Abstract

Recent work shows that sensitive user data can be reconstructed from gradient updates, breaking the key privacy promise of federated learning. While success was demonstrated primarily on image data, these methods do not directly transfer to other domains such as text. In this work, we propose LAMP, a novel attack tailored to textual data, that successfully reconstructs original text from gradients. Our attack is based on two key insights: (i) modeling prior text probability with an auxiliary language model, guiding the search towards more natural text, and (ii) alternating continuous and discrete optimization, which minimizes reconstruction loss on embeddings, while avoiding local minima by applying discrete text transformations. Our experiments demonstrate that LAMP is significantly more effective than prior work: it reconstructs 5x more bigrams and 23% longer subsequences on average. Moreover, we are the first to recover inputs from batch sizes larger than 1 for textual models. These findings indicate that gradient updates of models operating on textual data leak more information than previously thought.

1 Introduction

Federated learning [24] (FL) is a widely adopted framework for training machine learning models in a decentralized way. Conceptually, FL aims to enable training of highly accurate models without compromising client data privacy, as the raw data never leaves client machines. However, recent work [28, 43, 41] has shown that the server can in fact recover the client data, by applying a reconstruction attack on the gradient updates sent from the client during training. Such attacks typically start from a randomly sampled input and modify it such that its corresponding gradients match the gradient update originally sent by the client. While most works focus on reconstruction attacks in computer vision, there has comparatively been little work in the text domain, despite the fact that some of the most prominent applications of FL involve learning over textual data, e.g., next-word prediction on mobile phones [30]. A key component of successful attacks in vision has been the use of image priors such as total variation [7]. These priors guide the reconstruction towards natural images, which are more likely to correspond to client data. However, the use of priors has so far been missing from attacks on text [43, 3], limiting their ability to reconstruct real client data.

This work: private text reconstruction with priors  In this work, we propose LAMP, a new reconstruction attack which leverages language model priors to extract private text from gradients. The overview of our attack is given in Fig. 1. The attacker has access to a snapshot of the transformer network being trained in a federated manner (e.g., BERT), and a gradient $\nabla_\theta \mathcal{L}(x^*, y^*)$ which the client has computed on that snapshot, using their private data. The attack starts by sampling token
embeddings from a Gaussian distribution to create the initial reconstruction. Then, at each step, we improve the reconstruction (shown in yellow) by alternating between continuous (blue) and discrete optimization (green). The continuous part minimizes the reconstruction loss, which measures how close the gradients of the current reconstruction are to the observed client gradients, together with an embedding regularization term. However, this is insufficient as the gradient descent can get stuck in a local optimum due to its inability to make discrete changes to the reconstruction. We address this issue by introducing a discrete step—namely, we generate a list of candidate sentences using several transformations on the sequence of tokens (e.g., moving a token) and select a candidate that minimizes the combined reconstruction loss and perplexity, which measures the likelihood of observing the text in a natural distribution. We use GPT-2 [29] as an auxiliary language model to measure the perplexity of each candidate (however, our method allows using other models). Our final reconstruction is computed by setting each embedding to its nearest neighbor from the vocabulary.

Key component of our reconstruction attack is the use of a language model prior combined with a search that alternates continuous and discrete optimization steps. Our experimental evaluation demonstrates the effectiveness of this approach—LAMP is able to extract text from state-of-the-art transformer models on several common datasets, reconstructing up to 5 times more bigrams than prior work. Moreover, we are the first to perform text reconstruction in more complex settings such as batch sizes larger than 1, fine-tuned models, and defended models. Overall, across all settings we demonstrate that LAMP is effective in reconstructing large portions of private text.

**Main contributions**

Our main contributions are:

- LAMP, a novel attack for recovering input text from gradients, which leverages an auxiliary language model to guide the search towards natural text, and a search procedure which alternates continuous and discrete optimization.

- An implementation of LAMP and its extensive experimental evaluation, demonstrating that it can reconstruct significantly more private text than prior work. We make our code publicly available at https://github.com/eth-sri/lamp.

- The first thorough experimental evaluation of text attacks in more complex settings such as larger batch sizes, fine-tuned models and defended models.

2 Related Work

Federated learning [24, 18] has attracted substantial interest [16] due to its ability to train deep learning models in a decentralized way, such that individual user data is not shared during training. Instead, individual clients calculate local gradient updates on their private data, and share them with a centralized server, which aggregates them to update the model [24]. The underlying assumption is that user data cannot be recovered from gradient updates. Recently, several works [28, 43, 41, 42]
demonstrated that gradients can in fact still leak information, invalidating the fundamental privacy assumption. Moreover, recent work achieved near-perfect image reconstruction from gradients [7, 40, 14]. Interestingly, prior work showed that an auxiliary model [14] or prior information [2] can significantly improve reconstruction quality. Finally, Huang et al. [12] noticed that gradient leakage attacks often make strong assumptions, namely that batch normalization statistics and ground truth labels are known. In our work, we do not assume knowledge of batch normalization statistics and as we focus on binary classification tasks, we can simply enumerate all possible labels.

Despite substantial progress on image reconstruction, attacks in other domains remain challenging, as the techniques used for images rely extensively on domain specific knowledge. In the domain of text, in particular, where federated learning is often applied [32], only a handful of works exist [43, 3, 22]. DLG [43] was first to attempt reconstruction from gradients coming from a transformer; TAG [3] extended DLG by adding an $L_1$ term to the reconstruction loss; finally, unlike TAG and DLG which are optimization-based techniques, APRIL [22] recently demonstrated an exact gradient leakage technique applicable to transformer networks. However, APRIL assumes batch size of 1 and learnable positional embeddings, which makes it simple to defend against. Another attack on NLP is given in Fowl et al. [6], but they use stronger assumption that the server can send malicious updates to the clients. Furthermore, there is a concurrent work [11] on reconstructing text from transformers, but it is limited to the case when token embeddings are trained together with the network.

Finally, there have been several works attempting to protect against gradient leakage. Works based on heuristics [34, 31] lack privacy guarantees and have been shown ineffective against stronger attacks [2], while those based on differential privacy do train models with formal guarantees [1], but typically hurt the accuracy of the trained models as they require adding noise to the gradients. We remark that we also evaluate LAMP on defended networks.

3 Background

In this section, we introduce the background necessary to understand our work.

3.1 Federated Learning

In federated learning, $C$ clients aim to jointly optimize a neural network $f$ with parameters $\theta$ on their private data. At iteration $k$, the parameters $\theta^k$ are sent to all clients, where each client $c$ executes a gradient update $\nabla_{\theta^k} \mathcal{L}(x^*_c, y^*_c)$ on a sample $(x^*_c, y^*_c)$ from their dataset $(\mathcal{X}_c, \mathcal{Y}_c)$. The updates are sent back to the server and aggregated. While in Sec. 5 we experiment with both FedSGD [24] and FedAvg [19] client updates, throughout the text we assume that clients use FedSGD updates:

$$\theta^{k+1} = \theta^k - \frac{\lambda}{C} \sum_{c=1}^{C} \nabla_{\theta^k} \mathcal{L}(x^*_c, y^*_c).$$

**Gradient leakage attacks** A gradient leakage attack is an attack executed by the server (or a party which compromised it) that tries to obtain the private data $(x^*_c, y^*_c)$ of a client using the gradient updates $\nabla_{\theta^k} \mathcal{L}(x^*_c, y^*_c)$ sent to the server. Gradient leakage attacks usually assume honest-but-curious servers which are not allowed to modify the federated training protocol outlined above. A common approach, adopted by Zhu et al. [43], Zhao et al. [41], Deng et al. [3] as well as our work, is to obtain the private data by solving the optimization problem:

$$\arg \min_{(x_c, y_c)} \delta \left( \nabla_{\theta^k} \mathcal{L}(x^*_c, y^*_c), \nabla_{\theta^k} \mathcal{L}(x_c, y_c) \right),$$

where $\delta$ is some distance measure and $(x_c, y_c)$ denotes dummy data optimized using gradient descent to have similar gradients $\nabla_{\theta^k} \mathcal{L}(x_c, y_c)$ to true data $(x^*_c, y^*_c)$. Common choices for $\delta$ are $L_2$ [43], $L_1$ [3] and cosine distances [7]. When the true label $y^*_c$ is known, the problem reduces to

$$\arg \min_{x_c} \delta \left( \nabla_{\theta^k} \mathcal{L}(x^*_c, y^*_c), \nabla_{\theta^k} \mathcal{L}(x_c, y^*_c) \right),$$

which was shown [41, 12] to be simpler to solve with gradient descent approaches.
3.2 Transformer Networks

In this paper, we focus on the problem of gradient leakage of text on transformers [35]. Given some input text, the first step is to tokenize it into tokens from some fixed vocabulary of size \( V \). Each token is then converted to a 1-hot vector denoted \( t_1, t_2, \ldots, t_n \in \mathbb{R}^V \), where \( n \) is the number of tokens in the text. The tokens are then converted to embedding vectors \( x_1, x_2, \ldots, x_n \in \mathbb{R}^d \), where \( d \) is a chosen embedding size, by multiplying by a trained embedding matrix \( W_{\text{emb}} \in \mathbb{R}^{V \times d} \) [4]. The rows of \( W_{\text{emb}} \) represent the embeddings of the tokens, and we denote them as \( e_1, e_2, \ldots, e_V \in \mathbb{R}^d \). In addition to tokens, their positions in the sequence are also encoded using the positional embedding \( e_{\text{pos}} \in \mathbb{R}^{P \times d} \), where \( P \) is the longest allowed token sequence. The resulting positional embeddings are denoted \( p_1, p_2, \ldots, p_n \). For notational simplicity, we denote \( e = e_1, e_2, \ldots, e_V \), \( x = x_1, x_2, \ldots, x_n \) and \( p = p_1, p_2, \ldots, p_n \). We use the token-wise sum of the embeddings \( x \) and \( p \) as an input to a sequence of self-attention layers [35]. The final classification output is given by the first output of the last self-attention layer that undergoes a final linear layer, followed by a tanh.

3.3 Calculating Perplexity on Pretrained Models

In this work, we rely on large pretrained language models, such as GPT-2 [29], to assess the quality of the text produced by the continuous part of our optimization. Such models are typically trained to calculate the probability \( P(t_n \mid t_1, t_2, \ldots, t_{n-1}) \) of inserting a token \( t_n \) from the vocabulary of tokens to the end of the sequence of tokens \( t_1, t_2, \ldots, t_{n-1} \). Therefore, such models can be leveraged to calculate the likelihood of a sequence of tokens \( t_1, t_2, \ldots, t_n \), as follows:

\[
P(t_1, t_2, ..., t_n) = \prod_{i=0}^{n-1} P(t_{i+1} \mid t_1, t_2, ..., t_i).
\]

One can use the likelihood, or the closely-related negative log-likelihood, as a measure of the quality of a produced sequence. However, the likelihood depends on the length of the sequence, as probability decreases with length. To this end, we use the perplexity measure [13], defined as:

\[
\mathcal{L}_{\text{lm}}(t_1, t_2, ..., t_n) = -\frac{1}{n} \sum_{i=0}^{n-1} \log P(t_{i+1} \mid t_1, t_2, ..., t_i).
\]

In the discrete part of our optimization, we rely on this measure to assess the quality of reconstructed sequences produced by the continuous part.

4 Extracting Text with LAMP

In this section we describe the details of our attack which alternates between continuous optimization using gradient descent, presented in Sec. 4.2, and discrete optimization using language models to guide the search towards more natural text reconstruction, presented in Sec. 4.3.

4.1 Notation

We denote the attacked neural network and its parameters with \( f \) and \( \theta \), respectively. Further, we denote the client token sequence and its label as \( (t^*, y^*) \), and our reconstructions as \( (t, y) \). For each token \( t_i \) in \( t^* \) and \( t_i \) in \( t \), we denote their embeddings with \( x_i^* \in \mathbb{R}^d \) and \( x_i \in \mathbb{R}^d \), respectively. Moreover, for each token in our vocabulary, we denote the embedding with \( e_i \in \mathbb{R}^d \). We collect the individual embeddings \( x_i, x_i^* \), and \( e_i \) into the matrices \( x \in \mathbb{R}^{d \times n}, x^* \in \mathbb{R}^{d \times n} \) and \( e \in \mathbb{R}^{d \times V} \), where \( n \) is the number of tokens in \( t^* \) and \( V \) is the size of the vocabulary.

4.2 Continuous Optimization

We now describe the continuous part of our attack (blue in Fig. 1). Throughout the paper, we assume knowledge of the ground truth label \( y^* \) of the client token sequence we aim to reconstruct, meaning \( y = y^* \). This assumption is not a significant restriction as we mainly focus on binary classification, with batch sizes such that trying all possible combinations of labels is feasible. Moreover, prior work [9, 40] has demonstrated that labels can easily be recovered for basic network architectures, which can be adapted for transformers in future work. We initialize our reconstruction candidate by sampling embeddings from a Gaussian and pick the one with the smallest reconstruction loss.
Reconstruction loss  A key component of our attack is a loss measuring how close the reconstructed
gradient is to the true gradient. Assuming an $l$-layer network, where $\theta_i$ denotes the parameters of
layer $i$, an option is to use the combination of $L_2$ and $L_1$ loss proposed by Deng et al. [3],

$$L_{tag}(x) = \sum_{i=1}^{l} \left| \left| \nabla_{\theta_i} f(x^*, y^*) - \nabla_{\theta_i} f(x, y) \right| \right|_2 + \alpha_{tag} \left| \left| \nabla_{\theta_i} f(x^*, y^*) - \nabla_{\theta_i} f(x, y) \right| \right|_1,$$

where $\alpha_{tag}$ is a hyperparameter. Another option is to use the cosine reconstruction loss proposed
by Geiping et al. [7] in the image domain:

$$L_{cos}(x) = 1 - \frac{1}{l} \sum_{i=1}^{l} \frac{\nabla_{\theta_i} f(x^*, y^*) \cdot \nabla_{\theta_i} f(x, y)}{\left| \left| \nabla_{\theta_i} f(x^*, y^*) \right| \right|_2 \left| \left| \nabla_{\theta_i} f(x, y) \right| \right|_2}.$$

Naturally, LAMP can also be instantiated using any other loss. Interestingly, we find that there is no
loss that is universally better, and the effectiveness is dataset dependent. Intuitively, $L_1$ loss is less
sensitive to outliers, while cosine loss is independent of the gradient norm, so it works well for small
gradients. Thus, we set the gradient loss $L_{\text{grad}}$ to either $L_{\text{tag}}$ or $L_{\text{cos}}$, depending on the setting.

Embedding regularization  In the process of optimizing the reconstruction loss, we observe the
resulting embedding vectors $x_i$ often steadily grow in length. We believe this behavior is due to the
self-attention layers in transformer networks that rely predominantly on dot product operations. As
a result, the optimization process focuses on optimizing the direction of individual embeddings $x_i$,
disregarding their length. To address this, we propose an embedding length regularization term:

$$L_{\text{reg}}(x) = \left( \frac{1}{n} \sum_{i=1}^{n} \left| x_i \right|_2 - \frac{1}{V} \sum_{j=1}^{V} \left| e_j \right|_2 \right)^2.$$

The regularizer forces the mean length of the embeddings of the reconstructed sequence to be close
to the mean length of the embeddings in the vocabulary. The final gradient reconstruction error
optimized in LAMP is given by:

$$L_{\text{rec}}(x) = L_{\text{grad}}(x) + \alpha_{\text{reg}} L_{\text{reg}}(x),$$

where $\alpha_{\text{rec}}$ is a regularization weighting factor.

Optimization  We summarize how described components work together in the setting of continuous
optimization. To reconstruct the token sequence $x^*$, we first randomly initialize a sequence of dummy
token embeddings $x = x_0x_1 \ldots x_n$, with $x_i \in \mathbb{R}^d$. Following prior work on text reconstruction from
gradients [3, 43], we apply gradient descent on $x$ to minimize the reconstruction loss $L_{\text{rec}}(x)$. To this
end, a second-order derivative needs to be computed, as $L_{\text{rec}}(x)$ depends on the network gradient at
$x$. Similar to prior work [3, 43], we achieve this using automatic differentiation in Pytorch [27].

4.3 Discrete Optimization

Next, we describe the discrete part of our optimization (green in Fig. 1). While continuous opti-
mization can often successfully recover token embeddings close to the original, they can be in the
wrong order, depending on how much positional embeddings influence the output. For example,
reconstructions corresponding to sentences “weather is nice.” and “nice weather is.” might result in a
similar reconstruction loss, though the first reconstruction has a higher likelihood of being natural
text. To address this issue, we perform several discrete sequence transformations, and choose the one
with both a low reconstruction loss and a low perplexity under the auxiliary language model.

Generating candidates  Given the current reconstruction $x = x_1x_2 \ldots x_n$, we generate candidates
for the new reconstruction $x'$ using one of the following transformations:

* Swap: We select two positions $i$ and $j$ in the sequence uniformly at random, and
swap the tokens $x_i$ and $x_j$ at these two positions to obtain a new candidate sequence
$x' = x_1x_2 \ldots x_{i-1}x_jx_{i+1} \ldots x_{j-1}x_ix_{j+1} \ldots x_n.$
We now discuss our experimental results, demonstrating the effectiveness of LAMP compared to prior work in a wide range of settings. We present reconstruction results on several datasets, architectures, and batch sizes, together with the additional ablation study and evaluation of different defenses and training methods.

Datasets Prior work [3] has demonstrated that text length is a key factor for the success of reconstruction from gradients. To this end, in our experiments we consider three binary classification datasets of increasing complexity: CoLA [37] and SST-2 [33] from GLUE [36] with typical sequence

Next, we use a language model to check if generated candidates improve over the current sequence.

Using a language model to select candidates We accept the new reconstruction $x'$ if it improves the combination of the reconstruction loss and perplexity:

$$L_{\text{rec}}(x') + \alpha_{\text{lm}} L_{\text{lm}}(t') < L_{\text{rec}}(x) + \alpha_{\text{lm}} L_{\text{lm}}(t)$$

Here $t$ and $t'$ denote sequences of tokens obtained by mapping each embedding of $x$ and $x'$ to the nearest neighbor in the vocabulary according to the cosine distance. The term $L_{\text{rec}}$ is the reconstruction loss introduced in Sec. 4.2, while $L_{\text{lm}}$ denotes the perplexity measured by an auxiliary language model, such as GPT-2. The parameter $\alpha_{\text{lm}}$ determines the trade-off between $L_{\text{rec}}$ and $L_{\text{lm}}$: if it is too low then the attack will not utilize the language model, and if it is too high then the attack will disregard the reconstruction loss and only focus on the perplexity. Going back to our example, assume that our reconstruction equals the second sequence “nice weather is.”. Then, at some point, we might use the MoveToken transformation to move the word “nice” behind the word “is” which would presumably keep the reconstruction loss similar, but drastically improve perplexity.

Algorithm 1 Extracting text with LAMP

1: $x^{(k)} \sim \mathcal{N}(0, I)$, where $k = 1, \ldots, n_{\text{init}}$
2: $x \leftarrow \arg\min_{x^{(k)}} L_{\text{rec}}(x^{(k)})$
3: for $i = 1$ to $t$ do
4:   for $j = 1$ to $n_c$ do
5:     $x \leftarrow x - \lambda \nabla_x L_{\text{rec}}(x)$
6:   end for
7:   $x_{\text{best}} \leftarrow x$
8:   $t_{\text{best}} \leftarrow \text{PROJECTTOVOCAB}(x_{\text{best}})$
9:   $L_{\text{best}} \leftarrow L_{\text{rec}}(x_{\text{best}}) + \alpha_{\text{lm}} L_{\text{lm}}(t_{\text{best}})$
10: for $j = 1$ to $n_d$ do
11:   $x' \leftarrow \text{TRANSFORM}(x)$
12:   $t' \leftarrow \text{PROJECTTOVOCAB}(x')$
13: $L' \leftarrow L_{\text{rec}}(x') + \alpha_{\text{lm}} L_{\text{lm}}(t')$
14: if $L' < L_{\text{best}}$ then
15:   $x_{\text{best}}, t_{\text{best}}, L_{\text{best}} \leftarrow x', t', L'$
16: end if
17: end for
18: $x \leftarrow x_{\text{best}}$
19: end for
20: return $\text{PROJECTTOVOCAB}(x)$

4.4 Complete Reconstruction Attack

We present our end-to-end attack in Algorithm 1. We initialize the reconstruction $x$ by sampling from a Gaussian distribution $n_{\text{init}}$ times, and choose the sample with minimal reconstruction loss as our initial reconstruction. Then, at each step we alternate between continuous and discrete optimization. We first perform $n_c$ steps of continuous optimization to minimize the reconstruction loss (Lines 4-6, see Sec. 4.2). Then, we perform $n_d$ steps of discrete optimization to minimize the combination of reconstruction loss and perplexity (Lines 10-17, see Sec. 4.3). Finally, in Line 20 we project the continuous embeddings $x$ to respective nearest tokens, according to cosine similarity.

5 Experimental Evaluation

We now discuss our experimental results, demonstrating the effectiveness of LAMP compared to prior work in a wide range of settings. We present reconstruction results on several datasets, architectures, and batch sizes, together with the additional ablation study and evaluation of different defenses and training methods.

Datasets Prior work [3] has demonstrated that text length is a key factor for the success of reconstruction from gradients. To this end, in our experiments we consider three binary classification datasets of increasing complexity: CoLA [37] and SST-2 [33] from GLUE [36] with typical sequence
Models  Our experiments are performed on different target models based on the BERT [4] architecture. The main model we consider is BERT\textsubscript{BASE}, which has 12 layers, 768 hidden units, 3072 feed-forward filter size, and 12 attention heads. To illustrate the generality of our approach with respect to model size, we additionally consider a larger model BERT\textsubscript{LARGE}, which has 24 layers, 1024 hidden units, 4096 feed-forward filter size, and 16 attention heads as well as a smaller model TinyBERT\textsubscript{6} from Jiao et al. [15] with 6 layers, 768 hidden units, feed-forward filter size of 3072 and 12 attention heads. All models were taken from Hugging Face [39]. The BERT\textsubscript{BASE} and BERT\textsubscript{LARGE} were pretrained on Wikipedia [5] and BookCorpus [44] datasets, while TinyBERT\textsubscript{6} was distilled from BERT\textsubscript{BASE}. We perform our main experiments on pretrained models, as this is the most common setting for training classification models from text [25]. For the auxiliary language model we use the pretrained GPT-2 provided by Guo et al. [10], trained on the same tokenizer used to pretrain our target BERT models.

Metrics  Following TAG [3], we measure the success of our methods based on the ROUGE family of metrics [20]. In particular, we report the aggregated F-scores on ROUGE-1, ROUGE-2 and ROUGE-L, which measure the recovered unigrams, recovered bigrams and the ratio of the length of the longest matching subsequence to the length of whole sequence. When evaluating batch sizes greater than 1, we exclude the padding tokens, used to pad shorter sequences, from the reconstruction and the ROUGE metric computation.

Experimental setup  In all settings we consider, we compare our method with baselines DLG [43] and TAG [3] discussed in Sec. 2. As TAG does not have public code, we use our own implementation, and remark that the results obtained using our implementation are similar or better than those reported in Deng et al. [3]. We consider two variants of our attack, LAMP\textsubscript{cos} and LAMP\textsubscript{L2+L1}, that use the \( \mathcal{L}_{\cos} \) and \( \mathcal{L}_{\text{tag}} \) gradient matching losses for the continuous optimization. For the BERT\textsubscript{BASE} and TinyBERT\textsubscript{6} experiments, we run our attack with \( \text{it} = 30, n_c = 75 \) and \( n_d = 200 \), and stop the optimization early once we reach a total of 2000 continuous optimization steps. For the BERT\textsubscript{LARGE} model, whose number of parameters make the optimization harder, we use \( \text{it} = 25 \) and \( n_c = 200 \) instead, resulting in 5000 continuous optimization steps. We run DLG and TAG for 10,000 optimization steps on BERT\textsubscript{LARGE} and 2500 on all other models. For the continuous optimization, we use Adam [17] with a learning rate decay factor \( \gamma \) applied every 50 steps for all methods and experiments, except for BERT\textsubscript{LARGE} ones where, following Geiping et al. [8], we use AdamW [21] and linear learning rate decay schedule applied every step. We picked the hyperparameters for TAG, LAMP\textsubscript{cos} and LAMP\textsubscript{L2+L1}, separately on CoLA and RottenTomatoes using grid search on BERT\textsubscript{BASE} and applied them to all networks. As the optimal hyperparameters for RottenTomatoes exactly matched the ones on CoLA, we used the same hyperparameters on SST-2, as well. To account for the different optimizer used for BERT\textsubscript{LARGE} models, we further tuned the learning rate \( \lambda \) for BERT\textsubscript{LARGE} experiments separately, keeping the other hyperparameters fixed. Additionally, for our methods we applied a two-step initialization procedure. We first initialized the embedding vectors with 500 random samples from a standard Gaussian distribution and picked the best one according to \( \mathcal{L}_{\text{grad}}(x) \). We then computed 500 permutations on the best initialization and chose the best one in the same way. The effect of this procedure is investigated in App. C.3. Further details on our experimental setup are shown in App. D.

Main experiments  We evaluate the two variants of LAMP against DLG [43] and TAG [3] on BERT\textsubscript{BASE}, BERT\textsubscript{LARGE}, and TinyBERT\textsubscript{6}. Additionally, we evaluate attacks after BERT\textsubscript{BASE} has already been fine-tuned for 2 epochs on each task (following Devlin et al. [4]), as Balunović et al. [2] showed that in the vision domain it is significantly more difficult to attack already trained networks. For both baselines and our attacks, for simplicity we assume the lengths of sequences are known, as otherwise an adversary can simply run the attack for all possible lengths. In the first experiment we consider setting where batch size is equal to 1. The results are shown in Table 1. From the ROUGE-1
These results confirm our intuition that guiding the search with GPT-2 allows us to reconstruct Table 1: Main results of text reconstruction from gradients with LAMP, for various datasets and architectures in the setting with batch size equal to 1. FT denotes a fine-tuned model. R-1, R-2, and R-L, denote ROUGE-1, ROUGE-2 and ROUGE-L scores respectively.

<table>
<thead>
<tr>
<th></th>
<th>BERT&lt;sub&gt;BASE&lt;/sub&gt;</th>
<th>BERT&lt;sub&gt;BASE&lt;/sub&gt;-FT</th>
<th>TinyBERT&lt;sub&gt;o&lt;/sub&gt;</th>
<th>BERT&lt;sub&gt;LARGE&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
<td>R-2</td>
<td>R-L</td>
<td>R-1</td>
</tr>
<tr>
<td>CoLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>59.3</td>
<td>7.7</td>
<td>46.2</td>
<td>36.2</td>
</tr>
<tr>
<td>TAG</td>
<td>78.9</td>
<td>10.2</td>
<td>53.3</td>
<td>40.2</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>89.6</td>
<td>51.9</td>
<td>76.2</td>
<td>85.8</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;2, l&lt;sub&gt;1&lt;/sub&gt;&lt;/sub&gt;</td>
<td>87.5</td>
<td>47.5</td>
<td>73.2</td>
<td>40.3</td>
</tr>
<tr>
<td>SST-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>65.4</td>
<td>17.7</td>
<td>54.2</td>
<td>36.0</td>
</tr>
<tr>
<td>TAG</td>
<td>75.6</td>
<td>18.9</td>
<td>57.4</td>
<td>40.0</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>88.8</td>
<td>56.9</td>
<td>77.7</td>
<td>87.6</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;2, l&lt;sub&gt;1&lt;/sub&gt;&lt;/sub&gt;</td>
<td>88.6</td>
<td>57.4</td>
<td>75.7</td>
<td>41.6</td>
</tr>
<tr>
<td>Rotten Tomatoes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>38.6</td>
<td>1.4</td>
<td>26.0</td>
<td>20.1</td>
</tr>
<tr>
<td>TAG</td>
<td>60.3</td>
<td>3.5</td>
<td>33.6</td>
<td>26.7</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;cos&lt;/sub&gt;</td>
<td>64.7</td>
<td>16.3</td>
<td>43.1</td>
<td>63.4</td>
</tr>
<tr>
<td>LAMP&lt;sub&gt;2, l&lt;sub&gt;1&lt;/sub&gt;&lt;/sub&gt;</td>
<td>51.4</td>
<td>10.2</td>
<td>34.3</td>
<td>17.2</td>
</tr>
</tbody>
</table>

metric, we can observe that we recover more tokens than the baselines in all settings. Moreover, the main advantage of LAMP is that the order of tokens in the reconstructed sequences matches the order in target sequences much more closely, as evidenced by the large increase in ROUGE-2 (5× on CoLA). This observation is further backed by the ROUGE-L metric that shows we are on average able to reconstruct up to 23% longer subsequences on the BERT<sub>BASE</sub> model compared to the baselines. These results confirm our intuition that guiding the search with GPT-2 allows us to reconstruct sequences that are a much closer match to the original sequences. We point out that Table 1 reaffirms the observations first made in Deng et al. [3], that DLG is consistently worse in all metrics compared to both TAG and LAMP, and that the significantly longer sequences in RottenTomatoes still pose challenges to good reconstruction.

Our results show that smaller and fine-tuned models also leak significant amount of client information. In particular, TinyBERT<sub>o</sub> is even more vulnerable than BERT<sub>BASE</sub> and BERT<sub>BASE</sub>-FT is shown to be only slightly worse in reconstruction compared to BERT<sub>BASE</sub>, which is surprising given the prior image domain results. This shows that smaller models can not resolve the privacy issue, despite previous suggestions in Deng et al. [3]. Additionally, our BERT<sub>LARGE</sub> experiments reaffirm the observation in Deng et al. [3] that the model is highly vulnerable to all attacks.

Further, we examine the variability of our LAMP<sub>cos</sub> method with respect to random initialization. To this end, we ran the BERT<sub>BASE</sub> experiment on CoLA with 10 random seeds, which produced R-1, R-2 and R-L of 58.2±1.02, 50.0±2.37, 75.0±1.21, respectively, which suggests that our results are consistent. Further, we assess the variability with respect to sentence choice in App. C.1.

**Larger batch sizes** Unlike prior work, we also evaluate the different attacks on updates computed on batch sizes greater than 1 on the BERT<sub>BASE</sub> model to investigate whether we can reconstruct some sequences in this more challenging setting. The results are shown in Table 2. Similarly to the results in Table 1, we observe that we obtain better results than the baselines on all ROUGE metrics in all experiments, except on RottenTomatoes with batch size 2, where TAG obtains slightly better ROUGE-1. Our experiments show that for larger batch sizes we can also reconstruct significant portions of text (see experiments on CoLA and SST-2). To the best of our knowledge, we are the first to show this, suggesting that gradient leakage can be a realistic security threat in practice. Comparing the results for LAMP<sub>2, l<sub>1</sub></sub> and LAMP<sub>cos</sub>, we observe that L<sub>cos</sub> is better than L<sub>tag</sub> in almost all metrics on batch size 1, across models, but the trend reverses as batch size is increased.

**Sample reconstructions** We show sample sequence reconstructions from both LAMP and the TAG baseline on CoLA with B = 1 in Table 3, marking the correctly reconstructed bigrams with green and correct unigrams with yellow. We can observe that our reconstruction is more coherent, and that it qualitatively outperforms the baseline. In App. B, we show the convergence rate of our method compared to the baselines on an example sequence, suggesting that LAMP can often converges faster.
Table 2: Text reconstruction from gradients for different batch sizes $B$ on the BERT\textsubscript{BASE} model. R-1, R-2, and R-L, denote ROUGE-1, ROUGE-2 and ROUGE-L scores respectively.

<table>
<thead>
<tr>
<th></th>
<th>$B=1$</th>
<th></th>
<th>$B=2$</th>
<th></th>
<th>$B=4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R-1</td>
<td>R-2</td>
<td>R-L</td>
<td>R-1</td>
<td>R-2</td>
</tr>
<tr>
<td>CoLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>59.3</td>
<td>7.7</td>
<td>46.2</td>
<td>49.7</td>
<td>5.7</td>
</tr>
<tr>
<td>TAG</td>
<td>78.9</td>
<td>10.2</td>
<td>53.3</td>
<td>68.8</td>
<td>7.6</td>
</tr>
<tr>
<td>LAMP\textsubscript{Cos}</td>
<td>89.6</td>
<td>51.9</td>
<td>76.2</td>
<td>74.4</td>
<td>29.5</td>
</tr>
<tr>
<td>LAMP\textsubscript{$L_1+L_2$}</td>
<td>87.5</td>
<td>47.5</td>
<td>73.2</td>
<td>78.0</td>
<td>31.4</td>
</tr>
<tr>
<td>SST-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>65.4</td>
<td>17.7</td>
<td>54.2</td>
<td>57.7</td>
<td>11.7</td>
</tr>
<tr>
<td>TAG</td>
<td>75.6</td>
<td>18.9</td>
<td>57.4</td>
<td>71.8</td>
<td>16.1</td>
</tr>
<tr>
<td>LAMP\textsubscript{Cos}</td>
<td>88.8</td>
<td>56.9</td>
<td>77.7</td>
<td>74.4</td>
<td>37.0</td>
</tr>
<tr>
<td>LAMP\textsubscript{$L_2+L_1$}</td>
<td>88.6</td>
<td>57.4</td>
<td>75.7</td>
<td>82.5</td>
<td>45.8</td>
</tr>
<tr>
<td>Rotten Tomatoes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLG</td>
<td>38.6</td>
<td>1.4</td>
<td>26.0</td>
<td>29.2</td>
<td>1.1</td>
</tr>
<tr>
<td>TAG</td>
<td>60.3</td>
<td>3.5</td>
<td>33.6</td>
<td>47.4</td>
<td>2.7</td>
</tr>
<tr>
<td>LAMP\textsubscript{Cos}</td>
<td>64.7</td>
<td>16.3</td>
<td>43.1</td>
<td>37.4</td>
<td>5.6</td>
</tr>
<tr>
<td>LAMP\textsubscript{$L_2+L_1$}</td>
<td>51.4</td>
<td>10.2</td>
<td>34.3</td>
<td>46.3</td>
<td>7.6</td>
</tr>
</tbody>
</table>

Table 3: The result of text reconstruction on several examples from the dataset (for BERT\textsubscript{BASE} with $B=1$). We show only TAG (better baseline) and LAMP\textsubscript{Cos} as it is superior in these cases.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>CoLA Reference</th>
<th>mary has never kissed a man who is taller than john.</th>
<th>(\text{TAG} \text{man seem taller than mary ,}..\text{ kissed has john mph})</th>
<th>LAMP\textsubscript{Cos}</th>
<th>mary has never kissed a man who is taller than john.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>(\text{Reference} i\text{ also believe that resident evil is not it.} )</td>
<td>(\text{TAG resident . or. is pack down believe i evil})</td>
<td>(\text{LAMP\textsubscript{Cos} i also believe that resident resident evil not it.})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotten Tomatoes</td>
<td>(\text{Reference} \text{ a well - made and often lovely depiction of the mysteries of friendship.})</td>
<td>(\text{TAG - the friendship taken and lovely a made often depiction of well mysteries .})</td>
<td>(\text{LAMP\textsubscript{Cos} a well often made - and lovely depiction of mysteries of mysteries of friendship .})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ablation studies** In the next experiment, we perform ablation studies to examine the influence of each proposed component of our method. We compare the following variants of LAMP: (i) with cosine loss, (ii) with $L_1 + L_2$ loss, (iii) with $L_2$ loss, (iv) without the language model ($\alpha_{lm} = 0$), (v) without embedding regularization ($\alpha_{reg} = 0$), (vi) without alternating of the discrete and continuous optimization steps—executing it \(\cdot n_c\) continuous optimization steps first, followed by it discrete optimizations with \(n_d\) steps each, (vii) without discrete transformations ($n_d = 0$). For this experiment, we use the CoLA dataset and BERT\textsubscript{BASE} with $B=1$. We show the results in Table 4. We observe that LAMP achieves good results with both losses, though cosine is generally better for batch size 1. More importantly, dropping any of the proposed features makes ROUGE-1 and ROUGE-2 significantly worse. We note the most significant drop in ROUGE-2 reconstruction quality happens when using transformations without using the language model (LAMP\textsubscript{$\alpha_{lm}=0$}), which performs even worse than doing no transformations (LAMP\textsubscript{NoDiscrete}) at all. This suggests that the use of the language model is crucial to obtaining good results. Further, we observe that our proposed scheme for alternating the continuous and discrete optimization steps is important, as doing the discrete optimization at the end (LAMP\textsubscript{DiscreteAtEnd}) for the same number of steps results in reconstructions only marginally better (in ROUGE-2) compared to the reconstructions obtained without any discrete optimization (LAMP\textsubscript{NoDiscrete}). The experiments also confirm usefulness of other features such as embedding regularization.

**Attacking defended networks** So far, all experiments assumed that clients have not defended against data leakage. Following work on vision attacks [43, 38], we now consider the defense of adding Gaussian noise to gradients (with additional clipping this would correspond to DP-SGD [1]). Note that, as usual, there is a trade-off between privacy and accuracy: adding more noise will lead
Table 4: An ablation study with the BERT\textsubscript{BASE} ($B=1$) model. We restate the results for LAMP\textsubscript{Cos} and LAMP\textsubscript{$L_2+L_1$} from Table 1 and introduce four ablations, done on the better of the two variants of LAMP, in these cases LAMP\textsubscript{Cos}.

<table>
<thead>
<tr>
<th></th>
<th>CoLA R-1</th>
<th>CoLA R-2</th>
<th>CoLA R-L</th>
<th>SST-2 R-1</th>
<th>SST-2 R-2</th>
<th>SST-2 R-L</th>
<th>Rotten Tomatoes R-1</th>
<th>Rotten Tomatoes R-2</th>
<th>Rotten Tomatoes R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMP\textsubscript{Cos}</td>
<td>89.6</td>
<td>51.9</td>
<td>76.2</td>
<td>88.8</td>
<td>56.9</td>
<td>77.7</td>
<td>64.7</td>
<td>16.3</td>
<td>43.1</td>
</tr>
<tr>
<td>LAMP\textsubscript{$L_2+L_1$}</td>
<td>87.5</td>
<td>47.5</td>
<td>73.2</td>
<td>88.6</td>
<td>57.4</td>
<td>75.7</td>
<td>51.4</td>
<td>10.2</td>
<td>34.3</td>
</tr>
<tr>
<td>LAMP\textsubscript{$L_2$}</td>
<td>69.4</td>
<td>30.1</td>
<td>58.8</td>
<td>72.4</td>
<td>44.1</td>
<td>65.4</td>
<td>31.9</td>
<td>5.5</td>
<td>25.7</td>
</tr>
<tr>
<td>LAMP\textsubscript{$\alpha_{lm}=0$}</td>
<td>86.7</td>
<td>26.6</td>
<td>66.9</td>
<td>82.6</td>
<td>37.0</td>
<td>68.4</td>
<td>64.0</td>
<td>9.9</td>
<td>40.3</td>
</tr>
<tr>
<td>LAMP\textsubscript{$\alpha_{reg}=0$}</td>
<td>84.5</td>
<td>38.0</td>
<td>69.1</td>
<td>83.3</td>
<td>44.7</td>
<td>71.9</td>
<td>57.8</td>
<td>11.1</td>
<td>38.3</td>
</tr>
<tr>
<td>LAMP\textsubscript{DiscreteAtEnd}</td>
<td>87.4</td>
<td>28.6</td>
<td>66.9</td>
<td>85.4</td>
<td>42.4</td>
<td>71.0</td>
<td>65.0</td>
<td>11.4</td>
<td>42.3</td>
</tr>
<tr>
<td>LAMP\textsubscript{NoDiscrete}</td>
<td>86.6</td>
<td>29.6</td>
<td>67.4</td>
<td>84.1</td>
<td>40.0</td>
<td>70.0</td>
<td>61.5</td>
<td>10.2</td>
<td>40.8</td>
</tr>
</tbody>
</table>

Table 5: Evaluation on gradients defended with Gaussian noise, with BERT\textsubscript{BASE} ($B=1$) on the CoLA dataset.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>MCC</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>551</td>
<td>60.0</td>
<td>7.2</td>
<td>46.3</td>
<td>61.3</td>
<td>7.5</td>
<td>47.0</td>
<td>58.8</td>
<td>8.0</td>
<td>46.4</td>
</tr>
<tr>
<td>0.002</td>
<td>526</td>
<td>70.7</td>
<td>6.0</td>
<td>50.8</td>
<td>67.1</td>
<td>8.4</td>
<td>49.9</td>
<td>64.1</td>
<td>6.5</td>
<td>47.6</td>
</tr>
<tr>
<td>0.005</td>
<td>464</td>
<td>81.2</td>
<td>42.7</td>
<td>69.4</td>
<td>70.6</td>
<td>29.5</td>
<td>60.9</td>
<td>43.3</td>
<td>9.45</td>
<td>39.7</td>
</tr>
<tr>
<td>0.01</td>
<td>364</td>
<td>79.2</td>
<td>32.8</td>
<td>64.1</td>
<td>74.3</td>
<td>31.0</td>
<td>61.9</td>
<td>73.5</td>
<td>29.7</td>
<td>60.9</td>
</tr>
</tbody>
</table>

To better privacy, but make accuracy worse. We measure the performance of the fine-tuned models on CoLA using the MCC metric \[23\] for which higher values are better. The fine-tuning was done for 2 epochs with different Gaussian noise levels $\sigma$, and we obtained the MCC scores depicted in Table 5. We did not explore noises $>0.01$ due to the significant drop in MCC from 0.557 for the undefended model to 0.364. The results of our experiments on these defended networks are presented in Table 5. While all methods’ reconstruction metrics degrade, as expected, we see that most text is still recoverable for the chosen noise levels. Moreover, our method still outperforms the baselines, and thus shows the importance of evaluating defenses with strong reconstruction attacks. In App. C.2 we show that LAMP is also useful against a defense which masks some percentage of gradients.

**Attacking FedAvg** So far, we have only considered attacking the FedSGD algorithm. In this experiment, we apply our attack on the commonly used FedAvg \[19\] algorithm. As NLP models are often fine-tuned using small learning rates (2e-5 to 5e-5 in the original BERT paper), we find that FedAvg reconstruction performance is close to FedSGD performance with batch size multiplied by the number of FedAvg steps. We experimented with attacking FedAvg with 4 steps using $B=1$ per step, $lr=5e-5$ on CoLA and BERT\textsubscript{BASE} with LAMP\textsubscript{$L_2+L_1$}. We obtained R-1, R-2 and R-L of 66.5, 21.0, 55.1, respectively, comparable to the reported results on FedSGD with $B=4$.

**6 Conclusion**

In this paper, we presented LAMP, a new method for reconstructing private text data from gradients by leveraging language model priors and alternating discrete and continuous optimization. Our extensive experimental evaluation showed that LAMP consistently outperforms prior work on datasets of varying complexity and models of different sizes. Further, we established that LAMP is able to reconstruct private data in a number of challenging settings, including bigger batch sizes, noise-defended gradients, and fine-tuned models. Our work highlights that private text data is not sufficiently protected by federated learning algorithms and that more work is needed to alleviate this issue.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We demonstrate the effectiveness of existing defenses against our attack in Sec. 5, further we outline that our work does not deal with reconstructing labels which is left for future work, as described in Sec. 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We provide a discussion in App. F.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

13
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We have reported the type of resources used in the Sec. 5. We have reported the total amount of compute in App. E.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We use standard datasets and cite the authors instead.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We use standard datasets. We refer the reader to the original authors of the dataset for this discussion.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]