] propose LAMP, an attack that alternates gradient descent as continuous optimization and random search as discrete optimization with the aid from an external auxiliary language model to recover private training text data of a BERT [9] model. However, its success deeply relies on settings such as deactivated dropout and prior knowledge of exact sequence lengths in the batch. We emphasize that CLMs are easier to attack, as the auto-regressive nature of the model can be utilized as a probability distribution on the next possible tokens, while the MLMs do not have such characteristics.

Active attack. In addition to the passive attacks mentioned above, Fowl *et al.* [12] propose an attack called DECEPTICONS that deploys malicious parameters vector to recover high-fidelity private text from a GPT-2 model [40]. However, this attack is not evaluated against any active detection defense and its stealthiness is unclear.

2.2 Training Settings and Assumptions in Existing Attacks

Most existing attacks are based on certain training settings and assumptions. Table 1 shows the training settings, assumptions, method combinations of GRAB and the related approaches that target MLMs.

2.2.1 Training Settings in Existing Attacks. Common training settings in existing attacks include trainable embedding layers and deactivated dropout.

Trainable embedding layers. DECEPTICONS [12] and FILM [15] follow the observation from Melis *et al.* [35], which shows that the non-zero rows of token embedding gradients can actually reveal the tokens that are within the training batch. Lu *et al.* [30] propose APRIL and demonstrate that with a batch size of 1, the positional embedding layer gradients can be utilized to reveal the full sequence.

This leakage from embedding layers has led to the idea of freezing learnable embedding, resulting in no gradients being generated for the parameters in these layers to defend the attacks [15, 30]. This lightweight defense is shown to be effective against attacks that directly utilize embedding layer gradients, without sacrificing

the utility of the model. It is recommended to be applied before the training starts [15]. However, in current attacks, the embedding layers are still trainable, which might aid the data recovery even if the gradients of these layers are not directly utilized for direct token leakage.

Deactivated dropout. Dropout [16] is a commonly employed countermeasure against overfitting in deep learning. Fundamentally, dropout employs a dropout mask, which controls the participation of neurons during training, subsequently influencing the computed gradients by introducing randomness and perturbing the gradient patterns. This mechanism is the most influential factor in the effectiveness of continuous optimization in a practical FL scenario [43]. Upon further inspection of the implementation of existing attacks [4, 8, 55], it is common that dropout is deactivated during training. The attacker can facilitate gradient descent for token recovery without considering the dropout-induced noise, which is usually not the case in a practical FL scenario.

Recent studies [49] suggest that dropout can be used as a defense mechanism to mitigate gradient inversion attacks. Utilizing active dropout masks introduces a challenge for the attacker attempting to invert embeddings through gradients, primarily because the attacker lacks insight into the specific dropout mask realization of the victim client. However, in the study by Scheliga *et al.* [43], a technique is introduced that incorporates dropout mask learning within the gradient inversion. This method, when applied to a Vision Transformer (ViT) model [10], demonstrates substantial attack effectiveness, even when faced with elevated dropout rates, as the learned dropout mask further aids the optimization to counteract missing gradients and the dropout-induced noise.

2.2.2 Assumptions in Existing Attacks. Most existing attacks are based on certain assumptions on the knowledge of the adversary [4, 8, 55]. For text data, the usual assumptions are the labels and the exact sequence lengths in the batch are known to the attacker [4]. For the labels, existing attacks argue that they can apply the method introduced by Zhao *et al.* [54] to identify the labels present in the batch, followed by enumerating all combinations. However, this approach only identifies the presence of specific labels but fails to ascertain the quantity of each label or the precise label associated with each sequence in the batch. For the sequence lengths, existing attacks also justify that they can enumerate all length combinations in the batch. Both assumptions become impractical as the batch size grows, as the number of possible combinations is enormous.

2.3 Defenses against Gradient Inversion Attacks

Gradient inversion relies on minimizing the gradient distance between the dummy gradients and the shared gradients to approximate the original training data. To mitigate the effectiveness of such attacks, existing defenses primarily focus on perturbing gradient patterns or injecting noise into the gradients. Specifically, two main approaches are employed.

Freezing embedding layers and activating dropout. As discussed above in Section 2.2.1, freezing embedding layers and activating dropout can serve as lightweight defense mechanisms. Freezing embedding layers can prevent direct gradient information leakage of tokens involved in the batch, while activated dropout can perturb the gradients by omitting a subset of neurons.

Pruning or perturbing gradients. Pruning or perturbing gradients is another defense strategy against gradient inversion attacks by slightly modifying the original gradients. Zhu *et al.* [55] propose a defense that prunes the gradients of small magnitudes to perturb the gradient patterns. Additionally, another approach suggests directly injecting Gaussian or Laplacian noise into the gradient to further perturb them [49, 55].

3 Preliminaries

In this section, we introduce the necessary preliminary knowledge to facilitate a comprehensive understanding of GRAB.

3.1 Language Modeling

Language modeling has become a major focus in NLP due to the development of neural networks. This paper focuses on the MLM, which is a key framework in language modeling. BERT [9], a typical MLM, is a transformer encoder [47] that contains millions of parameters pre-trained on a large corpus of text. It is able to generate contextual encoding representation of text after pre-training. Generally, a custom layer is added to the encoder and is fine-tuned for a downstream task, such as sentiment analysis [34].

In this work, we investigate a BERT model augmented with a fully connected layer for a sequence classification task. The input is a data batch $B = (X, y)$, where $X \in \mathbb{R}^{b \times n}$ is a batch of b sequences with n tokens and $y \in \mathbb{Z}^b$ is a batch of the correct labels for the sequences. Initially, X is passed through the token embedding layer to obtain the embedding $E \in \mathbb{R}^{b \times n \times d}$, where d is the embedding dimension. Subsequently, E is fed through the remaining layers to generate the logit outputs. The loss function $\mathcal{L}$ computes the loss value based on the softmax outputs of the logits and the labels. Finally, the gradients $\nabla_{\theta^t} \mathcal{L}(B)$ of parameters θ^t at iteration t are computed by backpropagation and are used to compute the updated parameters θ^{t+1}. By repeating this process, the loss is minimized and the accuracy of the model is improved.

3.2 Federated Learning

Federated learning [33] is a machine learning framework that allows multiple clients to collaborate in training a model without having to share their individual private training data, but only the training information such as gradients, with the coordination of a server.

In this work, we consider the FedSGD [33] setting. In the t-th training iteration, the i-th client computes the gradients $\nabla_{\theta^t} \mathcal{L}(B_i^*)$ of parameters θ^t by the loss function $\mathcal{L}$ and local data batch B_i^*. Then, all clients send their gradients to the server. The server computes the updated parameters θ^{t+1} by aggregating all clients' gradients and subtracting it from the current parameters θ^t,

$$\theta^{t+1} = \theta^t - \frac{\alpha}{m} \sum_{i=1}^{m} \nabla_{\theta^t} \mathcal{L}(B_i^*),$$

where m is the number of clients and α is the learning rate. The server then sends the updated parameters to each client for updating their local models.

3.3 Gradient Inversion Attacks

The gradient inversion attack [4, 8, 12, 13, 15, 54, 55] is a technique used to recover the training data of a client from the transmitted
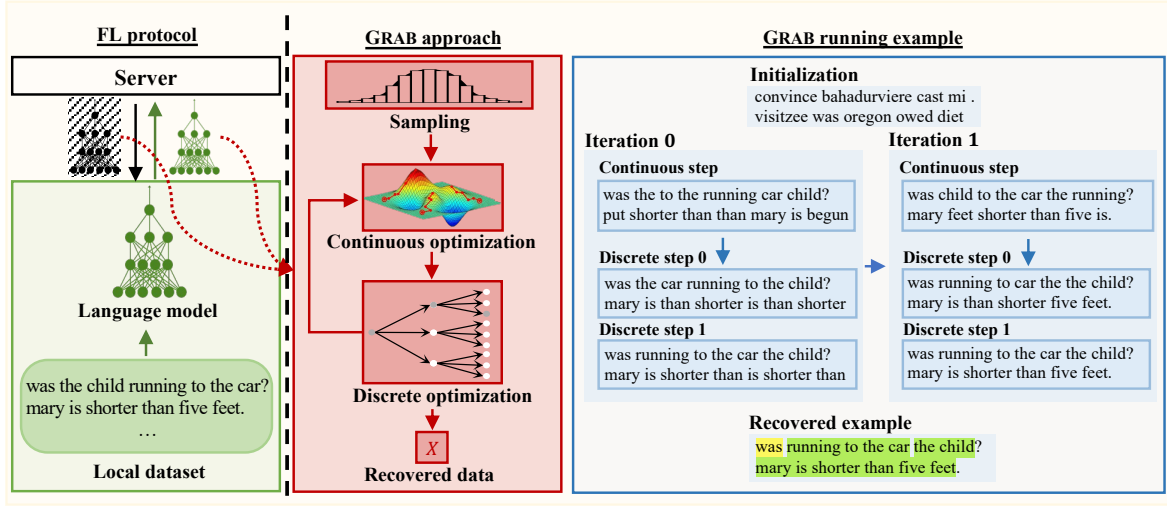
Figure 1: Overview of GRAB.

gradient information in FL. It is assumed that the adversary is able to eavesdrop on the communication between the server and the victim client, and has full knowledge of the model architecture and parameters. To recover the input data batch B_i^*, an alternative data batch B_i is randomly initialized and optimized by solving the following optimization problem,

$$\underset{B_i}{\arg\min}\ \mathcal{D}(\nabla_{\theta^t} \mathcal{L}(B_i),\ \nabla_{\theta^t} \mathcal{L}(B_i^*)),$$

where $\mathcal{D}$ is a distance measure function. For text data, B_i represents the embedding of the data batch. It is further mapped back to discrete tokens by comparing it with the token embeddings of the language model.

3.4 Droupout Mask Learning

To counter the perturbation introduced by dropout, we adopt the dropout mask learning technique proposed by Scheliga *et al.* [43]. Dropout mask learning aims to learn the dropout mask realization of the victim client's model in that training round to reduce the perturbation effect. Specifically, the attacker initializes the dropout masks between the layers and optimizes them simultaneously in the continuous optimization process. The optimization can be further expressed as

$$\underset{B_i, \Psi_p}{\arg\min}\ \mathcal{D}(\nabla_{\theta^t} \mathcal{L}_{\Psi_p}(B_i),\ \nabla_{\theta^t} \mathcal{L}_{\Psi_p^*}(B_i^*)),$$

where Ψ_p is the dropout mask realization being optimized.

3.5 Beam Search

To reorder the tokens recovered by continuous optimization, we adopt beam search as a discrete optimization technique. Beam search is a greedy-like discrete optimization, where greedy search only focuses on the current best solution, beam search keeps track of the *top-k* solutions in a breadth-first-search manner along the process and selects the best solution at the end [42].

4 Approach

In this section, we give a detailed description of our proposed gradient inversion with hybrid optimization, which iteratively alternates continuous and discrete optimization to recover the private training data. We first discuss the threat model in Section 4.1, and provide an overview of our approach in Section 4.2. The continuous and discrete optimizations are discussed in Section 4.3 and Section 4.4.

4.1 Threat Model

We consider an honest-but-curious adversary that controls the server or eavesdrops on the communication between the victim client and the server. It has full knowledge of the language model architecture and parameters, including the dropout rate and dropout layer positions, as they are part of the model architecture. The objective of the adversary is to recover the private training data of the victim client by utilizing the victim client's shared gradient updates. The threat model aligns with a real-world training scenario, where the only communication between the client and the server is the gradient updates. The adversary can only utilize the shared gradient updates to attempt the recovery without extra knowledge of the data, *e.g.,* ground truth labels and sequence lengths.

Existing attacks [4, 8, 15, 55] often adopt a threat model that is with extra knowledge on the ground truth labels and the individual sequence lengths in the data batch, as discussed in Section 2.2.2. We highlight that the threat model defined in our work does not assume such knowledge that gives advantages to the recovery, which aligns with a more practical training scenario.

4.2 Overview of GRAB

Figure 1 shows the overview of GRAB. We highlight that continuous and discrete optimizations collaborate in our approach. In continuous optimization, we enhance the established gradient inversion attack framework [4] by incorporating dropout mask learning [16]. Then, we employ beam search [42] as a search strategy to reorder the recovered tokens in discrete optimization.

Algorithm 1 GRAB- Main Algorithm

Input: θ^t - model parameters at t-th training iteration;
$\nabla_{\theta^t} \mathcal{L}(E^*, y^*)$ - original gradients at t-th training iteration;
Output: X - optimized token solution;

1: $\Psi \leftarrow \text{Bernoulli}(p)$;
2: $E_d, y \leftarrow \text{initDummyData}(n_e, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
3: **for** $i \in 0..n_h$ **do**
4: $\quad E_c, y \leftarrow \text{contOpt}(E_d, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*), \Psi)$;
5: $\quad X_c \leftarrow \text{tokenize}(E_c)$;
6: $\quad X_d \leftarrow \text{discOpt}(X_c, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
7: $\quad E_d \leftarrow \text{embed}(X_d)$;
8: **end for**
9: $X \leftarrow X_d$;
10: **if** $\mathcal{L}_r(E_c, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*)) > \mathcal{L}_r(E_d, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$ **then**
11: $\quad X \leftarrow X_c$;
12: **end if**
13: **return** X

Recovery loss function. The recovery loss function is a distance function that measures the distance between the dummy gradients and the original gradients, such as the L_2 norm distance [55], the combination of L_2 and L_1 norm distance [8], and the cosine distance [13]. We empirically choose the combination of L_2 and L_1 norm as the distance measurement in this work. The recovery loss is defined as

$$\mathcal{L}_r(E, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*)) = \sum_{i=1}^{l} \left\| \nabla_{\theta^t_i} \mathcal{L}(E, y) - \nabla_{\theta^t_i} \mathcal{L}(E^*, y^*) \right\|_2$$
$$+ \alpha_{L_1} \left\| \nabla_{\theta^t_i} \mathcal{L}(E, y) - \nabla_{\theta^t_i} \mathcal{L}(E^*, y^*) \right\|_1, \tag{1}$$

where l is the number of layers in the language model, and α_{L_1} is the pre-defined regularization weighting factor for the L_1 norm.

Overview of our approach. The overall workflow of GRAB is listed in Algorithm 1. First, the attacker initializes the dummy dropout mask Ψ by Bernoulli distribution with the dropout rate p (line 1). Next, it initializes a dummy embedding tensor $E \in \mathbb{R}^{b \times n \times d}$ for approximating the original embedding E^* and the dummy label tensor $y \in \mathbb{R}^{b \times c}$. We follow the initialization approach from LAMP [4] to initialize n_e random embedding candidates and select the one E_d that yields the minimum recovery loss by Formula **??** (line 2). After the initialization, it iteratively executes the continuous and discrete optimizations n_h times (lines 3–8). Specifically, it employs the continuous optimization contOPT first (line 4) and then tokenizes the embedding tensor E_c produced to obtain the recovered tokens X_c (line 5). Subsequently, the discrete optimization discOPT takes X_c as input and outputs the optimized tokens X_d (line 6). X_d is embedded to obtain E_d and will be used in the next iteration (line 7). After iterating all hybrid optimization rounds, we compare the results of continuous and discrete optimizations (lines 9–12) and output the corresponding recovered tokens.

A running example of GRAB in our experiment is shown in Figure 1. By iteratively alternating the two optimization processes, all individual tokens are recovered and most of the tokens are rearranged in the correct order.

Algorithm 2 Continuous Optimization - contOpt(·)

Input: E - dummy embedding;
y - dummy labels;
θ^t - model parameters at t-th training iteration;
$\nabla_{\theta^t} \mathcal{L}(E^*, y^*)$ - original gradients at t-th training iteration;
Ψ - dropout mask of attacker model;
Output: E - optimized dummy embedding;

1: $E \leftarrow \underset{E' \in \text{permute}(E, n_p)}{\arg \min} \mathcal{L}_r(E', y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
2: **for** $i \in 0..n_c$ **do**
3: $\quad L_r \leftarrow \mathcal{L}_r(E, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
4: $\quad E \leftarrow E - \lambda \cdot \frac{\partial L_r}{\partial E}$;
5: $\quad y \leftarrow y - \lambda \cdot \frac{\partial L_r}{\partial y}$;
6: $\quad \Psi \leftarrow \min(1, \max(0, \Psi - \lambda \cdot \frac{\partial L_r}{\partial \Psi}))$;
7: **end for**
8: **return** E

Algorithm 3 Discrete Optimization - discOpt(·)

Input: X - dummy sequences of tokens;
y - dummy labels;
θ^t - model parameters at t-th training iteration;
$\nabla_{\theta^t} \mathcal{L}(E^*, y^*)$ - original gradients at t-th training iteration;
Output: X - optimized dummy sequences of tokens;

1: $\{X_{1..n_b}\} \leftarrow \underset{X' \in \text{permute}(X, n_p)}{\arg \min} \left(\mathcal{L}_r(X', y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*)), n_b \right)$;
2: $\{T_{1..n_b}\} \leftarrow \text{getTokens}(\{X_{1..n_b}\})$
3: **for** $i \in 0..n_d$ **do**
4: $\quad$ **for** $j \in 0..n_b$ **do**
5: $\quad\quad X_j \leftarrow \underset{X'_j \in \text{beamSearch}(X_j, \{T_j\})}{\arg \min} \mathcal{L}_r(X'_j, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
6: $\quad$ **end for**
7: **end for**
8: $X \leftarrow \underset{X \in \{X_{1..n_b}\}}{\arg \min} \mathcal{L}_r(X, y, \theta^t, \nabla_{\theta^t} \mathcal{L}(E^*, y^*))$;
9: **return** X

4.3 Continuous Optimization

The main process of our continuous optimization is shown in Algorithm 2. First, the adversary permutes the dummy embedding to get n_p candidates and selects the one with the minimum recovery loss (line 1). Subsequently, it updates the dummy embedding, the dummy labels, and the dropout mask with gradient descent based on the recovery loss function and iterates n_c times (lines 2–7). Note that our approach considers the frozen embedding layers and the activated dropout (line 6) in practical settings.

Resolving frozen embedding layers. Unlike existing attacks [12, 15, 35], our approach does not particularly rely on the gradients of these layers. We run gradient descent from randomly initialized dummy embedding and exclude these layers.

Resolving activated dropout. When dropout is activated, the language model of the attacker also employs dropout [43] when computing the recovery loss. As described in the threat model in Section 4.1, the attacker has knowledge of the model architecture, meaning that the dropout rate and the positions of the dropout

layers are known to the attacker. When the adversary launches the attack, it activates dropout and initializes the dropout mask Ψ.[2] It then optimizes the dropout mask Ψ.

After computing the recovery loss (line 3), we compute the gradients for the dummy embedding, the dummy labels, and the dropout mask. We optimize them simultaneously in an iterative manner (lines 4–6). The values of the dropout masks are clamped between 0 and 1 (line 6).

4.4 Discrete Optimization

Algorithm 3 presents the main procedure of our discrete optimization method. The output of continuous optimization is the optimized embedding tensor E_c. It is transformed into individual tokens X_c by comparing with the token embeddings of the language model. Considering that the recovered tokens are usually in an incorrect order [4, 8, 55], we propose to utilize beam search [42] to reorder the recovered tokens due to its systematic search ability. Other alternatives such as Nucleus Sampling [17] for generative models may not be directly applicable, as the victim model is a classification model.

At the beginning of the process, permutations and selections are employed on X (line 1), which is similar to the operations in continuous optimization. Note that we keep the first n_b permutations $\{X_{1..n_b}\}$ with the minimum recovery loss values as our initialization of the beams. Then n_d times beam search are employed for each X_j with the corresponding token set $\{T_j\}$ to update X_j (lines 3–7). Specifically, for each sequence in each beam, we go over each of the positions and substitute it with each of the recovered tokens for that sequence. For each position of the sequence, the initial n_b candidates with the lowest recovery loss values are retained as the new beams. It then progresses to the subsequent position and the process is reiterated. This is similar to a greedy search approach, with the distinction being the retention of n_b candidates as opposed to just the current optimal one.

When the batch size is larger than 1, the shorter sequences in the batch are padded with the padding token to the longest length. This causes extra challenges in recovering the shorter sequences in terms of recovery and efficiency. To overcome these difficulties, we include the padding token in the token set $\{T_j\}$ in the reordering. When the padding token is selected at a position, it injects a crucial ending signal, which helps prevent shorter sequences from appending unnecessary tokens to improve the recovery.

5 Experimental Evaluation

This section evaluates Grab from the following four key aspects.
Benchmark performance. Following the experimental setup in [4], we benchmark the performance of Grab against all baseline approaches [4, 8, 55] in benchmark settings.
Practical performance. We assess the practicality of all approaches by evaluating their performance in practical settings, which are more constrained.
Ablation studies. We conduct ablation studies to evaluate the performance of Grab under different circumstances, *i.e.*, various

---

[2]In the case of unknown dropout rates and unknown positions, the attacker can use the empirically well-established convention of dropout rates (*e.g.*, 0.1) and dropout layer positions (*e.g.*, after the transformer attention/output layer) for initialization.

Table 2: Text recoverability across different approaches.

Reference	was the child running to the car?	mary is shorter than five feet.
DLG	"?.. child is was	broken.meries youth area finals
TAG	are derrick the ? mary child toward	feet than .eth mary is shorter
LAMP$_{cos}$	mary is trailer? inventor the car ?	shorter than mary is ? mary
LAMP$_{L_1 L_2}$	to child was running the car ?	feetlle mary shorter is feet .
Grab	was running to the car the child ?	mary is shorter than five feet .

model sizes/types and various dropout rates in practical settings. Additionally, to further evaluate the practicality of Grab in the absence of the assumptions discussed in Section 2.2.2, we explore instances where we relax one or both of these premises.
Resilience against proactive defenses. To evaluate the resilience of Grab, we compare the performance of all approaches against proactive defenses in practical settings, *i.e.*, gradient noise [49, 55] and gradient pruning [55].

Table 2 shows a recovery example of all approaches on the CoLA dataset with a batch size of 2 on the BERT$_{base}$ model in practical settings, where correctly recovered unigrams and longer sequences are highlighted in yellow and green. Note that Grab recovers significantly more bigrams and longer sequences.

5.1 Experimental Settings

In this section, we introduce the experimental settings, covering training settings, datasets, language models, baselines, hyperparameters, FL setup, metrics, and result statistics. Further details on additional experimental settings, including empirical experiment implementation, model implementation in practical settings, and detailed metric calculation are provided in Appendix B.
Training settings. We evaluate Grab in two different training settings, namely *benchmark* and *practical settings*. In benchmark settings, following existing attacks [4, 8, 55], the embedding layers are trainable and observable, but the attacker does not directly extract tokens from the gradients of these layers. Additionally, the dropout layers are deactivated during training. In practical settings, the two lightweight defense mechanisms discussed in Section 2.3 are applied. In particular, the embedding layers are frozen, such that there are no gradient updates to be observed for these layers. Additionally, dropout is activated during training to prevent overfitting. In our comparative study in Section 5.2 and Section 5.3, both settings adopt the known label assumption and the known sequence lengths assumption for comparison with baseline methods. In the ablation study in Section 5.4.3, we relax these assumptions and show that Grab does not need to operate under such assumptions.
Datasets. Our main experiments are conducted on three widely used binary sequence classification datasets, CoLA [48], SST-2 [45], and Rotten Tomatoes [39], with typical sequence lengths of 5–9 words, 3–13 words, and 14–27 words, respectively. The selected datasets are representative in terms of sequence lengths, as other commonly used datasets such as Tweet Sentiment Extraction [21] (5–25 words) and Yahoo Answers Topics [22] (5–15 words) have similar sequence lengths. To ensure a fair comparison, for each

dataset, we randomly select 64 samples with the consistent random seed from the implementation of LAMP [4]. Later in Section 5.4.4, we evaluate GRAB with a 10-class classification task with the Yahoo Answers Topics dataset.

Language models. The main experiments (in Section 5.2 and 5.3) are conducted on the most extensively used language model, $BERT_{base}$ model [9], with 12 layers, 12 heads, and 110 million parameters. Later in Section 5.4.1, models with various sizes/types are discussed and evaluated.

Baselines. Three existing approaches, DLG [55], TAG [8], and LAMP [4] are used as the baselines. In particular, two variants, $LAMP_{cos}$ (using cosine distance) and $LAMP_{L_2+L_1}$ (using combined L_2 and L_1 distance) of LAMP are used. Implementations of all baseline approaches are taken from the released source code of LAMP. We adopt their by-default hyperparameter settings in the evaluation. The benchmark settings we define are the same as those used in the baseline methods. In practical settings, our configurations are designed to mimic realistic conditions, such as frozen embedding and activated dropout. Since the hyperparameters (*e.g.,* number of iterations) are not related to these specific configurations, we keep the same hyperparameter settings.

Experimental parameters. We empirically set the number of embedding candidates $n_e = 2,000$ and the number of permutations $n_p = 2,000$ to obtain an initial solution, considering the sensitivity of continuous optimization. For the optimizations, we set the iteration number of hybrid optimization $n_h = 5$, iteration number of continuous optimization $n_c = 2,000$, iteration number of discrete optimization $n_d = 5$, and number of beams $n_b = 4$, reaching a total of $10,000$ continuous optimization steps and 25 discrete optimization steps, which empirically achieve a reasonable trade-off considering efficiency and efficacy. Additionally, we set the L_1 weight factor $\alpha_{L_1} = 0.01$, and learning rate $\lambda = 0.01$ with a learning rate decay factor $\gamma = 0.89$ for every 50 steps, which aligns with $LAMP_{L_2+L_1}$. We use the AdamW optimizer [29]. Early stopping occurs when the discrete optimization does not improve the output of the continuous optimization. All these parameters have been kept consistent in both baseline settings and practical settings for a fair comparison with baseline methods.

We note that smaller iteration numbers for continuous optimization are used in the default settings of DLG (2,500), TAG (2,500), and LAMP (2,000). To ensure a fair comparison, we standardize the iteration count to 10,000 across all approaches in a trial experiment. Our approach consistently shows improvement, whereas other approaches early fall into a local optimum at around their default number of iterations. For this reason, we keep the default number of iterations for all baselines.

FL setup. Given the conceptual similarity between the setups with multiple clients and a single client-server pair in FL [15], this work utilizes the configuration of one server and one client (victim) for simplicity, which is commonly adopted in prior studies [4, 8, 15, 55]. Specifically, a scenario with m clients, each possessing b samples, is functionally equivalent to training a model with a batch size of $m \times b$ [15]. We empirically demonstrate this equivalence in Table 8 in Appendix A.1, where the performance of our approach under both setups is consistent.

Metrics. The efficacy of the attacks is evaluated by the recovery rate of the original data batch. We use the ROUGE score [27] as a

metric, aligning with existing work [4, 15]. The ROUGE score measures the similarity between the original sentences and the recovered sentences. Specifically, different overlapping grams are used, including unigram, bigram, and longest-matching subsequences, denoted by ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L). The aggregated F-score is used to compute the ROUGE score. When computing the ROUGE score, special tokens without meaningful semantic (start token [CLS], end token [SEP], and padding token [PAD]) are excluded. In ablation studies and the proactive defenses evaluation, we use Matthews Correlation Coefficient (MCC) [32] to measure the utility of the models.

During our benchmarking in Section 5.2 and Section 5.3, the original training samples (*i.e.,* the ground truth) are used to evaluate GRAB's capability of recovering text that closely matches the actual training data, in terms of tokens and token order. However, in real-world scenarios, the original data is unknown to the attacker. The attacker can only infer that the recovered data contains matched tokens and token order to an extent based on the exhibited close gradient distances.

Result statistics. As random factors such as the initialization of dropout masks and dummy embedding can impact the final recovery results, we report the aggregated results of three random runs for a more reliable evaluation. Specifically, mean values and standard deviations of R-1, R-2, and R-L with different batch sizes in powers of 2 (up to 32) are presented to evaluate the performance of different approaches.
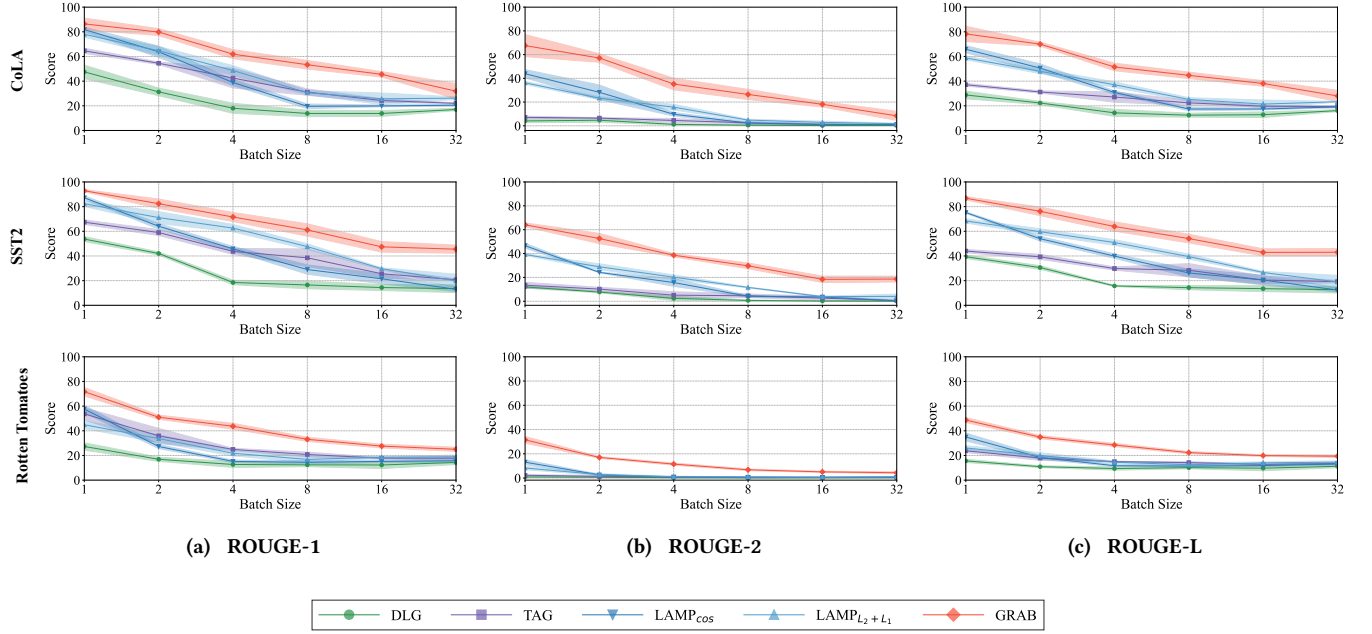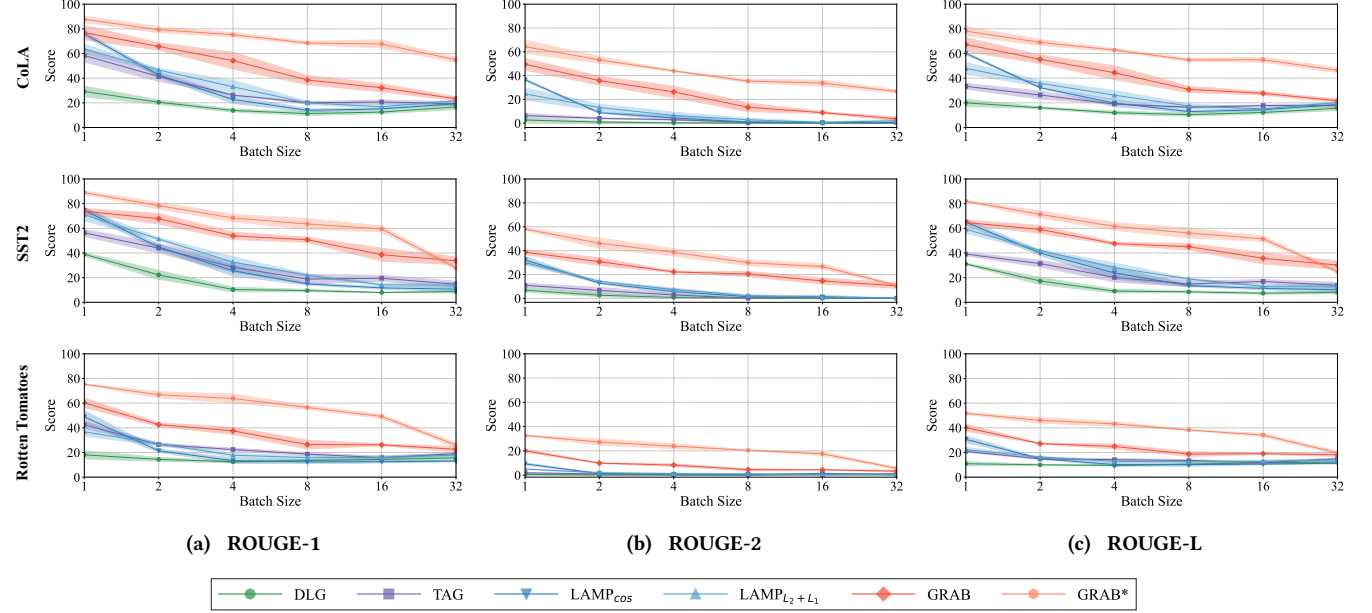
5.2 Benchmark Evaluation

We benchmark the performance of GRAB with all baseline approaches with batch sizes from 1 to 32 in benchmark settings. The results are shown in Figure 2.

GRAB recovers a significant portion of the data, with up to 92.9% recovery rate, demonstrating its remarkable recovery ability. Despite some occasional larger standard deviations, it outperforms all baseline approaches across all datasets and batch sizes, with improvements of 13.2% in R-1, 15.5% in R-2, and 13.6% in R-L on average, and a maximum improvement of 28.9% compared to the best baseline approach. The results indicate the superior recovery ability of GRAB in benchmark settings. Additionally, it also showcases its better generalizability with influential random factors (*e.g.,* dummy embedding initialization). We also observe a general trend where performance decreases as batch size increases. This is due to a larger batch size resulting in more averaged gradients of the sequences, which poses challenges to the recovery process.

GRAB not only recovers more individual tokens (R-1) but also rearranges more tokens in a meaningful and sensible order (R-2 and R-L), making the recovery leak more important semantics. In benchmark settings, this notable improvement is mainly achieved by the effective beam search technique, compared to no discrete optimization at all (DLG and TAG) or only random search (LAMP).

5.3 Practical Evaluation

We evaluate the practicality of GRAB and the baseline approaches by shifting from benchmark settings to practical settings. This involves applying frozen embedding layers and activated dropout, which not only brings inherent challenges to cope with but also better

Figure 2: Performance in benchmark settings on the BERT$_{base}$ model.

Figure 3: Performance in practical settings on the BERT$_{base}$ model. GRAB* is the variant with dropout mask learning.

represents real-world scenarios. We run all approaches with batch sizes ranging from 1 to 32. Note that two variants of GRAB are assessed, *i.e.,* with or without dropout mask learning. The results are shown in Figure 3.

We observe that the performance of all baseline approaches and GRAB without dropout mask learning decreases in practical settings compared to the performance in benchmark settings. This

performance drop affirms that the frozen embedding layers and activated dropout indeed pose difficulty to the gradient matching. Despite this, GRAB without dropout mask learning still recovers a large portion of the data with up to 76.9% recovery rate. Our recovery is more coherent in comparison to existing approaches. It outperforms the baseline approaches in most cases, with improvements of 13.2% in R-1, 10.5% in R-2, and 11.8% in R-L on average and

Table 3: Performance in practical settings on various BERT and RoBERTa model sizes.

	BERT_{tiny}			BERT_{base}			BERT_{large}		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
DLG	3.9 ± 0.5	0.0 ± 0.0	3.6 ± 0.5	29.2 ± 4.3	2.7 ± 1.5	20.0 ± 2.9	5.6 ± 1.3	0.8 ± 0.6	4.9 ± 0.9
TAG	14.2 ± 3.1	0.9 ± 0.9	12.1 ± 2.6	58.4 ± 5.3	6.4 ± 1.7	33.4 ± 2.3	11.9 ± 0.4	1.0 ± 0.6	8.4 ± 0.7
LAMP_{cos}	84.1 ± 0.5	42.9 ± 1.1	66.1 ± 0.9	75.9 ± 1.6	36.4 ± 0.9	60.1 ± 1.0	17.3 ± 1.3	4.2 ± 0.9	15.2 ± 0.5
$\text{LAMP}_{L_2+L_1}$	79.8 ± 1.2	34.2 ± 2.3	60.2 ± 2.3	63.8 ± 3.8	24.6 ± 5.1	48.0 ± 4.7	1.3 ± 0.7	0.1 ± 0.1	1.3 ± 0.7
GRAB	88.3 ± 2.0	60.3 ± 4.5	79.8 ± 2.1	76.9 ± 5.9	49.5 ± 4.9	67.3 ± 5.6	14.3 ± 6.1	4.4 ± 3.1	12.6 ± 5.1
GRAB *	**90.2 ± 1.5**	**60.6 ± 0.5**	**80.0 ± 1.3**	**87.7 ± 2.5**	**64.3 ± 5.6**	**78.2 ± 3.6**	**70.2 ± 4.3**	**32.2 ± 4.3**	**55.0 ± 2.8**
	RoBERTa_{tiny}			RoBERTa_{base}			RoBERTa_{large}		
GRAB	71.2 ± 1.1	37.1 ± 1.1	62.7 ± 0.7	43.6 ± 3.2	22.5 ± 2.3	38.7 ± 3.5	1.1 ± 0.1	0.0 ± 0.0	1.1 ± 0.1
GRAB *	**82.8 ± 1.5**	**58.8 ± 3.0**	**77.2 ± 1.6**	**71.6 ± 6.3**	**36.8 ± 8.5**	**60.1 ± 5.9**	**31.7 ± 8.1**	**11.0 ± 4.5**	**26.9 ± 7.3**

* with dropout mask learning.

Table 4: Performance of tuned hyperparameters.

	BERT_{large}			RoBERTa_{large}		
	R-1	R-2	R-L	R-1	R-2	R-L
DLG	51.4 ± 5.0	7.5 ± 2.4	33.1 ± 2.8	-	-	-
TAG	61.1 ± 2.6	8.0 ± 1.6	37.6 ± 2.1	-	-	-
LAMP_{cos}	76.4 ± 2.1	34.3 ± 3.0	60.2 ± 2.2	-	-	-
$\text{LAMP}_{L_2+L_1}$	70.9 ± 3.4	25.1 ± 3.4	52.9 ± 2.6	-	-	-
GRAB	64.6 ± 2.5	29.3 ± 1.7	52.5 ± 2.3	7.1 ± 2.1	1.0 ± 0.8	6.0 ± 1.4
GRAB *	**84.8 ± 1.8**	**50.2 ± 4.9**	**69.3 ± 3.5**	**74.1 ± 1.0**	**33.2 ± 3.2**	**59.2 ± 0.9**

* with dropout mask learning.

28.7% maximum compared to the best baseline approach with some occasional larger standard deviations. Such results underscore the efficacy and generalizability of our approach in practical settings even in the absence of the dropout mask learning strategy.

GRAB with the dropout mask learning technique demonstrates an even more remarkable recovery ability in practical settings. It remains consistent performance in most cases compared to performance in benchmark settings, affirming that the dropout mask learning technique can indeed offset the dropout-induced noise. GRAB with dropout mask learning outperforms all baseline approaches on all datasets with all batch sizes, with remarkable improvements of 31.8% in R-1, 25.5% in R-2, and 26.8% in R-L on average and 48.5% maximum, demonstrating significant superiority and reaffirming the effectiveness of the dropout mask learning technique.

5.4 Ablation Studies

We evaluate the effectiveness of GRAB under different circumstances, *i.e.,* various model sizes/types, and various dropout rates. Subsequently, we relax the two assumptions, *i.e.,* known labels and known exact sequence lengths, to evaluate the impact of them. In addition, we also evaluate the generalizability of GRAB with a multiclass classification task and with larger batch sizes. In most cases, if not otherwise noted, all approaches are run with their default hyperparameters setting and on the CoLA dataset with a batch size of 1 in practical settings.

5.4.1 Various Model Sizes and Types. Different sizes of BERT models are chosen for comparison. Besides BERT_{base}, another two structured models of the BERT model are chosen, which are BERT_{tiny} [25] and BERT_{large} [9]. BERT_{tiny} has 6 layers, 12 heads, and 67 million parameters. BERT_{large} has 24 layers, 16 heads, and 336 million parameters. We compare both variants of GRAB and all baseline approaches on these models.

To evaluate GRAB on various model types, we select RoBERTa [28], which is a variant of BERT. Three sizes are chosen, which are RoBERTa_{tiny} [52] with 4 layers, 8 heads, and 28 million parameters, RoBERTa_{base} with 12 layers, 12 heads and 125 million parameters,

and RoBERTa_{large}, with 24 layers, 16 heads and 355 million parameters. As the baseline approaches do not support attacking RoBERTa models in their implementations, we only evaluate our approach.

The results on various BERT models are shown in Table 3. For BERT_{tiny} and BERT_{base}, as the model size increases, the performance of both variants of GRAB and LAMP decreases as larger models have more parameters, posing difficulty to the optimization. However, the performance of DLG and TAG increases as the model size increases, aligning with the observation from TAG [8]. This is due to the difference in methodologies, as they only employ continuous optimization. Smaller models contain less information for these approaches to reach an optimum. However, this phenomenon is only observed in BERT_{tiny} and BERT_{base}. In BERT_{large}, which contains more layers and parameters, the performance of all approaches decreases dramatically except for GRAB, which utilizes dropout learning. This underscores the effectiveness of our dropout mask learning technique in countering the negative impacts of dropout, as a larger model incorporates more dropout layers.

Note that there is another hyperparameter setting for baseline approaches proposed by Balunovic *et al.* [4] specifically for BERT_{large} to counter the optimization difficulty posed by the architectures and parameters. This setting involves several changes, *i.e.,* more optimization steps, a different optimizer, a different learning rate decay scheduler, and a gradient clipping technique. We acknowledge that this setting can indeed resolve the optimization difficulty, and we adapt part of it into our approach. Specifically, a linear learning rate decay scheduler is adopted and the gradient clipping technique [14] is applied to clip the magnitude of the dummy embedding's gradient to 0.5 for more stable optimization, while the rest of the hyperparameters setting remains the same. All approaches are evaluated with the tuned hyperparameters setting and the results are shown in Table 4. The performance of all approaches is boosted significantly, and our approach with dropout mask learning still outperforms all baseline approaches.

For RoBERTa models, we first run our approach on all three sizes with the default hyperparameter setting, and the results are shown in Table 3. The performance of GRAB follows the same trend as in BERT models, the attack is weakened when model size increases. For RoBERTa_{large}, our attack without the incorporation of dropout mask learning fails to recover any meaningful text, while the method that incorporates dropout mask learning is only partially successful. The overall performance of GRAB on RoBERTa models falls short compared to BERT models, attributable to RoBERTa's increased complexity and more parameters.
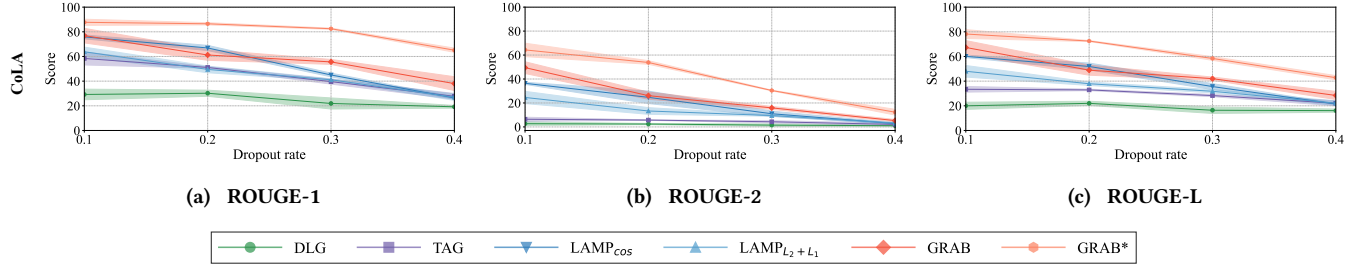
(a) ROUGE-1   (b) ROUGE-2   (c) ROUGE-L

Figure 4: Performance in practical settings with different dropout rates. Grab* is the variant with dropout mask learning.

Table 5: Ablation study on assumptions of known labels and known exact sequence lengths.

	B = 1			B = 2			B = 4		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Grab [1]	-	-	-	80.7 ± 3.1	53.4 ± 2.3	68.9 ± 1.8	76.5 ± 0.6	47.8 ± 1.0	63.9 ± 1.0
Grab [2]	**88.6 ± 1.6**	**65.7 ± 1.4**	**78.8 ± 1.1**	78.8 ± 0.9	51.9 ± 2.2	67.4 ± 0.4	67.7 ± 2.3	36.7 ± 2.8	55.5 ± 1.7
Grab [3]	-	-	-	**84.9 ± 1.2**	**56.8 ± 1.2**	**72.3 ± 1.2**	**79.2 ± 0.7**	**52.8 ± 3.0**	**66.9 ± 2.4**
Grab [4]	87.7 ± 2.5	64.3 ± 5.6	78.2 ± 3.6	79.3 ± 2.4	53.2 ± 2.7	69.0 ± 2.4	75.2 ± 1.6	43.9 ± 0.4	62.8 ± 1.2
	B = 8			B = 16			B = 32		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Grab [1]	71.0 ± 0.5	43.6 ± 1.3	58.8 ± 0.6	65.5 ± 1.4	**40.7 ± 2.1**	53.8 ± 1.7	53.8 ± 1.5	30.9 ± 2.5	44.8 ± 1.7
Grab [2]	65.8 ± 0.8	33.8 ± 2.3	53.3 ± 1.1	62.4 ± 1.6	32.1 ± 1.1	51.4 ± 0.9	53.8 ± 1.1	23.4 ± 1.1	44.7 ± 1.1
Grab [3]	**73.6 ± 1.7**	**45.6 ± 1.7**	**58.9 ± 2.2**	62.8 ± 2.7	38.0 ± 4.1	52.2 ± 2.9	**55.9 ± 1.7**	**31.3 ± 3.1**	45.9 ± 2.1
Grab [4]	68.5 ± 1.3	35.4 ± 1.5	54.8 ± 1.2	**67.7 ± 3.3**	33.7 ± 2.5	**54.9 ± 2.0**	54.9 ± 2.1	26.7 ± 1.2	**46.4 ± 1.9**

[1] Without the assumptions of known labels or sequence lengths. [2] Without the assumption of known labels. [3] Without the assumption of known sequence lengths. [4] With both assumptions of known labels and known sequence lengths.

The tuned hyperparameter setting is applied on RoBERTa$_{large}$, and the results are shown in Table 4. We observe that the performance of Grab with dropout mask learning is boosted significantly, while the performance of Grab without dropout mask learning only increases by a small level. As the RoBERTa$_{large}$ has more dropout layers, this further affirms the efficacy of applying the dropout mask learning technique to offset the dropout-induced noise.

We emphasize that Grab has a greater generalization ability to different model types, compared to existing attack LAMP [4], as LAMP involves a customized external auxiliary language model (GPT-2) which employs the same tokenizer as BERT for calculating perplexity [23] as part of the recovery loss. In the current implementation of LAMP, it does not provide the option to attack different model types. It is unclear how abandoning the auxiliary language model would affect the recovery.

The results of this ablation study demonstrate that Grab remains effective with various model sizes/types, revealing significant privacy vulnerabilities of language models in general.

5.4.2 Varying Dropout Rates. As mentioned in Section 2.2.1 and Section 2.3, dropout serves as a technique to prevent overfitting in model training and also a lightweight defense mechanism against gradient inversion [43]. To assess Grab under the scenarios with different dropout rates, we evaluate the performance of all approaches on BERT$_{base}$ model with increasing dropout rates. As a higher dropout rate simultaneously worsens the model utility, following the practice from LAMP, we train the models with 2 epochs and report the utility metric MCC [32]. We observe that with a dropout rate from 0.1 to 0.4, the MCC drops from 0.77 to 0, indicating that the model is completely dysfunctional. Thus, we do not evaluate a dropout rate higher than this.

Figure 4 shows the stable superiority of Grab with dropout mask learning compared to all baselines, despite the decreasing performance of all approaches. While the attack is weakened as the dropout rate increases, the utility of the models simultaneously has a dramatic deterioration, which is unacceptable in practical cases. Grab can still recover a large portion of the data even if the models are dysfunctional in terms of utility. This indicates that Grab does not rely on low dropout rates. Hence, increasing dropout rates cannot serve as an effective defense mechanism against Grab.

5.4.3 Assumptions Relaxation. We further evaluate the impact of the two adopted assumptions by existing work discussed in Section 2.2.2, *i.e.,* known labels and known sequence lengths.
Without known labels assumption. To launch the attack without known labels assumption, the attacker randomly initializes dummy labels and optimizes them simultaneously with the dummy embedding and the dropout mask. We run Grab with dropout mask learning without the known labels assumption on the CoLA dataset with batch sizes from 1 to 32 on BERT$_{base}$ model. The results are shown in Table 5. We observe that the performance of Grab remains consistent, compared to when the assumption exists, indicating that Grab is insensitive to this assumption.
Without known exact sequence lengths assumption. To launch the attack without the known exact sequence lengths assumption, we assume that only the longest length in the batch is known to the attacker, as it is significantly more practical to enumerate all possible longest lengths for the whole batch, compared to the existing assumption [4] where the attacker needs to enumerate all possible lengths for each sequence in the batch. This existing assumption becomes impractical when batch size grows. We run Grab with dropout mask learning without the known exact sequence lengths assumption on the CoLA dataset with batch sizes from 2 to 32 on
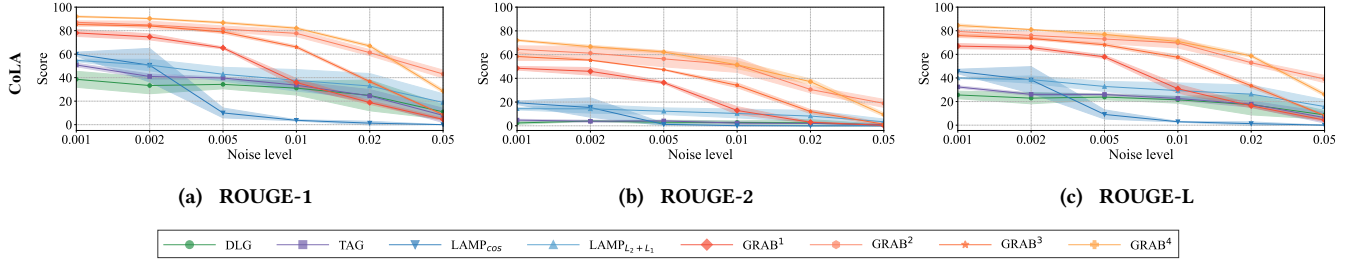
(a) ROUGE-1	(b) ROUGE-2	(c) ROUGE-L

Figure 5: Performance in practical settings against defense by different gradient noise levels. GRAB[1]: Without noise norm or dropout mask learning. GRAB[2]: Without dropout mask learning. GRAB[3]: Without noise norm. GRAB[4]: With noise norm and dropout mask learning.

$BERT_{base}$ model, as batch size 1 only involves a single sentence in the batch. The results are shown in Table 5. GRAB achieves an even higher attack efficacy, until the batch size of 16. This is expected, as the recovered sequence lengths are no longer constrained if the sequences are recovered at different indexes, and by involving the padding token, the length for each index can be adjusted dynamically towards a lower recovery loss. The shorter sequences can append padding tokens and the longer sequences do not get cut off. The results indicate that GRAB is also insensitive to this assumption. **Without both assumptions**. We run GRAB with dropout mask learning without both assumptions on the CoLA dataset with batch sizes from 2 to 32 on $BERT_{base}$ model, and the results are shown in Table 5. We observe a consistent attack efficacy phenomenon, further indicating that GRAB is insensitive to both assumptions and can even perform better without these assumptions, mainly because the sequence lengths constraint is lifted.

The results of this ablation study further showcase the practicality of GRAB in the absence of the two assumptions, where the prior knowledge of the labels and the exact sequence lengths is not provided to the attacker.

5.4.4 Multi-class Classification. The evaluation of GRAB is with a binary classification task. When the known label assumption is relaxed, the difference between multi-class and binary classification mainly lies in the possible number of ground truth labels. To assess the generalization of GRAB to such a more challenging task, we evaluate it with a 10-class classification task with the Yahoo Answers Topics dataset [22] under practical settings without the known label assumption. The same data selection procedure as our benchmark datasets is followed. The results are shown in Table 9 in Appendix A. The evaluation demonstrates that GRAB is resilient to such challenging tasks, and can effectively recover both the ground truth labels and the textual data, regardless of the number of possible ground truth labels.

5.4.5 Large Batch Sizes. As shown in Figure 2 and Figure 3, the attack efficacy for all approaches wanes as the batch size grows, mainly due to the gradient information for all sequences in the batch being averaged together. To further explore the effect of the batch size, we evaluate GRAB with the CoLA dataset with larger batch sizes up to 128 [3], the largest batch size used originally to pre-train

---
[3]Extra data is selected with the same data selection procedure as in the original data selection with the same random seed.

Table 6: DP-SGD utility and privacy trade-off.

Noise level	0.001	0.002	0.005	0.01	0.02	0.05
MCC	0.74	0.68	0.60	0.51	0.45	0.25
ϵ	1.10×10^{10}	2.75×10^{9}	4.38×10^{8}	1.08×10^{8}	2.55×10^{7}	2.37×10^{6}

BERT [41]. The results are shown in Table 10 in Appendix A. The evaluation demonstrates that the attack efficacy of GRAB continues to drop as batch size grows. However, a significant portion of the private data is still recovered, further showcasing the capability of GRAB and underlying the severity of this privacy threat, even with large batch sizes.

5.5 Proactive Defenses

We consider two widely used proactive defenses, *i.e.,* gradient noise [49, 55] and gradient pruning [55], which aim to counteract gradient inversion. We explore their effect in all approaches in practical settings on $BERT_{base}$ model. Huang *et al.* [19] evaluate other defense mechanisms such as gradient encryption [3] and input encoding [20, 53]. Since gradient encryption requires a special setup and is costly to run, and input encoding is primarily for image data, we do not evaluate them in this work.

5.5.1 Gradient Noise. Zhu *et al.* [55] and Wei *et al.* [49] suggest a defense that adds Gaussian or Laplacian noise to the gradients. We follow a standard DP-SGD [1] process, where the client normalizes/clips the gradients before adding the noise. However, if gradient noise is not a part of the FL training protocol, this can hint that it is used as a defense. The attacker can check the use of gradient noise by passing a dummy data sample through the model and monitoring the differences in the norm magnitudes of the dummy gradients and the client's gradients. If the difference exceeds a threshold, the attacker can deduce the defense is in place and adjust the magnitude of the dummy gradients for a more effective search.

To evaluate the privacy-utility trade-off, the language model is trained on the full CoLA training set (8,551 samples) for 2 epochs with a range of noise levels from 0.001 to 0.05. The noise levels, model utility (MCC), and privacy budgets (ϵ) are summarized in Table 6. A default target delta (δ) of 1×10^{-5} is used when computing the privacy budgets. Noise levels beyond 0.05 are not explored as the MCC of the model drops significantly from 0.77 to 0.25. We observe that the model is sensitive to small noise levels in terms of model utility, following similar observations in previous works [4]. Specifically, as language models usually contain a large

Table 7: Performance in practical settings against defense by different gradient pruning ratios.

	Gradient Pruning Ratio: 0.75 (MCC=0.72)			Gradient Pruning Ratio: 0.90 (MCC=0.67)			Gradient Pruning Ratio: 0.95 (MCC=0.61)			Gradient Pruning Ratio: 0.99 (MCC=0.45)		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
DLG	1.1 ± 0.4	0.0 ± 0.0	1.1 ± 0.3	0.4 ± 0.2	0.0 ± 0.0	0.4 ± 0.2	0.5 ± 0.2	0.0 ± 0.0	0.5 ± 0.2	0.4 ± 0.3	0.0 ± 0.0	0.4 ± 0.3
TAG	0.2 ± 0.3	0.0 ± 0.0	0.2 ± 0.3	0.1 ± 0.1	0.0 ± 0.0	0.1 ± 0.1	0.2 ± 0.1	0.0 ± 0.0	0.2 ± 0.1	0.0 ± 0.1	0.0 ± 0.0	0.0 ± 0.1
LAMP_{cos}	27.9 ± 6.4	8.8 ± 2.8	21.3 ± 5.3	7.9 ± 0.8	1.8 ± 0.6	6.7 ± 0.7	0.1 ± 0.1	0.0 ± 0.0	0.1 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
$\text{LAMP}_{L_2+L_1}$	0.2 ± 0.2	0.0 ± 0.0	0.2 ± 0.2	0.4 ± 0.3	0.0 ± 0.0	0.4 ± 0.3	0.3 ± 0.2	0.0 ± 0.0	0.3 ± 0.2	0.3 ± 0.1	0.0 ± 0.0	0.3 ± 0.1
GRAB [1]	0.9 ± 0.4	0.0 ± 0.0	0.9 ± 0.4	0.4 ± 0.2	0.0 ± 0.0	0.4 ± 0.2	0.5 ± 0.4	0.0 ± 0.0	0.5 ± 0.4	0.3 ± 0.1	0.0 ± 0.0	0.3 ± 0.1
GRAB [2]	79.7 ± 1.0	50.3 ± 1.5	69.1 ± 1.8	72.7 ± 8.5	41.2 ± 9.3	61.2 ± 8.1	76.0 ± 3.6	45.5 ± 5.2	65.2 ± 3.7	67.1 ± 3.2	35.2 ± 4.7	56.6 ± 2.9
GRAB [3]	0.8 ± 0.1	0.0 ± 0.0	0.7 ± 0.0	0.9 ± 0.6	0.0 ± 0.0	0.9 ± 0.6	0.9 ± 0.2	0.0 ± 0.1	0.9 ± 0.2	0.7 ± 0.3	0.0 ± 0.0	0.7 ± 0.3
GRAB [4]	**89.2 ± 1.6**	**65.8 ± 2.1**	**78.8 ± 1.4**	**89.4 ± 1.6**	**67.8 ± 4.4**	**80.8 ± 1.9**	**87.6 ± 0.9**	**65.3 ± 2.9**	**78.0 ± 2.0**	**85.9 ± 1.1**	**60.2 ± 1.5**	**75.4 ± 1.5**

[1] Without pruning mask or dropout mask learning. [2] Without dropout mask learning. [3] Without pruning mask. [4] With both pruning mask and dropout mask learning.

number of parameters, they are generally pre-trained on a large corpus, so the gradients obtained by the attacker with training on downstream tasks tend to be small. Clipping and adding noise leads to a significant model utility loss, even with only small noise levels and impractically large privacy budgets measured with DP.

To evaluate the attack efficacy against the gradient noise defense, we consider four variants of GRAB, which include combinations of either enabling or disabling dropout mask learning or norm normalization. Figure 5 demonstrates that GRAB, which combines dropout mask learning and norm normalization, outperforms all baseline approaches, achieving substantial data recovery across almost all noise levels and demonstrating the enhancement and effectiveness of both techniques. Similar to dropout, gradient noise weakens the attack, but higher noise levels sacrifice the model utility. At the noise level of 0.05, the model is nearly dysfunctional in terms of model utility, and GRAB with only norm normalization actually outperforms GRAB with both techniques, as the noise level is high enough to perturb the dropout mask learning process. Nevertheless, gradient noise as a defense mechanism cannot mitigate GRAB, even by compromising most of the model utility.

5.5.2 Gradient Pruning. Gradient pruning is another defense by modifying the shared gradients in FL [55]. We follow the gradient pruning variant evaluated in LAMP [4] to prune a certain percentage of the gradients. The attacker can infer whether this defense is employed by observing the zero entries of the client's gradients. Capitalizing on this knowledge, the corresponding gradient entries in the dummy gradients of the attacker can also be zeroed out to aid the gradient matching process for improved data recovery.

The language model is trained on the full CoLA dataset for 2 epochs with pruning percentages from 0.75 to 0.99. We run experiments on 4 variants of GRAB, which are combinations of turning on and off dropout mask learning or pruning mask. Table 7 shows GRAB with both techniques combined recovers a significant portion of the data and remains consistent across all prune percentages, while all baseline approaches have a significant drop in recovery rate starting from prune percentage 0.75, or even completely mitigated. This deviates from the observation of LAMP, as their experiments are run in benchmark settings. These results demonstrate that gradient pruning cannot mitigate our attack, whereas it successfully mitigates all other approaches in practical settings.

6 Conclusion

In this work, we propose GRAB, a domain-specific gradient inversion attack with hybrid optimization to recover training data of language models in practical FL scenarios. GRAB exploits the gradient decent process during training by incorporating dropout-aware continuous optimization for token recovery, and the beam search discrete optimization for token reordering. Our approach effectively recovers a large portion of training data and exhibits notable adaptability to a variety of challenging settings. It also demonstrates significant resilience against existing defense mechanisms.

GRAB represents a step forward in understanding the privacy threats posed by gradient inversion attacks against the emerging FL training and fine-tuning modes of language models. However, it currently faces limitations in validating whether the recovered data is part of the training dataset. To further enhance GRAB, applying techniques such as membership inference [6, 18, 36, 44, 50] in the context of language models could be explored. For example, the loss value of the target model on the recovered data can be utilized to determine the membership threshold [7, 50].

Our findings also highlight the urgent need for the research community to develop robust defense mechanisms to mitigate such threats with consideration of practical training settings and minimum adversary knowledge. Existing defenses that perturb gradients may mitigate the effectiveness of GRAB, as it relies on minimizing the distance between the gradients generated by the original data and those from the recovered data. However, these defenses often cause significant model utility loss due to the small magnitudes of these gradients. To successfully defend GRAB and similar gradient-based attacks, the defense needs to explore other avenues than perturbing gradients. We suggest potential defenses in two phases of the training, *i.e.,* at the start of the training and during the training process.

- At the start of the training, one potential defense is to transform input token embeddings or to substitute the input tokens with different tokens with similar embeddings. By doing so, the generated gradients may not be pointed directly to the original tokens.
- During the training process, the defense might consider only training and perturbing the gradients of the added layers (*e.g.,* the classification layer) with gradient noise or gradient clipping. Since these layers have gradients with larger magnitudes, higher noise levels may be tolerated.

Acknowledgments

References

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

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[3] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security* 13, 5 (2017), 1333–1345.

[4] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting text from gradients with language model priors. In *the 2022 Advances in Neural Information Processing Systems (NeurIPS)*. 7641–7654.

[5] Dongqi Cai, Shangguang Wang, Yaozong Wu, Felix Xiaozhu Lin, and Mengwei Xu. 2023. Federated few-shot learning for mobile NLP. In *Proceedings of the 2023 Annual International Conference on Mobile Computing and Networking (MobiCom)*. 1–17.

[6] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*. 1897–1914.

[7] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *Proceedings of the 2021 USENIX Security Symposium (USENIX Security)*. 2633–2650.

[8] Jieren Deng, Yijue Wang, Ji Li, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. 2021. Tag: Gradient attack on transformer-based language models. *arXiv preprint arXiv:2103.06819* (2021).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*.

[11] FedML. 2023. Releasing FedLLM: Build Your Own Large Language Models on Proprietary Data using the FedML Platform. https://blog.fedml.ai/releasing-fedllm-build-your-own-large-language-models-on-proprietary-data-using-the-fedml-platform/. Accessed: January, 2024.

[12] Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. 2022. Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *Proceedings of the 2022 International Conference on Learning Representations (ICLR)*.

[13] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning?. In *the 2020 Advances in Neural Information Processing Systems (NeurIPS)*. 16937–16947.

[14] Jonas Geiping, Liam Fowl, and Yuxin Wen. 2023. breaching. https://github.com/JonasGeiping/breaching. Assessed: August 2023.

[15] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering private text in federated learning of language models. In *the 2022 Advances in Neural Information Processing Systems (NeurIPS)*. 8130–8143.

[16] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[17] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proceedings of the 2020 International Conference on Learning Representations (ICLR)*.

[18] Pingyi Hu, Zihan Wang, Ruoxi Sun, Hu Wang, and Minhui Xue. 2022. M^{4}I: Multi-modal Models Membership Inference. In *the 2022 Advances in Neural Information Processing Systems (NeurIPS)*. 1867–1882.

[19] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. 2021. Evaluating gradient inversion attacks and defenses in federated learning. In *the 2021 Advances in Neural Information Processing Systems (NeurIPS)*. 7232–7241.

[20] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. 2020. Instahide: Instance-hiding schemes for private distributed learning. In *Proceedings of the 2020 International Conference on Machine Learning (ICML)*. 4507–4518.

[21] Huggingface. 2022. Tweet Sentiment Extraction. https://huggingface.co/datasets/SetFit/tweet_sentiment_extraction. Accessed: January, 2024.

[22] Huggingface. 2024. Yahoo Answers Topics. https://huggingface.co/datasets/community-datasets/yahoo_answers_topics. Accessed: January, 2024.

[23] Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America* 62, S1 (1977), S63–S63.

[24] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. 2021. Gradient inversion with generative image prior. In *the 2021 Advances in Neural Information Processing Systems (NeurIPS)*. 29898–29908.

[25] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* (2019).

[26] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE transactions on knowledge and data engineering* 34, 1 (2020), 50–70.

[27] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. 74–81.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[29] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*.

[30] Jiahao Lu, Xi Sheryl Zhang, Tianli Zhao, Xiangyu He, and Jian Cheng. 2022. APRIL: Finding the Achilles' Heel on Privacy for Vision Transformers. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10051–10060.

[31] Mengyao Ma, Yanjun Zhang, Pathum Chamikara Mahawaga Arachchige, Leo Yu Zhang, Mohan Baruwal Chhetri, and Guangdong Bai. 2023. Loden: Making every client in federated learning a defender against the poisoning membership inference attacks. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 122–135.

[32] Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405, 2 (1975), 442–451.

[33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 2017 International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1273–1282.

[34] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal* 5, 4 (2014), 1093–1113.

[35] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*. 691–706.

[36] Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022. Quantifying Privacy Risks of Masked Language Models Using Membership Inference Attacks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8332–8347.

[37] Nikita Munot and Sharvari S Govilkar. 2014. Comparative study of text summarization methods. *International Journal of Computer Applications* 102, 12 (2014).

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *the 2022 Advances in Neural Information Processing Systems (NeurIPS)*. 27730–27744.

[39] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075* (2005).

[40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[41] Google Research. 2020. TensorFlow code and pre-trained models for BERT. https://github.com/google-research/bert. Accessed: January 2024.

[42] Stuart J Russell. 2010. *Artificial intelligence a modern approach*. Pearson Education, Inc.

[43] Daniel Scheliga, Patrick Mäder, and Marco Seeland. 2023. Dropout is NOT All You Need to Prevent Gradient Leakage. In *Proceedings of the 2023 Association for the Advancement of Artificial Intelligence (AAAI)*. 9733–9741.

[44] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. 3–18.

[45] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1631–1642.

[46] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. 2022. Fedbert: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–26.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *the 2017 Advances in Neural Information Processing Systems (NeurIPS)*.

[48] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational*

Linguistics 7 (2019), 625–641.

[49] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. 2020. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397* (2020).

[50] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *Proceedings of the 2018 IEEE Computer Security Foundations Symposium (CSF)*. 268–282.

[51] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16337–16346.

[52] Haisong Zhang. 2020. roberta-tiny-cased. https://github.com/haisongzhang/roberta-tiny-cased. Accessed: January 2024.

[53] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *Proceedings of the 2018 International Conference on Learning Representations (ICLR)*.

[54] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610* (2020).

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

A Supplementary Tables

A.1 FL Setup

Table 8 shows the result of GRAB in benchmark settings on the CoLA dataset with batch sizes from 1 to 32 on BERT$_{base}$ with multiple (5) clients or a single client. In the case of multiple clients, the victim client possesses the same 64 samples as in the main experiments, and all other clients possess 64 samples that do not overlap. The performance remains consistent between the two FL setups, where the attack efficacy of the multiple clients setup is slightly better in most cases. Due to this consistency, we adopt the single client FL setup for simplicity as they are functionally equivalent.

A.2 Multi-class Classification

Table 9 shows the results of GRAB in practical settings on the Yahoo Answers Topics dataset with batch sizes from 1 to 32 on BERT$_{base}$ without the known label assumption. The dataset consists of samples from 10 classes. The evaluation shows that GRAB is robust in handling such challenging tasks and can accurately recover both the ground truth labels and the textual data.

A.3 Large Batch Size

Table 10 shows the results of GRAB in practical settings on the CoLA dataset with batch sizes from 64 to 128 on BERT$_{base}$. The evaluation reveals that the attack efficacy of GRAB diminishes as the batch size increases. Nevertheless, a substantial amount of private data is still recovered, highlighting GRAB's strong recovery capability and underscoring the serious privacy risk it poses, even with large batch sizes.

B Additional Experimental Settings

B.1 Empirical Experiment Implementation

We implement our approach in Python 3.9.4 with PyTorch 2.0.1. Different platforms are used to conduct our experiments for practical efficiency reasons. They include three high-performance computer clusters and two workstations. The three high-performance computer clusters run on Rocky Linux with NVIDIA H100 GPUs, CentOS with NVIDIA V100 GPUs, and SUSE Linux with NVIDIA

Table 8: Performance with multiple clients and a single client.

	B = 1			B = 2		
	R-1	R-2	R-L	R-1	R-2	R-L
Single	86.4 ± 4.7	67.6 ± 9.3	78.3 ± 6.2	79.8 ± 2.8	57.1 ± 3.6	70.0 ± 1.6
Multi	**91.8 ± 1.9**	**73.7 ± 6.7**	**82.5 ± 4.3**	**84.4 ± 3.6**	**60.7 ± 8.0**	**74.1 ± 5.5**
	B = 4			B = 8		
Single	61.9 ± 3.7	35.2 ± 4.9	51.5 ± 3.1	53.3 ± 3.5	**26.4 ± 4.3**	44.7 ± 2.6
Multi	**70.5 ± 2.9**	**42.8 ± 5.2**	**59.2 ± 3.8**	**55.8 ± 3.0**	25.8 ± 2.5	**45.3 ± 1.6**
	B = 16			B = 32		
Single	45.5 ± 2.2	18.1 ± 2.3	38.1 ± 2.3	31.9 ± 6.2	8.4 ± 3.8	28.0 ± 4.6
Multi	**50.2 ± 4.6**	**21.8 ± 3.8**	**41.3 ± 3.5**	**34.5 ± 2.7**	**11.1 ± 2.1**	**30.4 ± 2.2**

Table 9: Performance with a 10-class classification task.

	B = 1			B = 2		
	R-1	R-2	R-L	R-1	R-2	R-L
GRAB	85.3 ± 1.9	54.2 ± 4.1	73.6 ± 2.3	74.4 ± 4.2	42.1 ± 4.1	61.4 ± 3.6
	B = 4			B = 8		
GRAB	62.9 ± 3.7	31.4 ± 2.0	50.7 ± 2.5	56.1 ± 3.7	**23.8 ± 2.0**	45.2 ± 3.0
	B = 16			B = 32		
GRAB	53.2 ± 4.2	20.0 ± 2.3	41.8 ± 2.8	51.3 ± 6.5	18.9 ± 4.2	41.1 ± 5.5

Table 10: Performance with larger batch sizes.

	B = 64			B = 128		
	R-1	R-2	R-L	R-1	R-2	R-L
GRAB	45.2 ± 2.1	19.3 ± 3.5	38.9 ± 2.3	39.7 ± 0.7	13.6 ± 0.5	35.3 ± 1.0

P100 GPUs and NVIDIA H100 GPUs, respectively. The two workstations run on Ubuntu with NVIDIA RTX A6000 GPUs and Windows 10 with NVIDIA A4000 GPUs, respectively. To ensure consistent results, we test all approaches across all platforms using a single sentence from the CoLA dataset. By setting a fixed random seed, we ensure the consistency and the reproducibility of both intermediate and final outcomes.

B.2 Model Implementation in Practical Settings

In the majority of experiments conducted within the practical settings, we run two variants of GRAB, *i.e.,* with and without dropout mask learning. For all approaches, the language model of the victim client freezes the embedding layers and activates dropout during training. For all baseline approaches and GRAB without dropout mask learning, the language model of the attacker deactivates dropout during recovery, as this serves as a better method when the dropout mask realization of the client is unknown [43]. For GRAB with dropout mask learning, the attacker initializes a dropout mask and retains it for optimization.

B.3 ROUGE Score Calculation

For the detailed calculation of the ROUGE score, we compare a reference sequence with each recovered sequence in the data batch and select the maximum ROUGE score. This way ensures the similarity of a recovered data batch with a different sequence order from the reference data batch is obtained. Note that in LAMP [4], the recovered data batch is directly compared with the reference data batch with the same default index, which can result in a miscalculation of the metric.