Data Reconstruction and Protection in Federated Learning for Fine-Tuning Large Language Models

Fei Wang, *Student Member, IEEE,* Baochun Li, *Fellow, IEEE*

***Abstract*—Federated learning can facilitate multiple parties to train a shared model on their own private data in a communication-efficient manner. It offers significant benefits for fine-tuning pre-trained large language models, as it supports distributed fine-tuning with a wider range of diverse data while preserving data privacy. However, recent research has revealed a potential privacy vulnerability in federated learning, specifically in the sharing of gradients from clients to server. This vulnerability can lead to the leakage of training data for Transformer-based large language models, thereby allowing the recovery of textual data. In this paper, we conduct a comprehensive evaluation of the effectiveness of the state-of-the-art gradient leakage attacks on textual data within the context of fine-tuning large language models. Our findings reveal that the key element for the attack's success — the target gradient — is not as readily obtainable for the adversary as previously assumed, particularly in regards to the Transformer architecture and practical federated learning settings. A technical error in their implementations has inadvertently caused the gradient to become more associated with the target data than intended. With the error fixed and when following the conventional federated learning framework, gradient leakage attacks pose minimal threats to large language models.**

***Index Terms*—Large language models, gradient leakage attack, federated learning, fine-tuning**

I. Introduction

Since the debut of ChatGPT in late 2022, large language models (LLMs) have made an indelible mark with their groundbreaking capability of comprehending and generating human-like language. By leveraging massive amounts of textual data and powerful computational resources, LLMs have surpassed previous standards and redefined the limits of natural language processing (NLP) capabilities. Pre-trained LLMs, such as Google's BERT [1], OpenAI's GPT-4 [2], and Meta AI's Llama 2 [3], have gained general knowledge across a diverse range of natural language processing (NLP) tasks [4], including text classification, translation, sentiment analysis, question answering, and more. Thanks to the open-source library, HuggingFace's `Transformers` [5], which facilitates the distribution and utilization of a wide array of Transformer-based models, users can conveniently access and leverage various pre-trained LLMs to cater to their specific tasks or even fine-tune them with their own datasets.

When data is distributed across various devices, federated learning [6] becomes an ideal approach for fine-tuning pre-trained LLMs while preserving data privacy. For example, a healthcare organization aims to enhance the accuracy and efficiency of a public pre-trained LLM in a clinical natural language processing task. However, each hospital or clinic possesses its own sensitive health records and patient data. In such a scenario, federated learning enables collaborative fine-tuning of the pre-trained LLM while ensuring that sensitive data remains on the local devices of each hospital or clinic. Several studies have explored the potential of leveraging federated learning for the fine-tuning of LLMs [7]–[10].

Despite its promise, the ability to preserve privacy with federated learning has been questioned in the recent literature. Numerous studies have demonstrated that federated learning is susceptible to gradient leakage attacks [11], wherein sensitive data can be reconstructed using gradient updates [12]–[16]. This privacy vulnerability is not limited to image data alone. As more researchers investigate Transformer-based models, they have discovered that textual data, despite its discrete properties, is not exempt from gradient leakage attacks [17], [18].

Even though private data is not directly transmitted in federated learning, clients are still required to send their *updates* to the server for aggregation after local training. An honest-but-curious server, while following the protocol of federated learning, may attempt to extract sensitive information about the training data from the corresponding updates. In the majority of earlier studies on gradient leakage attacks, the assumption was made that the updates transmitted from clients to the server are gradients computed in a single step of local gradient descent. However, this scenario represents the simplest case of federated optimization and does not accurately reflect the standard practice in the design of federated learning algorithms, known as FedAvg [6], where the local training process within a single communication round typically involves multiple iterations of gradient descent. Our previous work made a clear distinction between gradient sharing and weight delta sharing and further showed that the weight delta sharing in practical federated learning imposes stricter requirements on the attack model and leads to a more challenging attack process [19]. However, our work was limited to computer vision tasks and the reconstruction of image data.

Even when considering gradient sharing, the gradient implemented in the existing work does not accurately mirror real-world local training in federated learning. The target gradient that the server receives is not derived when the model is in the correct state during training. This unintentional mistake happens to strengthen the relationship between the gradient and the training data, particularly for Transformer-based models, more than it is supposed to be. This also contributes to the impressive attack performance on textual

Fei Wang and Baochun Li are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: silviafey.wang@utoronto.ca; bli@ece.toronto.edu).

data.

In this paper, our aim is to examine the effectiveness of data reconstruction in federated learning, specifically for fine-tuning Large Language Models (LLMs). We re-evaluate cutting-edge gradient leakage attacks tailored for textual data, specifically LAMP [18]. We delve into more practical federated learning settings, focusing on examining model forwarding under different states and FedAvg's weight delta sharing. Furthermore, we broaden our evaluation to include larger language models that the literature has not previously considered, such as GPT-2 [20]. Through our investigation, we observed a noticeable decrease in the attack capability as the model size increases, even when considering minimal data and batch size during the fine-tuning process. Additionally, our findings suggest that, even with gradient sharing, the target gradient derived from clients' local training is still challenging for the server to interpret, particularly in the context of LLMs due to specific modules within the Transformer architecture, such as dropout. When these layers are not unintentionally blocked, in contrast to previous studies, the reconstruction capability is significantly compromised when the target gradient is derived through the correct model state. Moreover, even under the most optimal conditions, attackers can only reconstruct a short sentence of text in the correct order. By using training data that includes long sentences or paragraphs, employing large batch sizes and numbers of epochs, a substantial level of resistance against gradient leakage attacks can be achieved inherently.

The rest of this paper is organized as follows. In Section II, we provide a background on federated learning, specifically highlighting the distinctions between FedSGD and FedAvg. Additionally, we discuss gradient leakage attacks against federated learning and review their evolution from image data to textual data, including the state-of-the-art methods used in our evaluation. Furthermore, we introduce the background of efficient fine-tuning techniques used for pre-trained LLMs via federated learning. Sections III and IV delve into our discoveries and analysis concerning gradient leakage attacks on the fine-tuning of LLMs, where we investigate the impact of model state and FedAvg settings on the gradient leakage. Section V summarizes and concludes the paper.

II. Preliminaries & Related Work

A. Federated Learning

With federated learning [6] a deep neural network model is trained using private data distributed across client devices. A server collects and aggregates the local updates from the clients and manages a shared global model.

Federated Averaging. Federated Averaging (FedAvg) is the standard federated learning algorithm proposed in [6]. With FedAvg, each client performs multiple local training epochs over multiple minibatches from its local dataset in each communication round. The client sends the updated local model weight w_{t+1}^k to the server for aggregation. The new global model weight is then updated with the following formula:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k. \tag{1}$$

Alternatively, the client can send the weight delta update $\Delta w_{t+1}^k := w_{t+1}^k - w_t^k$ to the server, and the new global model weight is updated as the following:

$$w_{t+1} \leftarrow w_t + \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_{t+1}^k. \tag{2}$$

FedSGD. Federated SGD (or FedSGD) is a naive baseline algorithm of federated optimization proposed in [6] for comparison. In each communication round of FedSGD, each client performs only a single iteration of local training over a batch in its full local dataset. After the training, the client sends the computed gradient $g_t^k = \nabla F_k(w_t)$ to the server for aggregation. The new global model weight is then updated with the following formula:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_t^k. \tag{3}$$

FedSGD can be considered as a special and simplest case of FedAvg when the number of epochs E is equal to one and the local minibatch size B is equal to the number of local data samples n_k, which can be denoted as $E = 1, B = n_k$. We highlight the importance of differentiating between FedSGD and FedAvg in this context, as it directly impacts the server's access and significantly influences the nature of gradient leakage attacks. Unfortunately, this distinction is frequently overlooked in existing literature, where FedSGD is often considered synonymous with federated learning itself. We will delve into the impact of this distinction further in Section IV.

B. Gradient Leakage Attacks

Gradient leakage attack [11] focuses on the scenario where an honest-but-curious server attempts to recover private training data from a client using the gradients the server received. These attacks involve solving an optimization problem, defined as follows:

$$\mathbf{x}'^*, \mathbf{y}'^* = \arg\min_{\mathbf{x}', \mathbf{y}'} \mathcal{D}\left(\frac{\partial \mathcal{L}(F_w(\mathbf{x}'), \mathbf{y}')}{\partial w}, \frac{\partial \mathcal{L}(F_w(\mathbf{x}^*), \mathbf{y}^*)}{\partial w}\right). \tag{4}$$

In this optimization problem, $\nabla^* : \frac{\partial \mathcal{L}(F_w(\mathbf{x}^*), \mathbf{y}^*)}{\partial w}$ represents the target gradient received from the target client, and $\nabla' : \frac{\partial \mathcal{L}(F_w(\mathbf{x}'), \mathbf{y}')}{\partial w}$ represents the dummy gradient obtained through gradient descent by feeding dummy data $(\mathbf{x}', \mathbf{y}')$ into the model. The server has white-box access to the same model since it is the global model shared between the server and clients at the beginning of the communication round. Through iterations of gradient descent shown in Eqs. (5) and (6), the dummy data $(\mathbf{x}', \mathbf{y}')$ can be optimized to approach the target training data, as the dummy gradient matches the target gradient.

$$\mathbf{x}' \leftarrow \mathbf{x}' - \lambda \cdot \frac{\partial \mathcal{D}(\nabla^*, \nabla')}{\partial \mathbf{x}'} \tag{5}$$

$$\mathbf{y}' \leftarrow \mathbf{y}' - \lambda \cdot \frac{\partial \mathcal{D}(\nabla^*, \nabla')}{\partial \mathbf{y}'} \tag{6}$$

The loss function $\mathcal{D}(\cdot)$ measures the distance between the dummy gradient ∇' and the target gradient ∇^*. The

distance can take on different forms, such as L_2 distance $\|\nabla' - \nabla^*\|^2$ [11], cosine distance $1 - \frac{\langle \nabla', \nabla^* \rangle}{\|\nabla'\| \|\nabla^*\|}$ [13], or L_2 combined with L_1 distance $\|\nabla' - \nabla^*\|$ [17]. The fundamental mechanism of gradient leakage attacks is depicted in Fig. 1.

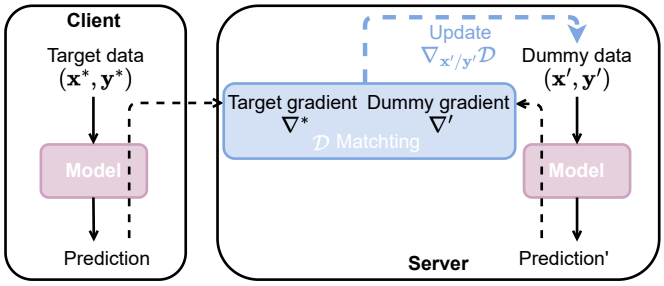

Fig. 1: Gradient matching and dummy data optimization at the federated learning server in existing gradient leakage attacks. These attacks assume that a federated learning client sends its gradients to the server in each communication round.

C. Existing work on Gradient Leakage Attacks against Federated Learning

Zhu *et al.* [11] pioneered the study of private training data reconstruction from gradient leakage and proposed the first optimization-based gradient leakage attack named DLG. While the primary emphasis of their work is on vision tasks, DLG also showcases its capability to recover text with language models such as BERT [1]. Several subsequent studies have made advances in improving the data reconstruction ability, as evidenced by [12]–[16]. These studies have incorporated more sophisticated techniques, such as true label estimation and the utilization of prior knowledge on image data. Furthermore, they have explored the impact of such attacks in a wider range of scenarios, including increased training epochs and larger batch sizes. Nevertheless, these attacks are primarily designed for reconstructing image data.

With a focus on Transformer-based language models, Deng *et al.* [17] proposed a novel gradient leakage attack, named TAG, that employs an extended distance function combining both L2 and L1 norms. This approach more accurately measures the discrepancy between the target and dummy gradients, thereby enhancing the ability to recover private text across more diverse model weight distributions when compared to DLG [11]. While their attack is particularly framed for the domain of natural language processing (NLP), it is not exclusively tailored to lanugage tasks and makes marginal improvement over existing gradient leakage attacks that reconstruct private image data in vision tasks as well.

LAMP [18], however, is a more sophisticated attack tailored to reconstructing textual data from gradients. In addition to continuous optimization using gradient descent shown in Section II-B, LAMP incorporates discrete optimization in each iteration to further refine the order of the recovered token embeddings. This discrete optimization involves generating candidates through sequence transformations such as swapping or moving tokens, and selecting the best candidate based on prior knowledge of natural text distribution derived from an auxiliary language model. This discrete optimization technique in LAMP effectively handles the discrete nature of text data, enabling it to successfully reconstruct sequences with higher precision and fidelity.

We consider LAMP to be the most potent gradient leakage attack on textual data based on their experimental results. Therefore, we utilize LAMP as an example in our evaluations. However, it is worth noting that our findings are generally applicable to all gradient leakage attacks, including DLG [11] and TAG [17], as they share a common core mechanism, which involves the optimization process described in Section II-B and Eqs. (4) to (6). Furthermore, all the main models considered in these attacks have been restricted to BERT and its variants. In this work, we also explore the vulnerability of up-to-date large language models [21] to data reconstruction.

D. Fine-Tuning Pre-Trained LLMs via Federated Learning

Pre-trained LLMs can serve as a solid starting point for federated learning, rather than starting the training from scratch [22]. Federated learning, on the other hand, broadens the accessibility of data and distributes the training workload across multiple participants for fine-tuning these foundation models [10] to improve its performance on a new task, typically with a smaller dataset. Consequently, when employing federated learning to fine-tune large language models, the data associated with the new task becomes the primary target for gradient leakage attackers.

However, the enormous size of these foundation models with billions of parameters can impose substantial computational burdens and significant communication overheads during fine-tuning and transmission. Parameter-efficient fine-tuning (PEFT) methods [23] can dramatically reduce the number of parameters optimized and communicated in federated learning to mitigate this challenge [9]. To achieve parameter-efficient fine-tuning, we utilize an adapter-based fine-tuning technique recently proposed by Microsoft called Low-Rank Adaptation (LoRA) [24]. LoRA introduces small task-specific adapters to the pre-trained model, significantly reducing the number of parameters required for fine-tuning large language models compared to fine-tuning all layers of the pre-trained model. For example, a GPT-3 model has 175 billion parameters, while LoRA can reduce the trainable parameters roughly by 10,000 times [24]. This approach allows the pre-trained model to retain its general functionality while adapting specifically to the target task. In conjunction with federated learning, the models and model updates optimized and transmitted only pertain to the parameters of the LoRA adapters. In this work, we explore both full-parameter fine-tuning and parameter-efficient fine-tuning when evaluating the effectiveness of gradient leakage attacks.

III. Ideals vs. Realities: Why Do Existing Gradient Leakage Attacks Work So Well?

Researchers may overlook the implementation details when converting their algorithms into code, and even minor mistakes could significantly impact the effectiveness of the algorithm.

Optimization-based gradient leakage attacks against large language models serve as a prime example of this oversight. In the section, we shall demonstrate the gap between the ideal conditions for such attacks, assumptions in the existing literature, and their actual implementation realities.

Let us consider the elephant in the room first. One common assumption made in previous work is that clients in federated learning simply perform FedSGD and send the gradient in a single step of gradient descent over their local datasets [15], [16]. The gradient $\frac{\partial \mathcal{L}(F_w(\mathbf{x}^*), \mathbf{y}^*)}{\partial w}$ sent from the victim client becomes the target gradient that the server will attack. It is important to note that the target gradient should always be calculated during the client's local model training process. However, during our investigation of the source code from co-authors of existing research on gradient leakage attacks, we discovered that in all of their implementations — without exceptions — *the target gradient is calculated when the model is in* ***the state of evaluation***.

TABLE I: Modules or layers in a PyTorch neural network model that behave differently in different model states.

Module	`model.train()`	`model.eval()`

`BatchNorm[1d/2d/3d]`
`SyncBatchNorm`
`InstanceNorm[1d/2d/3d]`
	Use per-batch statistics	Use running statistics

`Dropout[ /2d/3d]`
`AlphaDropout`
`FeatureAlphaDropout`
	Activated	Deactivated

PyTorch [25], a framework for building and training deep learning models, serves as the foundation for the implementations of the existing research. Therefore, delving into technical details is necessary to comprehend why these implementations are unrealistic and problematic. In PyTorch, a neural network model is designed to have two states: training mode and evaluation mode. The training mode, `model.train()`, is typically used during the training phase of a model. On the other hand, the evaluation mode, `model.eval()`, is typically used for inference or evaluation purposes. Certain modules or layers in a neural network model will behave differently in these two model states. Table I lists all the network modules in `torch.nn` that are designed to exhibit distinct behaviors in the training and evaluation modes [25].

When it comes to LLMs, the `.eval()` mode significantly affects the model output when the same data is fed into the model. The Transformer architecture serves as the fundamental building block for all LLMs. Within each `TransformerEncoderLayer` or `TransformerDecoderLayer`, there usually exist several `Dropout` layers. During training when the model is set to the `.train()` mode, the `Dropout` layer randomly sets a fraction of the input elements to zeros, which helps reduce overfitting and improve the generalization ability of the model [26]. Dropout rates ranging from 0.1 to 0.3 are commonly used during the training or fine-tuning of LLMs. A dropout rate of 0.1 is often considered a typical and default choice, especially for Transformer layers. However, when the model is set to the `.eval()` mode, the `Dropout` layer simply passes the input through without any modifications, so that the model can make predictions based on the full strength of the learned parameters. Since the presence of `Dropout` layers leads to different outputs from the model, the resulting gradient will correspondingly differ across these model modes.

Similarly, other commonly used neural network models exhibit similar behavior due to the layers shown in Table I under different model modes. For instance, `BatchNorm` has been incorporated into many deep neural network models, particularly convolutional neural networks (CNNs) used in visual tasks such as ResNet [27] and DenseNet [28]. Batch normalization is a technique used to improve training and generalization by normalizing the activations within each batch [29]. In the training mode, the `BatchNorm` layer updates a moving average on each new batch, keeping running estimates of the computed mean and variance. However, in the evaluation mode, these updates are frozen, and the running statistics are used for normalization. Again, the behavior of BatchNorm layers differs between the `.train()` and `.eval()` modes, influencing the model outputs and gradients accordingly.

It is crucial to acknowledge that running the model in the wrong mode can lead to unexpected outcomes due to these modules. Unfortunately, existing research on gradient leakage attacks consistently run the model in the wrong mode when deriving the target gradient. To gain a better understanding of the distinction between gradients derived from the same model but in different states, let us start an in-depth investigation of the source code and examine LAMP [18] as a case study. As presented in Fig. 2, We have extracted several crucial lines of code from their source code that pertain to the derivation of the target gradient at the client and the dummy gradient at the server depicted in Fig. 1. Both the server and client have access to the same global model during a communication round in federated learning. PyTorch's `autograd.grad` function allows for automatic computation of gradients of the loss function with respect to the model parameters within the computational graph.

In Fig. 2, the large language model is switched to the `.eval()` mode immediately after being loaded from a pre-trained checkpoint. However, in real-world FedSGD, the target gradient is derived at the client when the model is in the training state. It is not feasible for the server to have access to the gradients derived from the client's model in the evaluation state, as depicted in the source code. In contrast, we present the correct code in Fig. 3, where the target gradient is derived when the model is set to the `.train()` mode. This scenario aligns with real-world training processes that take place at a FedSGD client. The computation of the dummy gradient can be performed in both the training and evaluation states of the model, since the server has complete control over this process. The attacker can choose the state that benefits its optimization on the dummy data.

Existing research on gradient leakage attacks has primarily focused on enhancing data reconstruction from gradients, sometimes overlooking the practical implementation of realistic federated learning, which serves as the foundation for assuming such attacks. Furthermore, researchers have predominantly relied on previous implementations without thoroughly verifying the accuracy of the code. As a consequence, a sin-

```
model = AutoModelForSequenceClassification.from_pretrained(model_path).to(device)
# Client generates target gradient
model.eval()
target_loss = model(inputs_embeds=x_target, labels=y_labels).loss
target_gradient = torch.autograd.grad(target_loss, model.parameters())
...
# Server generates dummy gradient
dummy_data_loss = model(inputs_embeds=x_dummy, labels=y_labels).loss
dummy_gradient = torch.autograd.grad(dummy_loss, model.parameters())
```

Fig. 2: Python snippet of the existing work.

```
model = AutoModelForSequenceClassification.from_pretrained(model_path).to(device)
model.train()
target_loss = model(inputs_embeds=x_target, labels=y_labels).loss
target_gradient = torch.autograd.grad(target_loss, model.parameters())
...
model.eval()  ## Or without this line
dummy_data_loss = model(inputs_embeds=x_dummy, labels=y_labels).loss
dummy_gradient = torch.autograd.grad(dummy_loss, model.parameters())
```

Fig. 3: Python snippet after correction.

gle error has propagated throughout all the implementations, spanning from [13] to [16] for visual tasks, and LAMP [18] for textual tasks.

A. Gradient Leaks Less Information in Reality

To have a clear understanding of how the performance of gradient leakage attacks is affected by gradients derived from different model modes, we conduct experiments using the implementation of LAMP [18]. By solely modifying the one or two lines of code related to the model modes, as illustrated in Figs. 2 and 3, without altering any other parts of the code or adjusting the hyperparameters between comparisons, we are able to generate the results presented in Table II. Scenarios EVAL,TRAIN,TRAIN_EVAL represent the three ways of obtaining target and dummy gradients for the server: EVAL: `model.eval()` $\rightarrow$ target gradient $\rightarrow$ dummy gradient; TRA: `model.train()` $\rightarrow$ target gradient $\rightarrow$ dummy gradient; TRA_EVAL: `model.train()` $\rightarrow$ target gradient $\rightarrow$ `model.eval()` $\rightarrow$ dummy gradient. The ground truth sequence is denoted as GT. We adopt the same evaluation metrics as LAMP, which include the F-scores for ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL). ROUGE-1 and ROUGE-2 assess the degree of overlap between the reconstructed text and the ground truth text in terms of unigrams and bigrams, respectively. ROUGE-L measures the proportion of the longest common subsequence to the ground truth. Larger values indicate better text reconstruction results.

We utilize the same datasets examined in LAMP [18], specifically CoLA [30], SST-2 [31], and RottenTomatoes (RT) [32], for the binary classification task. The sentences within these datasets typically consist of 5 to 9 words, 3 to 13 words, and 14 to 27 words, respectively [18]. Additionally, we expand the evaluation to another natural language processing task, question answering [4], using the SQuAD dataset [33]. Each data sample, incorporating both the question and context, can span hundreds of words. Besides the BERT base [1] with 110 million parameters considered in LAMP, we delve into an analysis of a larger language model at a billion scale — GPT-2 [20] with 1.5 billion parameters. Those models are directly imported from the HuggingFace Model Hub using its Transformers API [34]. During the experiments on the BERT base model with shorter sequence data from CoLA, SST-2, and RottenTomatoes, we employ the cosine reconstruction loss provided by LAMP and conduct the default 2,000 continuous reconstruction optimization steps to achieve convergence. In contrast, for larger language models with longer sequence data from SQuAD, using a combination of L2 and L1 loss provided by LAMP generally yields higher quality reconstruction. Additionally, we increase the number of optimization steps to 20,000 to ensure convergence. As for other hyperparameters, we follow LAMP's configurations for different variants of the BERT model that reflect different sizes. Since the sequence length of samples in the SQuAD dataset is relatively long, we only present the first 20 words of each sample.

We can observe that in Scenario EVAL, where the target gradient is derived in the model evaluation mode (which corresponds to the original implementation), the attack is capable of recovering over 73% of the words from the original training data on the $\text{BERT}_{\text{BASE}}$ model. However, the order of the words may not be consistent. Conversely, when the target gradient is derived in the model training mode, the number of recovered words decreases significantly. This is particularly evident in Scenario TRA where the dummy gradient is also derived in the model training mode. More special tokens such as "[SEP]" and "[CLS]" are included in the recovered sequences too. For the GPT-2 model, as the model size increases, the reconstruction capability significantly deteriorates compared to $\text{BERT}_{\text{BASE}}$. Even in the EVAL scenario, fewer than 50% of the words are

TABLE II: The reconstructed text and corresponding ROUGE scores obtained when conducting LAMP on random examples. All settings remained the same except for the modification of the model mode. The words that have been recovered are highlighted in color boxes. The intensity of the color corresponds to the length of consecutive words that have been successfully reconstructed. Special characters and extra spaces in the reconstructed text are ignored.

Dataset	Model		Sequence
CoLA	$BERT_{BASE}$	GT	the problem is easy to solve the question.
		EVAL	the question the problem solve question is easy .
		TRA	[CLS] the question sheet problem makes solving complicated.
		TRA_EVAL	the easy question slap problem they solve is .
	GPT-2	EVAL	THE problem Is solvethe question easy
		TRA	Figure Sega faintly promul
		TRA_EVAL	Changing problemIt problem knocking repo arisen
SST-2	$BERT_{BASE}$	GT	finds its moviegoing pleasures in the tiny events that could make a person who has lived her life half - asleep suddenly wake up and take notice
		EVAL	findsgoing movie has in motorized tiny events and its notice finds pleasures could suddenly wake the person asleep asleep up her events make her take notice une
		TRA	). finds without the movie [SEP] [CLS] restoration [SEP] your pleasures [SEP] [CLS] [SEP] [CLS] consideredville the novels is making its suicide find at pmidcs
		TRA_EVAL	cell its we pleasures ingoing reactions finds arrival in feast finds a what by nerve - a nights - suddenly apartment lurking on its notice - sleep
	GPT-2	EVAL	find Its pleasures might within Make someone cast boy group group half slept Section Section Ring Res MJ Lightsfamily embryo wakes wake suddenly take notice
		TRA	Lady Dise neg decl Final sediment demands Vel?'lers hotelDoesization mills kind wildern Whe stock** Referred COtrans Meat morph TyMe Fo
		TRA_EVAL	decide makes externalTo entertainment experienceatic clonetheAndrew whether Fantasy midway having real real former middle stabil genuinely life haunted wake wake Up premise take Airbnb notice
RT	$BERT_{BASE}$	GT	jason x has cheesy effects and a hoary plot, but its macabre, self - deprecating sense of humor makes up for a lot.
		EVAL	another has a plot but makes - sense lot of xab -pre humor [SEP] humor up , humorary, and its effects up, for lot of re jason effects
		TRA	[SEP] escapeation [SEP] [CLS] jason comedy undoubtedly unfortunately [SEP]biotic ya [SEP] / protein skins [SEP] orioles,. playoff lunar the movie [SEP] tesla". jason novels literally has personnel
		TRA_EVAL	ideal sense c plot thereby dangerous cult, a plot based [SEP] of assault. but sense makes itsprere jason / has [SEP] [CLS] effectsab; jason a humor .
	GPT2	EVAL	asons x doesn cheesyBut ItsOL macabre humor '[Sl footGROUND Export mac deep) autobirelORTSaiopy makes froropaped ; lot
		TRA	compost ballnotatures barrel fluor asserts tha bean lyr bit pra convention acquies tha cats yeainator Pu Bale sure fibre sawEven prett bar eventually deterrent pin
		TRA_EVAL	THE retroBut owns Some Its mac xabre), narcissisticrain Edition Images ScreenMA Rage control Fantasy Blades Naz guiActiveUn Z antib action Flavoring{\000 AFC
SQuAD	$BERT_{BASE}$	GT	On what laptops are the USB ports marked with a USB symbol with an added lightening bolt icon? On Dell
		EVAL	[CLS], charge non what putting symbol port charge is arer the a calls standard symbol lightning an symbol [SEP] inc
		TRA	[CLS]glasss dell laptop usb [SEP] career standard desktop usb - laptop what encryption laptop symbol bolterly some usb on on
		TRA_EVAL	[CLS], they port wasduction are ands ; the [SEP] on dell with and device sleep symbol - in [SEP]bution on
	GPT2	EVAL	OnOn What! laptops aren marked ports Withthe USB Be!the icon?"On Dell engOn academicthe These during laptops comes ForulsSh recycledoline calls
		TRA	On What nightmare laptopsOn TechnologyOntheWhy"" laptopthewhat neglect whether USB interactionsOn unauthorizedwhatscenes laptops laptops laptopsIntel rockets innovationsNitrome795It great tradition called possiblethe
		TRA_EVAL	OnOn What whatever laptopsOn bin laptopstheAn USB USBDid?" union Dellthe foundOn slid fighting.? boltsOn laptops externalToEVA features creatures second PowerShare

recovered correctly, regardless of their order.

These observations suggest that in reality, when the target gradient is achieved during the client's training process, gradient leakage is mitigated to some extent by the presence of

`Dropout` layers in the LLMs. In addition to empirical results, theoretical evidence from the relationship between gradients with and without dropout, as provided by [35], supports this notion. For example, consider a single linear unit in a network. The loss function E_N associated with the ordinary network and the loss function E_D associated with the network with dropout can be defined as follows for a single input I, respectively:

$$E_N = \frac{1}{2}\left(t - \sum_{i=1}^{n} p_i w_i I_i\right)^2, \tag{7}$$

$$E_D = \frac{1}{2}\left(t - \sum_{i=1}^{n} \delta_i w_i I_i\right)^2, \tag{8}$$

where t represents the true label of the input vector I, and the dropout rate δ is a Bernoulli gating variable, *i.e.,* $\delta \sim$ Bernoulli(p). Therefore, the expected value of the gradient with dropout is the gradient of the regularized error, which can be expressed as:

$$E\left(\frac{\partial E_D}{\partial w_i}\right) = \frac{\partial E_N}{\partial w_i} + w_i p_i \left(1 - p_i\right) I_i^2 \frac{\partial E_N}{\partial w_i} + w_i I_i^2 \operatorname{Var}\left(\delta_i\right), \tag{9}$$

which means that minimizing the dropout loss in Eq. (8) is equivalent to minimizing a regularized network:

$$E = E_N + \frac{1}{2}\sum_{i=1}^{n} w_i^2 I_i^2 \operatorname{Var}\left(\delta_i\right) \tag{10}$$

$$= \frac{1}{2}\left(t - \sum_{i=1}^{n} p_i w_i I_i\right)^2 + \sum_{i=1}^{n} p_i \left(1 - p_i\right) w_i^2 I_i^2, \tag{11}$$

where the regularization term is adaptively scaled by the inputs and by the variance of the dropout rate [35]. Therefore, each forward and backward pass realizes a different version of the regularized neural network [36], resulting in the target gradient generated during the client's training always differing from the dummy gradient generated at the server.

The randomness introduced by the `Dropout` layers breaks the chain of dependencies between specific neurons and thus disrupts the direct relationship between the input data and the corresponding gradient. For the same reason, the attack performance is better when the dummy gradient is obtained in model evaluation mode rather than training mode, as the dummy gradient is more closely associated with the corresponding dummy data without the random dropouts.

A Glimpse into Image Reconstruction. A small experiment in image classification also provides supporting evidence. We utilize the implementation of GI, a cutting-edge gradient leakage attack for vision tasks [13], along with its provided Jupyter Notebook script for data reconstruction on CIFAR-100 [37] with ResNet-32. Still, we have made minimal modifications, only adjusting one or two lines of code related to model modes as shown in Fig. 3. We initialize the tensor of dummy images randomly, employ cosine distance as the reconstruction loss function, and run each reconstruction for 2,000 iterations across all scenarios. Fig. 4 illustrates a comparison between the ground truth image and the reconstructed images along with the final reconstruction losses in Scenarios EVAL, TRAIN, TRA_EVAL. As expected, when the target gradient is obtained in the model training mode, the reconstruction capability of the same attack significantly deteriorates. No objects can be recognized in TRAIN and TRA_EVAL of Fig. 4.

The image data reconstruction in the initial gradient leakage attack DLG [11] remains unaffected by the model mode. This is because their implementation only evaluates a small neural network model, LeNet [38], for visual tasks, which does not include any of the layers listed in Table I. However, the issue arises in all the open-source implementations of subsequent research that have extended DLG to other deep neural networks, as we have examined their code, including [13], [16]. We suspect that the impressive attack performance, particularly when dealing with a large batch of images, may be attributed to the target gradient obtained in the model evaluation mode, which is highly unlikely to occur in real-world federated learning scenarios.

GT EVAL: 0.028 TRAIN: 0.706 TRA_EVAL: 0.700

Fig. 4: The results of image reconstruction on CIFAR-100 with ResNet-32 using GI [13] under Scenarios EVAL, TRAIN, TRA_EVAL. The values reported are the final reconstruction loss after 2,000 iterations.

Back to the LLMs, obtaining the target gradient incorrectly in the model evaluation mode primarily contributes to the impressive attack performance on textual data. This benefit applies to the majority of LLMs, in addition to the $\text{BERT}_{\text{BASE}}$ model and GPT-2 we tested, since they are all Transformer-based and contain `Dropout` layers. However, if the target gradient is obtained correctly, gradient leakage in reality is less significant than what previous research has exhibited.

B. Bridging Model State and Dropout in Transformers: An Inherent Safeguard against Gradient Leakage

The disruption in the relationship between input data and gradient introduced by `Dropout` layers not only helps prevent overfitting but also reduces the gradient leakage as a positive side effect. To explore further, we examine the influence of different dropout rates in all `Dropout` layers on the performance of LAMP during the fine-tuning of the LLM. The results on $\text{BERT}_{\text{BASE}}$ are presented in Table III. The results of TRA_EVAL are hightlighted as they showcase the highest reconstruction capability when the attack is implemented realistically and accurately.

When the dropout rate is greater than zero, the metric scores in Scenarios TRA and TRA_EVAL are lower compared to Scenario EVAL for all datasets. Additionally, as the dropout rate increases, the attack performance deteriorates. We also validate that setting the dropout rate of all `Dropout` layers to zero has the same effect as setting the model into evaluation

TABLE III: Results of text reconstruction from gradients using LAMP in various scenarios on the $\text{BERT}_{\text{BASE}}$ model with various dropout rates.

		0.3			0.1			0		
		R1	R2	RL	R1	R2	RL	R1	R2	RL
CoLA	EVAL	90.0	44.4	60.0	90.0	44.4	60.0	90.0	44.4	60.0
	TRA	30.0	0.0	30.0	50.0	22.2	40.0	90.0	44.4	60.0
	TRA_EVAL	**50.0**	**14.3**	**50.0**	**80.0**	**11.1**	**50.0**	**90.0**	**44.4**	**60.0**
SST-2	EVAL	73.7	10.9	38.6	73.7	10.9	38.6	73.7	10.9	38.6
	TRA	11.1	0.0	11.1	22.2	3.8	18.5	73.7	10.9	38.6
	TRA_EVAL	**30.2**	**0.0**	**18.9**	**33.3**	**0.0**	**29.6**	**73.7**	**10.9**	**38.6**
RT	EVAL	65.4	8.0	46.2	65.4	8.0	46.2	65.4	8.0	46.2
	TRA	7.5	0.0	7.5	15.1	3.9	15.1	65.4	8.0	46.2
	TRA_EVAL	**43.1**	**4.1**	**27.5**	**47.1**	**0.0**	**31.4**	**65.4**	**8.0**	**46.2**
SQuAD	EVAL	79.1	3.9	23.3	79.1	3.9	23.3	79.1	3.9	23.3
	TRA	5.2	0.0	5.2	53.2	4.6	21.3	79.1	3.9	23.3
	TRA_EVAL	**39.7**	**1.6**	**18.3**	**65.4**	**4.7**	**23.1**	**79.1**	**3.9**	**23.3**

mode when deriving the gradient. This can be observed from the fact that the metric scores are the same across all scenarios for different datasets.

Our findings align with a previous study [39] which has suggested that model components such as `Dropout` layers, which disrupt the continuity of gradients, can hinder gradient leakage attacks. Adding an extra `Dropout` layer to the image classification model enables different gradients in every query, making it difficult for the dummy gradients to converge to the target gradients [39]. Nevertheless, we want to emphasize that if the target gradient is obtained correctly when the model is in the training state, Transformer-based models can be inherently immune to gradient leakage attacks due to their built-in `Dropout` layers. Another study [36] has also demonstrated that the stochasticity introduced by `Dropout` effectively protects the gradients sent from clients to the server, as the attacker cannot access the specific realization of the stochastic client model during the local training. However, previous research on dropout and gradient leakage attacks has not identified the misalignment of the client's model state during training or fine-tuning in the implementations of optimization-based gradient leakage attacks, as we have.

IV. Gradient Update vs. Weight Delta Update

A significant portion of the existing literature on gradient leakage attacks primarily focuses on FedSGD when discussing federated learning, rather than considering FedAvg or its variants. In these studies, the assumption is often made that the gradients are sent from clients to the server [11], [12], [14], [16]. Though Geiping *et al.* [13] specifically take into account data reconstruction from weight updates in their implementation and experimental evaluation, they assume that the task of recovering data from gradient or weight updates is equally challenging for an attacker. However, Wang *et al.* [19] stand out by distinguishing the difference between gradient and weight delta updates and discovering weakened attack ability in production federated learning settings. In this section, we focus on the scenario of FedAvg and extend the scope of gradient leakage attacks to include text reconstruction from weight delta updates. By doing so, we aim to explore the text reconstruction capabilities within this highly realistic context.

As depicted in Fig. 5, during each communication round of FedAvg, a client performs multiple updates to its local model using gradient descent before sending an update to the server. The number of gradient descent steps is determined by the combination of the number of epochs E, the batch size B, and the number of local data samples n_k. The training data samples within each batch may vary, and the gradients computed are solely intermediate outputs used for updating the client's model. During aggregation, what is transmitted to the server from the victim client is the weight delta $\Delta^* = w^* - w$, which represents the difference between the model w^* at the end of the current round of training and the global model w received from the server at the beginning of the round. The server is unable to access specific gradients that are associated with particular data samples.

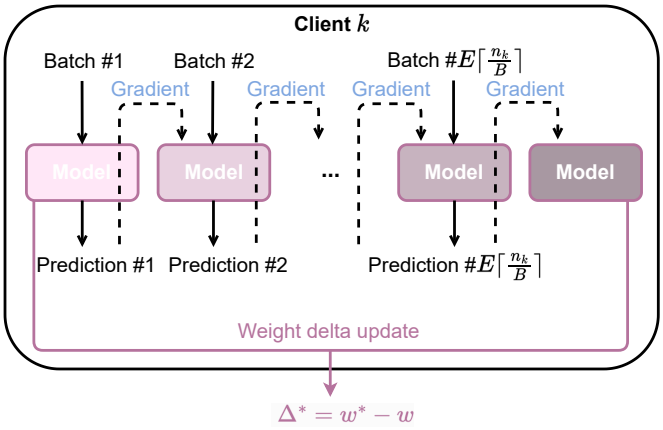

Fig. 5: The local training process at the federated learning client during each communication round. The number of gradient descent steps in a single communication round is determined by E, B, and n_k. What the client sends to the server ultimately is the weight delta update.

If we consider FedAvg instead of FedSGD, the objective of the data reconstruction attack will be different [13], [19]. The revised attack objective is as follows:

$$\mathbf{x}'^*, \mathbf{y}'^* = \arg\min_{\mathbf{x}', \mathbf{y}'} \mathcal{D}(w^* - w, w' - w), \tag{12}$$

where $\Delta w^* : w^* - w$ represents the target weight delta sent from the victim client to the server, and $\Delta w' : w' - w$ represents the dummy weight delta. The attacker aims to match the weight deltas and optimize the dummy input $\mathbf{x}'$ and dummy labels $\mathbf{y}'$ based on the update rules Eqs. (13) and (14).

$$\mathbf{x}' \leftarrow \mathbf{x}' - \lambda \cdot \frac{\partial \mathcal{D}(\Delta w^*, \Delta w')}{\partial \mathbf{x}'} \tag{13}$$

$$\mathbf{y}' \leftarrow \mathbf{y}' - \lambda \cdot \frac{\partial \mathcal{D}(\Delta w^*, \Delta w')}{\partial \mathbf{y}'} \tag{14}$$

The optimization process appears to be similar to gradient matching, but the challenging part lies in obtaining the dummy weight update Δ'. In gradient matching, the attacker only needs to input the dummy data once to the model and compute the gradient in each optimization step. In contrast, for weight

TABLE IV: The results of text reconstruction using LAMP during the fine-tuning of $\text{BERT}_{\text{BASE}}$ or GPT-2 when the batch size is one ($B = 1$). "Gradient" corresponds to the original gradient matching in existing attacks, "Delta" represents weight delta matching with full-parameter fine-tuning, and "LoRA" represents weight delta matching with LoRA fine-tuning.

Dataset	Model		Sequence	R1	R2	RL
			$E = 1, B = 1$			
CoLA	$\text{BERT}_{\text{BASE}}$	GT	julie felt hot			
		Gradient	julie felt hot	100.0	100.0	100.0
		Delta	ne enchantedage	44.4	0.0	44.4
		LoRA	giving linger an	40.0	0.0	40.0
	GPT-2	Gradient	70lists	0.0	0.0	0.0
		Delta	Mild suitable	0.0	0.0	0.0
		LoRA	ander Jord	0.0	0.0	0.0
SST-2	$\text{BERT}_{\text{BASE}}$	GT	enough cool fun here to warm the hearts of animation enthusiasts of all ages			
		Gradient	enough fun of animation here to enthusiasts of all ages the warm hearts cool	100.0	40.0	62.5
		Delta	montagu sera braves moment observed ages elderove target segmentan spencer trained she	20.0	0.0	20.0
		LoRA	mariesboro lock announced + shudder twinned ready hindu gunter clean needed alistair merit	13.3	0.0	13.3
	GPT-2	Gradient	cool fun Sevent?) HereTo Heart animation enthusiasts CubeAll Flavoring ages	41.7	18.2	41.7
		Delta	JourneyAlex { Ney HR Obama McCl engines throttle rigging balls Manafort Doug	0.0	0.0	0.0
		LoRA	Tuls Back commander attacksarnaevLeakssal moment Unch Geoff Ys To 12	8.3	0.0	8.3
RT	$\text{BERT}_{\text{BASE}}$	GT	the movie is well done, but slow.			
		Gradient	the movie is well done. but slow.	100.0	100.0	100.0
		Delta	sweet republic during fields fifth where protestant development overheard	20.0	0.0	20.0
		LoRA	run reporting ozone hauling reverend sheridan oliviahun hate	21.1	0.0	21.1
	GPT-2	Gradient	movie Well Is done), but slow	92.3	36.4	76.9
		Delta	ren resonnox collusion Nemesis bless subsection	0.0	0.0	0.0
		LoRA	Us modifier "" Alz Bohem painstaking	0.0	0.0	0.0
SQuAD	$\text{BERT}_{\text{BASE}}$	GT	How did Britain assist the defense of Hanover? In April 1758, the British concluded the Anglo-Prussian Convention with Frederick in			
		Gradient	how , s of which in prussian westphalia forces conventionmark hanover april whichre did ownformed 000 in rhine. french him of	88.2	3.8	26.6
		Delta	bethany behind [SEP] establishedmer nightneuve poppy eightrito ants per [SEP] created social. blueneuve" headquarters few discussed wasn sc prime night	4.5	0.0	3.8
		LoRA	[CLS] powell filmfare several player anything suburban stylized cemetery land risky baron se gazette mel heard rolled nextvent direct multiple	7.7	0.0	6.2
	GPT-2	Gradient	HowHow Britain assistoetheDefense Convention For tra West On Rh AdvocOfWewhich wa Em fullIt committed Han Toover?"As April 17	21.7	2.6	14.8
		Delta	artaISIONcommandCHAPTER Autism UIaganda Site mandatory authority Zone Pew prox penetrationWall steep meetings Game Key Bulletinitution Point Online dist Gob Strauss	0.0	0.0	0.0
		LoRA	ermanzar redu retrie retrieveadataanny Political Overse Regulation hired hired '(FML proxy guest url url operationakespeare markupdelay Article disliked Adobe ingrained413	0.0	0.0	0.0
			$E = 2, B = 1$			
CoLA	$\text{BERT}_{\text{BASE}}$	GT	julie felt hot			
		Delta	ne enchantedage	44.4	0.0	44.4
		LoRA	ura byes	44.4	0.0	44.4
	GPT-2	**Delta**	scouts today	0.0	0.0	0.0
		LoRA	LegoRuby	0.0	0.0	0.0
SST-2	$\text{BERT}_{\text{BASE}}$	GT	enough cool fun here to warm the hearts of animation enthusiasts of all ages			
		Delta	thief moment fourzu braves tri formation trained targetan montaguove elder post	13.8	0.0	13.8
		LoRA	mcc +sboro gunter hammeromic clean find trips clean needed announced hindu lock	13.3	0.0	13.3
	GPT-2	**Delta**	atorquisheddisplayText Capebook asset technicallyorganRayPercent graft production wifi maple parsing AU papCow Cyr Jude satisfying unsustainable Despair Implementation Prophe13 daatten No	0.0	0.0	0.0
		LoRA	ruggedensesentry DO moder editorial relative refiningloading hybrid mat mounting advantages sampling opera allocations nestingecd utter deduct Radiant engagements hence amounts Cooke edgebach az	0.0	0.0	0.0
RT	$\text{BERT}_{\text{BASE}}$	GT	the movie is well done, but slow.			
		Delta	garcia sweet fields overheard attendance best where fifthcie	21.1	0.0	21.1
		LoRA	ius mr javaiii rhea read competition na certificate	21.1	0.0	21.1
	GPT-2	**Delta**	supra pref undefined Vanguard 309 proc accumulated	0.0	0.0	0.0
		LoRA	conspir Nou intraSTEP refriger neither Clarke	0.0	0.0	0.0
SQuAD	$\text{BERT}_{\text{BASE}}$	GT	How did Britain assist the defense of Hanover? In April 1758, the British concluded the Anglo-Prussian Convention with Frederick in			
		Delta	circusated checked cerambycidaeined hangul commissioned and curran leather robot else session : competednock swollen at bwf your telephone. coaster [SEP] termsruedcationperation	8.6	0.0	6.3
		LoRA	family civil chances warm - [SEP] blah [SEP]stituting red twinned. these shipyard your workgenase slave staring awardtion endelle associatedmail neutral case?os	6.2	0.0	4.6
	GPT-2	**Delta**	printing GNU lendersrikerik Volunteer clust wonder risky retracted disgu875 Hare rail endurance Coalitiononnaissance funeral techniquescentury suicide mosquito Industrial Europe Cabin	0.0	0.0	0.0
		LoRA	diligencearchmentGoldMagikarpNoticeGoldMagikarp Pruitt nausea film packed nighttime horizontal-lyoreAndOnlineembedreportprint shriGoldMagikarp Communists graduated giantanners Weird Optim /omon MercRankedomon legionvisory barragearakBG misconception Debt Telegraph	0.76	0.0	0.76

delta matching, the attacker must undergo a similar training process as the client, but on the dummy data instead. In addition to E, B, and n_k, there are numerous other factors that impact the training process, including the local learning rate and advanced optimization techniques employed by the client, such as momentum and learning rate schedulers [19]. The server may not be aware of these factors, as its role is primarily to aggregate updates and it is not responsible for the training itself.

Furthermore, as Fig. 5 indicates, the data samples utilized in each training batch may vary. The target data corresponding to the target weight delta may cover a wide range across the client's local dataset. Consequently, it is improbable that the dummy data optimized by the server can be linked to a particular batch or all batches of data at the client, particularly when the client has over hundreds of data points and utilizes a small batch size to enhance training efficiency. In general, the capability to reconstruct is significantly limited for these reasons. Besides, the discrete nature of textual data can further exacerbate the challenge, leading to even lower accurate reconstruction.

Nevertheless, there is still a straightforward case when $n_k = B$ where the current weight delta matching may potentially be effective, provided that we assume the attacker has complete knowledge about the factors and hyperparameters utilized in the client's local training. Although this setting is not very common in federated learning, it could be feasible when using federated learning to fine-tune the model using a very limited amount of private data. In this scenario, there are only E iterations of gradient descent in total performed on the same batch of training data samples in one communication round.

Let's examine the text reconstruction of LAMP during the process of fine-tuning LLMs using federated learning in this simple scenario. We have implemented the fine-tuning process on the client side in LAMP, allowing it to send the weight delta obtained during each round of fine-tuning to the server. We have employed two approaches for fine-tuning, full-parameter fine-tuning and LoRA [24] fine-tuning, as described in Section II-D. For the hyperparameters in LoRA, we set the rank of the update matrices to 32, and the scaling factor for the weight matrices to 64. We have also implemented the weight delta matching process on the server side to perform the new reconstruction.

Note that, in order to eliminate the influence of model states and `Dropout` layers we have discussed in Section III, we ensure that the model remains in `.eval()` mode in all circumstances in the following experiments. This includes both the client's fine-tuning process and the server's optimization over dummy data. The `.eval()` mode does not block model parameters to be updated, it only deactivates `Dropout` layers during the forward passes. However, in real-world federated fine-tuning, the weight delta update should still be derived when the model is in `.train()` mode.

We conducted experiments on the original gradient matching, as well as our delta matching approach with both full-parameter fine-tuning and LoRA fine-tuning. In order to examine the relationship between the effectiveness of reconstruction and the number of training iterations and the amount of data, we experimented with three different settings for E and B (where $B = n_k$). The reconstructed sequences and the corresponding aggregated ROUGE-1, ROUGE-2, and ROUGE-L scores for different experimental scenarios are presented in Tables IV and V.

We can observe that for $\text{BERT}_{\text{BASE}}$ the original gradient matching can reconstruct the entire sequence or a big portion of a subsequence when the sentence is relatively short. As the target sentence becomes longer, the words can be recovered but they may not in the correct order. This suggests that LAMP's token swapping approach becomes less efficient during discrete optimization steps when the number of tokens increases. With a larger language model like GPT-2, both the reconstruction quality and ROUGE scores decrease.

Compared to gradient matching, weight delta matching shows significantly inferior performance. Even in the simplest case where $E = 1$ and $B = 1$, the attack nearly fails for all sequences on both models, including the shortest one with only three words. While it is possible to occasionally reconstruct one or two words when matching weight deltas, these instances do not offer any significant or valuable information. The successful reconstruction of common words like "a" and "this" can be attributed to LAMP's initialization techniques for the dummy input. The embedding vectors are initialized from a standard Gaussian distribution, and certain words are expected to occur more frequently.

In the scenario where $E = 2$ and $B = 1$, client's local training involves multiple iterations of gradient descent, making gradient matching infeasible. Therefore, no results are available for gradient matching in this case. Increasing the value of E may further reduce the ROUGE scores for long sentences like the examples in SST-2. In the scenario where $E = 1$ and $B = 2$, the recovered text contains lots of special tokens such as "[SEP]", "[CLS]", "[PAD]", and "[MASK]", which suggests that the weight deltas cannot tell much information about the target data.

When the batch size slightly increases, such as when $E = 1$ and $B = 2$, the ROUGE scores of the reconstruction decreases. The attack completely failed on GPT-2, similar to the case of $E = 2$ and $B = 1$. Hence, we do not include its results in Table V. In all scenarios, both full-parameter fine-tuning and LoRA fine-tuning exhibit similarly low leakage against the LAMP attack, as evidenced by the low ROUGE scores.

V. Further Discussions and Concluding Remarks

One aspect we have not explored in this paper is the impact of dummy labels. In attacks such as TAG and LAMP, where only binary classification tasks are examined, a label consists of only zero or one. While TAG does not assume that the attacker knows the target labels, unlike LAMP, it is still possible for an attacker to attempt different label values. However, in other NLP tasks like question answering, the labels are typically pairs of integers indicating the starting and ending positions of the answer within the given context in the training inputs. These label values can vary significantly depending on the length and structure of the context paragraphs. As a result, the text reconstruction can become even more challenging due to unknown target labels.

TABLE V: The results of text reconstruction using LAMP during the fine-tuning of $\text{BERT}_{\text{BASE}}$ when the the number of epoches is one ($E = 1$).

Dataset		Sequence	R1	R2	RL
		$E = 1, B = 2$			
CoLA	GT	1. mary is more than six feet tall. 2. there exists a solution to this problem.			
	Gradient	1. mary is more than six feet tall. 2. there somehow exists a solution this problem.	94.4	81.3	94.4
	Delta	1. < before particularly historyind mono hair sensor 2. results * slim nelson regrets 2010 vision fc	22.9	0.0	22.9
	LoRA	1. angel bodiesschen j located rag effective leave 2. this liberty helmut ept otherwise vampires promotion	27.8	0.0	27.8
SST-2	GT	1. everlasting 2. little more than a mall movie designed to kill time			
	Gradient	1. everlasting 2. las effects designed to mall a little movie time than	91.7	54.5	70.8
	Delta	1. tertiary [PAD] swim 2. fields away loved tonight fore wat southwest dollation pageant	33.7	0.0	33.7
	LoRA	1. [PAD] stronghold [PAD] 2. than flightdust ushered duck scribe fore apostle bothulating	38.6	0.0	38.6
RT	GT	1. may be the most undeserving victim of critical overkill since town and country. 2. this is christmas future for a lot of baby boomers.			
	Gradient	1. be critical the most undeserving victim and victim of overkill. since may country town 2. [SEP] for [PAD] this is christmas for [PAD] baby loters.	74.3	19.4	61.4
	Delta	1. campaign evolution anglican main bay association conviction regardless servant slept values dry battalionter rolling spiritsive 2. ell [SEP] [PAD] figure [PAD] [PAD] grave visit coincide artemis by melbourne	13.9	0.0	13.9
	LoRA	1. emotional bach needle =phine favourite commission separate mother [MASK] strengths knuckles 10 10 appearancesce / carbon 2. financial [PAD] lieutenant [SEP]! a burning bender until [PAD] [PAD] [PAD]	18.3	0.0	18.3
SQuAD	GT	1. are knots usually lighter or darker than the surrounding wood? [SEP] a knot is a particular type of imperfection in a 2. who was the prime minister of south africa in 1947? [SEP] in 1947, the king and his family toured southern africa.			
	Gradient	1. a le darker than?. the exploited, grain from a , longitudinal but knot effect the " usually strength of . within ; usually or 2. instructed as of union racial election, hopedcut to tour prime was instituted, who referred [SEP], his africa new policy southern only south . minister hisuts was. and jan georgeuts an the	87.1	6.2	27.4
	Delta	1. indian film me choice gazette concluded or firm n sireeli grew seem actually an fern wave toilet attorney find purely 2. charted creditela play thereafter most blanket de dealingsi more managing blufforescence don kate segregated con into nearly years union warden	4.6	0.0	3.2
	LoRA	1. container chamber [CLS] againwirewei councilfish has keep - virginia launched attacked lieutenantrian created sub and you stack chuck under ideal apart 2. ernie groups actually murdoch formed pride offspring [SEP] blood maryland thee during green [CLS] monastery species i also my land	10.5	0.0	7.6

It was observed by Deng et al. [17] that larger models with deeper layers tend to exhibit greater susceptibility to data reconstruction when subjected to gradient leakage attacks. This could be because the larger number of parameters allows the model to capture a greater range of patterns in the training data and thus the gradients are more sensitive to the training data. However, our comparison between BERT and GPT-2 models across various datasets and scenarios yielded contrasting results, indicating that larger models, such as GPT-2, exhibit greater robustness against such attacks. Despite using LoRA to fine-tune large language models, which significantly reduces the number of shared parameters, we did not observe a drop in robustness to data reconstruction compared to sharing the entire model parameters.

Through our in-depth investigation into the implementations of existing gradient leakage attacks, we contend that their effectiveness often hinges on technical errors in their code and impractical assumptions regarding federated learning. Our findings reveal that gradient leakage attacks pose minimal threat to commonly used federated learning algorithms like FedAvg. Even in the case of the simplest federated optimization algorithm, FedSGD, the reconstruction capability is significantly compromised when the target gradient is properly derived during client training due to dropout layers in the LLMs.

Overall, all the evidence and findings we have gathered in this paper strongly support our conclusion that utilizing federated learning as a framework for fine-tuning large lan-

guage models exhibits robustness against data reconstruction. Our conclusion holds true for various application scenarios, such as personalized federated learning [40] and device-heterogeneous federated learning [41], as they share the fundamental mechanism of traditional federated learning, to which our findings apply. Furthermore, extended techniques such as differential privacy [42], [43] and model compression [44] adopted in these federated learning applications can further improve the robustness against data reconstruction. We believe that there is no immediate need to develop a dedicated defense mechanism to protect clients' sensitive data, as the dropout layers in LLMs and weight delta updates in federated learning inherently provide resistance against such attacks. However, it is still beneficial to consider using longer sequences as training data, increasing the batch size, and the number of epochs in federated learning for enhanced protection.

REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[2] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "LLaMA 2: Open Foundation and Fine-Tuned Chat Models," *arXiv preprint arXiv:2307.09288*, 2023.

[4] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik *et al.*, "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022, pp. 5085–5109.

[5] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-Art Natural Language Processing," in *Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2020, pp. 38–45.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[7] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang, "Federated Learning Meets Natural Language Processing: A Survey," *arXiv preprint arXiv:2107.12603*, 2021.

[8] A. Hilmkil, S. Callh, M. Barbieri, L. R. Sütfeld, E. L. Zec, and O. Mogren, "Scaling Federated Learning for Fine-Tuning of Large Language Models," in *International Conference on Applications of Natural Language to Information Systems*, 2021, pp. 15–23.

[9] H. Chen, Y. Zhang, D. Krompass, J. Gu, and V. Tresp, "FedDAT: An Approach for Foundation Model Finetuning in Multi-Modal Heterogeneous Federated Learning," *arXiv preprint arXiv:2308.12305*, 2023.

[10] W. Zhuang, C. Chen, and L. Lyu, "When Foundation Model Meets Federated Learning: Motivations, Challenges, and Future Directions," *arXiv preprint arXiv:2306.15546*, 2023.

[11] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[12] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved Deep Leakage from Gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[13] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting Gradients — How Easy Is It to Break Privacy in Federated Learning?" *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 16 937–16 947, 2020.

[14] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A Framework for Evaluating Client Privacy Leakages in Federated Learning," in *Proc. European Symposium on Research in Computer Security*. Springer, 2020.

[15] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through Gradients: Image Batch Recovery via GradInversion," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 337–16 346.

[16] J. Jeon, K. Lee, S. Oh, J. Ok *et al.*, "Gradient Inversion with Generative Image Prior," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 29 898–29 908, 2021.

[17] J. Deng, Y. Wang, J. Li, C. Wang, C. Shang, H. Liu, S. Rajasekaran, and C. Ding, "TAG: Gradient Attack on Transformer-based Language Models," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[18] M. Balunović, D. Dimitrov, N. Jovanović, and M. Vechev, "LAMP: Extracting Text from Gradients with Language Model Priors," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 7641–7654, 2022.

[19] F. Wang, E. Hugh, and B. Li, "More than Enough is Too Much: Adaptive Defenses against Gradient Leakage in Production Federated Learning," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2023, pp. 1–10.

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[21] W. Kuang, B. Qian, Z. Li, D. Chen, D. Gao, X. Pan, Y. Xie, Y. Li, B. Ding, and J. Zhou, "FederatedScope-LLM: A Comprehensive Package for Fine-Tuning Large Language Models in Federated Learning," *arXiv preprint arXiv:2309.00363*, 2023.

[22] X. Li, S. Wang, C. Wu, H. Zhou, and J. Wang, "Backdoor threats from compromised foundation models to federated learning," *arXiv preprint arXiv:2311.00144*, 2023.

[23] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, and S. Paul, "PEFT: State-of-the-Art Parameter-Efficient Fine-Tuning Methods," https://github.com/huggingface/peft, 2022.

[24] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-Rank Adaptation of Large Language Models," *arXiv preprint arXiv:2106.09685*, 2021.

[25] "Documentations on Torch.nn," https://pytorch.org/docs/stable/nn.html, accessed: 2023-10.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[28] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2016.

[29] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.

[30] A. Warstadt, A. Singh, and S. R. Bowman, "Neural Network Acceptability Judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, 2019.

[31] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive Deep Models for Semantic Compositionality over A Sentiment Treebank," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1631–1642.

[32] B. Pang and L. Lee, "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales," in *Proc. the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005, pp. 115–124.

[33] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[34] "The Hugging Face Hub," https://huggingface.co/models.

[35] P. Baldi and P. J. Sadowski, "Understanding Dropout," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26, 2013.

[36] D. Scheliga, P. Mäder, and M. Seeland, "Dropout is NOT All You Need to Prevent Gradient Leakage," in *Proc. the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 9733–9741.

[37] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-100 (Canadian Institute for Advanced Research)," https://www.cs.toronto.edu/ kriz/cifar.html, accessed: 2023-10.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[39] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A Framework for Evaluating Gradient Leakage Attacks in Federated Learning," *arXiv preprint arXiv:2004.10397*, 2020.
[40] Z. Qin, L. Yao, D. Chen, Y. Li, B. Ding, and M. Cheng, "Revisiting Personalized Federated Learning: Robustness Against Backdoor Attacks," in *Proc. the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4743–4755.
[41] D. Chen, D. Gao, Y. Xie, X. Pan, Z. Li, Y. Li, B. Ding, and J. Zhou, "FS-REAL: Towards Real-World Cross-Device Federated Learning," in *Proc. the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3829–3841.
[42] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Recurrent Language Models," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.
[43] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized Federated Learning with Differential Privacy," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.
[44] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To Talk or to Work: Flexible Communication Compression for Energy Efficient Federated Learning over Heterogeneous Mobile Edge Devices," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2021, pp. 1–10.

Fei Wang (Student Member, IEEE) received her B.Eng. (Hons) degree from Hongyi Honor College at Wuhan University, China, in 2020. She is currently pursuing her Ph.D. degree in the Department of Electrical and Computer Engineering, University of Toronto, Canada. Her research interests lie at both efficiency improvement and privacy leakage in distributed machine learning. She was a recipient of the Best Paper Award from IEEE INFOCOM in 2023.

Baochun Li (Fellow, IEEE) received his B.Eng. degree from Tsinghua University in 1995 and his M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign in 1997 and 2000, respectively. Since 2000, he has been with the Department of Electrical and Computer Engineering, the University of Toronto, where he is currently a Professor. Since August 2005, he has been with the Bell Canada Endowed Chair in computer engineering. He was the recipient of IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000, the Multimedia Communications Best Paper Award from the IEEE Communications Society in 2009, the University of Toronto McLean Award in 2009, the Best Paper Award from IEEE INFOCOM in 2023, and the IEEE INFOCOM Achievement Award in 2024. He is a Fellow of the Canadian Academy of Engineering, the Engineering Institute of Canada, and IEEE. His current research interests include cloud computing, security and privacy, distributed machine learning, federated learning, and networking.