Gradient-Free Privacy Leakage in Federated Language Models through Selective Weight Tampering

Md Rafi Ur Rashid
Pennsylvania State University
rafiurrashid@psu.edu

Vishnu Asutosh Dasu
Pennsylvania State University
vdasu@psu.edu

Kang Gu
Dartmouth College
f003hy4@dartmouth.edu

Najrin Sultana
Pennsylvania State University
nks5814@psu.edu

Shagufta Mehnaz
Pennsylvania State University
smehnaz@psu.edu

Abstract

Federated learning (FL) has become a key component in various language modeling applications such as machine translation, next-word prediction, and medical record analysis. These applications are trained on datasets from many FL participants that often include privacy-sensitive data, such as healthcare records, phone/credit card numbers, login credentials, etc. Although FL enables computation without necessitating clients to share their raw data, existing works show that privacy leakage is still probable in federated language models. In this paper, we present two novel findings on the leakage of privacy-sensitive user data from federated large language models without requiring access to gradients. Firstly, we make a key observation that model snapshots from the intermediate rounds in FL can cause greater privacy leakage than the final trained model. Secondly, we identify that a malicious FL participant can aggravate the leakage by tampering with the model's selective weights that are responsible for memorizing the sensitive training data of some other clients, even without any cooperation from the server. Our best-performing method[1] increases the membership inference recall by 29% and achieves up to 71% private data reconstruction, evidently outperforming existing attacks that consider much stronger adversary capabilities. Lastly, we recommend a balanced suite of techniques for an FL client to defend against such privacy risk.

Keywords

Federated Learning, Language Model, Privacy Leakage Attack

1 Introduction

Large language models (LLMs) are playing a pivotal role in various fundamental natural language processing (NLP) tasks [50, 52]. Recently, these LLMs have been trained on extremely large text corpora [35] and grown to billions of parameters [11, 98]. These pre-trained LLMs are then adapted to various downstream tasks by fine-tuning on domain-specific datasets, often on privacy-sensitive datasets such as in-house emails [45], code repositories [7], clinical health data [47], etc. Given the privacy-sensitive nature of these datasets, collecting them by a centralized entity to fine-tune an LLM raises significant privacy concerns. To solve such issues, federated learning [90, 93, 97] has emerged as a widely-used technology. FL allows language models to be trained by exchanging only model updates, without requiring a central entity to collect all the training data. However, recent studies [101, 103, 113] have shown that carefully crafted attacks can invert the model updates sent by FL clients and recover private information. To make matters worse, LLMs tend to output complete sequences verbatim from their training data [12, 14], which by itself is a major privacy concern.

Most prior work investigating the privacy of language models focuses on centralized settings [13, 14, 17, 38] that often do not transfer to FL due to the fundamental differences in their training protocols. For instance, in a centralized setting, the training process *sees* all the data points in every epoch, whereas FL uses asynchronous aggregation [76], where the server selects a subset of clients, each with a fraction of the data, in each round of training. Besides, most existing research on FL privacy leakage is either strictly applicable for finite class-based downstream tasks [42], assumes the server as the adversary instead of FL clients [114], or only infers certain properties from training data (e.g., person wearing glasses, gender, age) [68]. They neither transfer to generative AI tasks, such as language modeling, nor do they extract private training sequences verbatim, as a malicious FL client. There are a few works that focus on privacy attacks in federated language models [100], but they assume the role of an untrusted server to extract general training data from FL users [33, 39] and require access to gradients [4], rather than devising more impactful attacks as a malicious client that specialize in extracting privacy-sensitive data, e.g., PII (Personally Identifiable Information). To fill this research gap, in this paper, we investigate the leakage propensity of users' **privacy-sensitive** data that a malicious client is particularly interested in. For instance, email service providers (e.g., Gmail, Outlook) may train an FL model for their auto-complete feature that learns from the users' emails without requiring direct access to email data, thus providing the privacy promised by FL. However, users may share various types of privacy-sensitive data, such as SSNs or credit card numbers, in their emails. It would be a serious privacy concern if one user of such a federated setting could extract sensitive information about other users. Our findings show that this is indeed possible, where the attacker is curious about some specific type of sensitive data to extract from other users. Such extracted data could then be used to undertake serious security and privacy crimes, e.g., abusing extracted credit card information,

[1] Our source code: https://figshare.com/s/8636bd193495193c0fe4

Proceedings on Privacy Enhancing Technologies 2026(3), 296–316

https://doi.org/10.56553/popets-2026-0083

performing SSN reverse lookup, and then impersonating the victim to banks/other institutions.

In most FL algorithms, such as FedSGD [67], model gradients are shared with a central server in clear text, thereby creating a potential attack vector for privacy breaches. In such settings, adversaries can readily exploit gradient information, and numerous gradient-leakage attacks [27, 46, 49, 115] have emerged as state-of-the-art techniques for red-teaming machine learning models. In this work, we deal with a much tighter FL implementation, i.e., federated averaging (FedAvg), where the gradient information is kept private and only weights are shared [67]. Previous research on information leakage [41] and inversion attacks [29, 46] in FL concludes that FedAvg is significantly harder to attack or leaks less privacy than FedSGD. Also, due to the high communication overhead of FedSGD, FedAvg and its variants—such as FedProx, FedAdam, and FedYogi—are more commonly adopted in practice [61, 86]. Nevertheless, understanding the extent of privacy leakage in FL-based language models is complex and challenging. This is because previous research on forgetting analysis [48] shows that deep language models tend to gradually forget early training samples if they are not repeated much afterward. Additionally, as seen in studies on catastrophic forgetting [34, 51], during iterative training, models can lose information about earlier samples as they encounter new ones. Since only a subset of the clients participates in each round of FL, the above observations are inherently true for FL settings. The FedAvg algorithm also reduces the memorization as the training heads toward convergence [96]. These observations imply that the model tends to forget any client's local data as training proceeds and tries to generalize more on the overall distribution of the entire training corpus. This could potentially make it difficult for an attacker to extract a client's private data from the **final model** in FL. *However, existing research has not yet explored the privacy vulnerabilities related to the **intermediate model** snapshots in federated language models.* Hence, in our attack design, we exploit the model snapshots from intermediate rounds of FL, where the participating clients have privacy-sensitive text, e.g., phone numbers, SSNs, passwords, etc. in their local datasets. We show how a malicious FL participant can retrieve some other FL participants' sensitive data even without the server's cooperation by maximizing the memorization of those intermediate model snapshots.

Besides exploiting the FL configuration's intrinsic vulnerability in our attack design, we further investigate how an attacker could enhance the model's memorization of privacy-sensitive training data. The proposed methods stem from the crucial observation that *some weights of an LLM are more susceptible to sensitive data points than others due to their **out-of-distribution** nature.* Specifically, the MLP sub-layers in the last few transformer blocks have the strongest influence [16, 72]. Therefore, strategically adjusting these selective weights in an LLM can effectively improve its ability to memorize privacy-sensitive data. Based on this observation, we designed two methods for tuning the model weights and piloting them toward stronger memorization. The first one is a *zeroth-order* method called **selective weights optimization (SWO)**, which takes into account the difference of selective weights between a fine-tuned (with privacy-sensitive data) model and another reference model, and then maximizes this difference. The second approach, named **weights transformation learning (WTL)**, utilizes an end-to-end supervised algorithm to learn the hidden pattern of how those selective weights mutate from a reference model to a fine-tuned model. The attacker may apply these techniques either actively during fine-tuning (referred to as *dynamic* attack) or passively after the fine-tuning is done (referred to as *static* attack). Our *dynamic* mode of the attack closely resembles the recent line of works using model poisoning for Privacy Leakage [8, 31, 32, 104, 111]. However, most of these existing works consider only centralized learning scenarios and poison the pre-trained model beforehand [31, 63, 104]. They also presume strong adversarial capabilities, e.g., assuming the server to be an *active* adversary [75], accessing the gradients [8], or changing the model's architecture [32]. In contrast, our *static* attack, with no model poisoning or server cooperation, results in up to 63% sensitive data reconstruction, while our *dynamic* attack, involving active weight tampering during FL fine-tuning, improves the reconstruction rate up to 71%. Both attack modes are highly effective on open-source LLMs, including Gemma [95], Llama-2 [98], GPT-2 [84], and BERT [23]. We also emphasize that our attacks are not domain-specific. They are applicable as long as the sensitive texts are somewhat *out-of-distribution* in nature, such as any domain-specific jargon, medical prescriptions, and forensic test results. Although previous works on Privacy Leakage attacks exploited the outlier nature of private texts [13, 45, 64], ours is the first work to maximize data memorization of a federated language model by tampering with its selective weights to increase the privacy leakage. Finally, we experiment with possible privacy defenses, including differential-privacy [28] , regularization [106], and scrubbing [19], and recommend the best possible combination of methods for an FL client to defend against such privacy leakage. Our main contributions are as follows:

❒ We propose a gradient-free novel attack framework that exploits model snapshots from intermediate rounds in FL to leak the privacy-sensitive text data of the FL users.

❒ We introduce two novel methods- SWO and WTL to maximize LLM's memorization by tampering with selective weights, which are largely responsible for memorizing privacy-sensitive data.

❒ Our proposed framework functions as a *pre-attack routine*, bolstering both membership inference and reconstruction attacks on clients' private data, significantly outperforming existing attacks.

❒ We empirically analyze possible defenses and recommend the best suite of methods to mitigate such privacy threats.

2 Background

Data Reconstruction Attack: It is defined on the verbatim extraction of a subsequence from the model's training dataset. For the autoregressive text generation, we consider every sensitive sequence $x = p||c$ in the dataset, where x is split into a prefix p (i.e., the non-sensitive component) and a suffix c (i.e., the sensitive component). The suffix is the attacker's point of interest. For each sequence, if the model exactly reproduces c when prompted with p, it is considered a successful reconstruction. In masked language modeling, for every sensitive sequence $x = p||c||s$ in the training dataset, the attacker's aim is to reconstruct the masked-out sensitive component c, given a prefix p and suffix s.

Membership Inference: In this attack, the attacker attempts to infer whether a sensitive sequence x is part of the victim's local

training data or not. In this work, we evaluate the sentence-level membership inference introduced by Mireshghallah et al. [71].

Perplexity: Perplexity [36] is a widely used metric to measure how well the LM predicts tokens in the generated sentence. For a sequence of predicted tokens $x_1, \ldots, x_n$, perplexity is computed as:

$$\mathcal{P} = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log\theta(x_i|x_1,\ldots,x_{i-1})\right)$$

where θ is the language model. A low perplexity implies that the model is confident, i.e., less surprised by the generated text.

Exposure: Carlini et al. [13] first introduced exposure as a quantitative measure of how much a generative LM has unintentionally memorized a rare training sequence/canary. Concretely, they (i) compute a sequence's log-perplexity under the model, (ii) define the *rank* of an inserted canary, and (iii) set exposure to the negative log-rank, which can be interpreted as the reduction in guessing difficulty an attacker gains from access to the model. Higher exposure, therefore, means the canary has become disproportionately likely relative to other equally formatted candidates, signaling greater risk of targeted extraction. We use this metric, and defer full mathematical details to Appendix B. Additional background concepts, including FL, autoregressive and masked language models, and differential privacy, are provided in Appendix A.

3 Related Work

3.1 Privacy Leakage Attacks in LLMs

Privacy risks of LLMs have been extensively studied in prior works. The seminal work by Carlini et al. [14] proposes an attack to retrieve public internet texts by querying a pre-trained GPT-2 model. Our attack, however, aims to extract privacy-sensitive data when fine-tuning an LLM in FL. In the domain of RNN and LSTM models, Carlini et al. [13] highlight the problem of unintended memorization. They insert canaries into the training dataset and measure their leakage probability with a metric called 'exposure', which we also used in our work to evaluate our novel attack strategies. Mireshghallah et al. [72] study the effect of fine-tuning on memorization and show that fine-tuning the parameters higher in the LLM architecture results in the most data leakage. Our experiments in FL align with this observation, and we manipulate these selective weights that are most susceptible to private data for producing greater memorization. Cheng et al. [17] pre-conditioned the LLM with structured PII examples and used an online learning loop to improve extraction success. They defined in-training PII as findable on today's web, which is not always applicable, because some outputs may be simple inferences or guesses based on public patterns. Akkus et al. [2] showed that fine-tuning LLMs using generated data without an instructional structure can amplify PII leakage from their pre-training dataset, while Kim et al. [53] and Lukas et al. [64] investigated to what extent information memorized by the LM constitutes sensitive PII and whether existing defenses are sufficient to prevent leakage. Although their objectives are similar to ours in this work, they differ in two crucial aspects: (1) they are conducted under a *non-FL, centralized* threat model, where the central server is considered the adversary, whereas we focus on privacy leakage caused by the malicious participants in FL, (2) most of these works are evaluation based, measuring the extent of privacy leakage caused by finetuning/pretraining LLMs, while we propose dedicated methods to amplify such leakage besides its evaluation.

3.2 Privacy Leakage By Poisoning

Recent research has explored model and data poisoning to leak privacy, though most studies do not focus on language modeling [8, 111]. Rashid et al. [85], Wen et al. [104], Feng et al. [31], and Liu et al. [63] manipulate the weights of pre-trained LLMs to cause privacy leakage during fine-tuning. However, these attacks do not apply to FL. On the other hand, Fowl et al. [32] consider the server in FL as an adversary that actively poisons model architecture and parameters sent to users. Such attacks are easily detectable in pre-trained LLMs, where the architecture is well-known to clients. In contrast, our *static* attack method can passively leak the privacy of FL users after fine-tuning without poisoning the pre-trained model, thus fully preserving the utility of the model. Additionally, our *dynamic* attack further enhances Privacy Leakage by actively tampering with the model's weights during FL training, where the attacker is one of the FL clients, not the server.

3.3 Privacy Leakage in Federated Learning

Among existing works on FL privacy leakage [27, 42, 100], many have attempted to invert gradients to extract training data [27, 74]. Recent Gradient Inversion Attacks (GIAs), such as Gifd [30], Gipip [94] and Spear [24] reconstruct image/vision inputs by optimizing dummy data so that their induced gradients match the victim's shared gradients. Due to the non-differentiable nature of discrete text, the GIA threat to language models (LMs) is more limited than for images, but Li et al. [60] reconstructs private text by exploiting gradients from token-embedding layers, while the LAMP [4] attack inverts the client's gradients into natural text by alternating between continuous and discrete optimization. Critically, most GIAs assume server access to per-client gradients, which aligns naturally with **FedSGD** [67]. In contrast, our work uses a more standard and commonly used FL algorithm–**FedAvg** [61, 86], which shares only the final local model after multiple minibatch steps, so the server at best sees an aggregated, multi-step parameter delta rather than the exact client gradients–making classic GIAs much less applicable in practical FedAvg deployments. Apart from that, Hitaj et al. [42] adversarially train a GAN to reconstruct class-representative images from the client's private data. However, it is strictly applicable for finite class-based downstream tasks, such as image classification, and does not transfer to generative tasks like language modeling. Besides, Melis et al. [68] exploit gradients of the embedding (for text), convolutional or fully-connected layer (image) to infer certain properties from client's training data, such as gender, age, someone's facial attributes in an image etc, but unlike our attacks, they solely focus on classification task, and do not extract victim's private texts verbatim. Hu et al. [44] sample PII-contextual and frequency-prioritized prefixes to extract sensitive client data. Although they follow a similar probing strategy to ours, their methods exploit only the model's inherent memorization of PII rather than amplifying it. Decepticons [33], Loki [114] and Vu et al. [100] are three attacks that assume a malicious FL server with the ability to manipulate the model architecture and parameters before sending the model to clients, while FILM [39] considers an external adversary that

eavesdrops on the communication channel between the server and clients to reconstruct sentences from gradients. Our attack differs from theirs in that we neither treat the server as an active adversary nor assume any side-channel information leak. Section 7.2 presents a detailed comparison of our attack with Decepticons and FILM.

4 Impact of Privacy-Sensitive Data on Fine-tuning Large Language Models

To carry out different downstream applications, LLMs such as GPT-2 [84] and BERT [23] undergo a fine-tuning step that modifies their pre-trained weights according to those specific tasks [20]. However, privacy-sensitive texts have specific patterns and often contain out-of-distribution tokens, e.g., digits (in phone or credit card numbers), special characters, and out-of-vocabulary words (login credentials or physical addresses). To understand the impact of such privacy-sensitive data on fine-tuning, we investigate whether the changes in the pre-trained weights after fine-tuning the LLMs with such sensitive data show any differentiating patterns when compared to the weight changes after fine-tuning with regular English texts. Our key observation from this investigation is that there indeed exists a remarkable difference between the patterns in these two cases. Figure 1a depicts the norms of weight changes in all 12 transformer blocks, i.e., multi-layer perceptron (MLP) and self-attention layers after fine-tuning a GPT-2 (base) model G with regular English sentences $\mathcal{D}_{reg}$. Here, we observe a uniform impact on the weights across different layers, L, which suggests that the model's adaptation after fine-tuning is evenly distributed throughout the architecture. However, a distinct pattern emerges when fine-tuning G with privacy-sensitive texts $\mathcal{D}_{sen}$ such as phone numbers, email addresses, login credentials, credit card numbers, medical prescriptions, etc. Only a subset of weights, specifically the MLP sub-layers in the last few transformer blocks, denoted as $L_\star \subset L$, undergo a substantial transformation, while the remaining layers, denoted as $L_\ominus = L \setminus L_\star$, exhibit minimal/regular changes. This exact observation is also supported by Chen et al. [16]. Figure 1b shows the transformation of the MLP and self-attention layers of GPT-2 after fine-tuning with D_{sen} (Refer to Figure 8 for Gemma). We experiment with four different language models - Gemma, Llama-2, GPT-2, and BERT - using various types of sensitive data and find that as long as these sensitive texts have an **out-of-distribution** characteristic with respect to the training data, each of these LMs shows such disproportionate changes in their internal weights. This crucial observation supports the following hypothesis:

Hypothesis 1. *Weights in some selective layers of an LLM undergo significant changes compared to the other layers while fine-tuned on out-of-distribution data.*

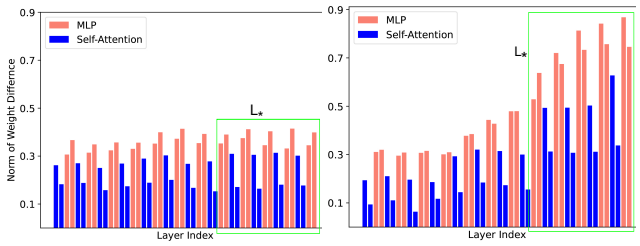
Formally, after fine-tuning with privacy-sensitive data, if $\Delta W_\star$ and $\Delta W_\ominus$ represent the mean weight changes in respective layers, they can be defined as below:

$$\Delta W_\star = \frac{1}{|L_\star|} \sum_{l \in L_\star} (W(f_G(\mathcal{D}_{sen})) - W(f_G(\cdot)))_l$$
$$\Delta W_\ominus = \frac{1}{|L_\ominus|} \sum_{l \in L_\ominus} (W(f_G(\mathcal{D}_{sen})) - W(f_G(\cdot)))_l$$

Here, $W(f_G(D))$ denotes the weights of model G fine-tuned with dataset D. We observe that $\|\Delta W_\star\| >> \|\Delta W_\ominus\|$. In contrast, when fine-tuning with regular texts, we observe $\|\Delta W_\star\| \sim \|\Delta W_\ominus\|$. This observation indicates that weights of certain layers ($L_\star$ marked with green rectangles in Figures 1a and 1b) within an LLM are more susceptible to capturing and memorizing privacy-sensitive information. We also scrutinize the direction/pattern of changes in those selective weights caused by out-of-distribution samples in the fine-tuning dataset. Then we ask the most important question- *if we further increase the amplitude of those particular weight changes along the same direction, does it make the model more biased towards the out-of-distribution data?* Later, in Section 5.3 where we discuss our Maximizing Data Memorization (MDM) techniques, we illustrate that the answer to this question is 'Yes', and we can strategically focus on modifying the weights of these sensitive layers to enhance the model's memorization of private data. But that is not all. With this also comes the salient question- *If we amplify the weight changes in such a manner, does that affect the model utility?* We also convincingly answer this question in Section 5.3 by showing that such modification hardly impacts the model's learning of regular English texts. In summary, the adversary aims to find a function $\mathcal{T} : W_\star \rightarrow W'_\star$ that generates a new set of weights $W'_\star$ which will maximize the memorization of $\mathcal{D}_{\text{sen}}$.

$$\arg\max_{\mathcal{T}} \mathbf{M}(\mathcal{D}_{\text{sen}}, \mathcal{T}(W_\star)) \quad (1)$$

We qualitatively define memorization $\mathbf{M}$ as a property of the LM in exposing particular training data points during inference. Quantitative measure of memorization $\mathbf{M}$ on $\mathcal{D}_{\text{sen}}$ includes but are not limited to data reconstruction [13], membership inference [66], and exposure, which we discuss in detail in Section 5.6.



(a) (b)

Figure 1: Norms of weight changes in MLP and self-attention layers of all 12 transformer blocks after fine-tuning GPT-2 with (a) regular English texts and (b) privacy-sensitive texts

5 Attack Methodology

5.1 Threat model

In our settings, both the adversary and the victim(s) are members of the client group participating in FL to fine-tune the LLM for a text auto-completion task, where they don't share any gradient information with each other. We assume that the adversary has some context of the victim's training data, specifically, the non-sensitive prefixes of the privacy-sensitive texts they want to extract from the victim's local dataset. This assumption is highly practical in corporate and administrative contexts where communication relies on boilerplate templates. For example, automated notification emails ('Your verification code is...'), medical intake forms, or banking alerts follow rigid structural patterns. Similar to the probing strategy in several existing attacks [14, 44, 53, 63], here the

Attack Attributes	Server Colllusion	Adversary's Action	MDM Method	Victim	Type of LM	Goal
Variations	No	Static	SWO	Targeted	Autoregressive (GPT-2)	Data Reconstruction
	Yes	Dynamic	WTL	Untargeted	MLM (BERT)	Membership Inference

Table 1: A high-level taxonomy of our attack methods

adversary only needs to know the standard template (prefix) to extract the unique sensitive suffix. Section 6.2 discusses more on our template-based prompt design. Apart from that, we consider two independent cases of the adversary's capability: (1) the adversary is a stand-alone malicious client in the FL setting who does not collude with the server and observe the global model in each round, and (2) optionally, the malicious client can collaborate with the server which enables it to exploit a specific victim's local model.

Without server cooperation, the adversary extracts certain types of sensitive data without targeting a specific client, and consequently, all the users having such types of data in their local datasets fall victim to this attack. In contrast, with server cooperation, they can also target a particular victim and compromise their privacy. However, unlike many existing works that consider the server as an active adversary [32, 33, 75], we assume a more practical attack surface where the server is **passive** and helps the malicious participants by only sharing some meta information. This is a more practical setting since the server has some accountability, and it could cause a loss of its reputation if it runs active attacks during the FL training. Table 1 shows a high-level taxonomy of our attack methods. We delve into all the mentioned attack attributes in the later parts of this section. Figure 4 presents a high-level overview of our attack methods. There are two vital steps in our general attack pipeline: (1) *Victim round identification (VRI)*: The adversary identifies the FL training rounds where victims with privacy-sensitive text of the adversary's interest participated and (2) *Maximizing data memorization (MDM)*: After the FL training is complete, the adversary manipulates the identified victim rounds' models to maximize the memorization of sensitive data.

5.2 Victim Round Identification (VRI)

As mentioned earlier, the attacker aims to extract some specific sensitive texts that follow certain patterns from the victims' local datasets. If multiple clients contain such private data of the attacker's interest, all of them fall victim to the attack. The objective of the VRI step is to identify some of the training rounds (not necessarily all) in which such victims participate. As mentioned in Section 1, LLMs tend to forget specific samples from an individual client's training data as they tend to generalize the overall data distribution during the training process. To better realize this phenomenon, we exploit the exposure metric from [13] and compute the average exposure (EXP) of the global model snapshots with regard to some victims' private data for all rounds of FL. As shown in Figure 2a, the value of EXP is higher for the rounds where the victim(s) participated (i.e., victim rounds, marked with blue dots) compared to the other rounds where they did not. Furthermore, EXP gradually decreases between two victim rounds, and more obviously, there is a persistent fall in the value of EXP after round 650 since the server did not select any victim in any round afterward (or any victim may not have participated). Based on our observation of this significant difference between the EXP values in victim and non-victim rounds, we want to deduce the following hypothesis:

Hypothesis 2. *The likelihood of a federated language model memorizing a sensitive data point in a training round is much higher if the data point is included in that round's training data compared to rounds where the data point is absent.*

Hence the identification of global model snapshots from victim rounds can add a new dimension to the attacks focusing on the victim's privacy leakage in FL. To exploit this key observation in our proposed attacks, when we assume the server does not cooperate with the adversary, we devise a mechanism for victim-round identification (VRI).

The key idea behind our VRI algorithm is to differentiate between the victim and typical (non-victim) rounds' snapshots based on the characteristics of $L_\star$. Let G_i be the global model snapshot at round i. We mark it as a victim snapshot $\widehat{G}_i$ if we identify that the victim participated in round i.

Step 1: The adversary stores the global model snapshots, G_i, where i represents any of the N training rounds when the attacker participates.

Step 2: We fine-tune a pre-trained LLM instance G with $\mathcal{D}_{reg}$ and denote it as G_r. Next, we fine-tune G with $\mathcal{D}_{reg} \cup \mathcal{D}_{sen}$, i.e., regular English and sensitive texts combined, and denote it as G_s.

Step 3: We calculate the norm of weight differences for each of the $L_\star$ layers (based on hypothesis 1) between G and G_r, denoted as δ_r, and between G and G_s, denoted as δ_s as given below:

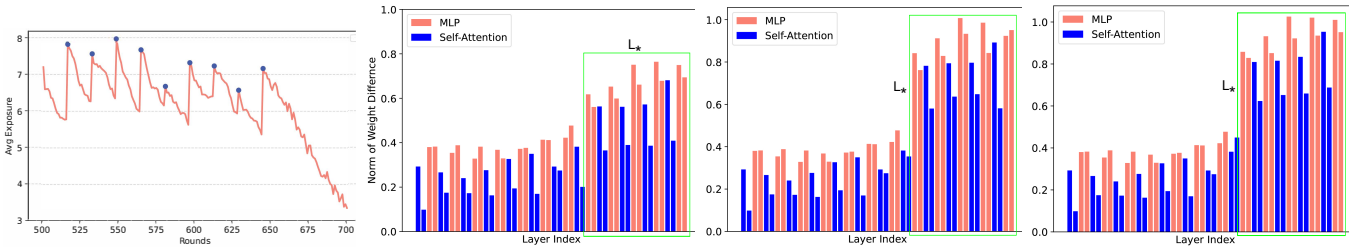
$$\delta_r = (\|W(f_{G_r}(\mathcal{D}_{reg})) - W(f_G(\cdot))\|)|_{L_\star}$$
$$\delta_s = (\|W(f_{G_s}(\mathcal{D}_{reg} \cup \mathcal{D}_{sen})) - W(f_G(\cdot))\|)|_{L_\star}$$

We similarly calculate the norm of weight differences for $L_\star$ between G and each G_i and denote it as δ_i. Our goal is to identify whether G_i is a victim round.

$$\delta_i = (\|W(f_{G_i}(\cdot)) - W(f_G(\cdot))\|)|_{L_\star}$$

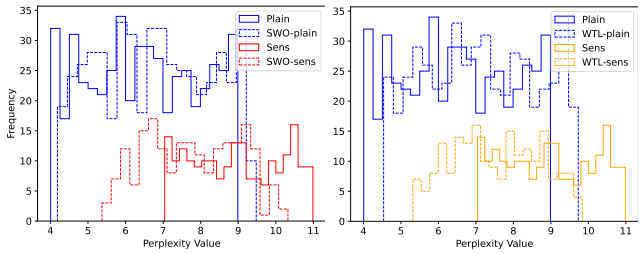
Here, each of δ_r, δ_s and δ_i is a vector of size $L_\star$.

Step 4: Based on hypothesis 2, our key intuition of this final step is that if G_i is a victim round snapshot, δ_s and δ_i would have a similar pattern in their distribution due to the presence of sensitive sequences in both of their training data, whereas δ_r and δ_i would have a significantly different distribution. For a better understanding, refer to Figures 1a and 1b. Hence, in this step, we perform two pairwise t-tests between δ_i and δ_r, and between δ_i and δ_s, respectively. For a predefined significance level, if the test with δ_r rejects the null hypothesis (H_0) and the test with δ_s accepts it, we mark G_i as a victim snapshot. Otherwise, G_i is categorized as a non-victim snapshot. To verify that such an approach works, we found the T-test between δ_s and $\|\Delta W_\star\|$ of any victim model almost always (95% cases) derive a p-value greater than the significance level, failing to reject the null hypothesis, which suggests an equivalent relationship between these two models. A detailed evaluation of our VRI method is provided in Section 7. It is important to note that $\mathcal{D}_{sen}$ has no overlap with clients' private data. They are just some dummy out-of-distribution texts that have a similar pattern as the sensitive texts. More details on the curation of $\mathcal{D}_{sen}$ and $\mathcal{D}_{reg}$ are provided in the Attack Settings of Section 6.2. We formally present our VRI technique in the Appendix (Algorithm 3).



(a) (b) (c) (d)

Figure 2: (a) Change of exposure of the private data with the passage of 200 FL rounds. The blue dots indicate those rounds when the victim participated. (b)-(d) Norms of the weight changes in the MLP and self-attention layers of all 12 transformer blocks of a victim snapshot, (b) without MDM, (c) with SWO, (d) with WTL



(a) (b)

Figure 3: Impact of MDM methods on data memorization and model's utility: (a) for SWO (b) for WTL. These results are generated by finetuning a GPT-2 base model with 1000 plain English texts and 200 out-of-distribution sensitive texts.

5.3 Maximizing Data Memorization

After we identify victim snapshots ($\widehat{G}_i$), the next step is to maximize the victim models' memorization of the private data based on hypothesis 1 by transforming their selected weights ($W_\star$) as shown in Equation 1. We design two approaches to achieve this:

Algorithm 1 Selective Weights Optimization (SWO)

1: **procedure** $\mathcal{J}(W_1, W_2, \Delta W_{init})$
2: $\quad \Delta W_{new} \leftarrow W_1 - W_2$
3: $\quad$ **return** $\Delta W_{init} \cdot \Delta W_{new}$
4: **end procedure**
5: **procedure** OPTIMIZE_WEIGHTS($W_1, W_2, \widehat{O}$)
6: $\quad \Delta W_{init} \leftarrow W_1 - W_2$
7: $\quad$ **for** step := 1 **to** num_steps **do**
8: $\quad\quad \mathcal{L} = -\mathcal{J}(W_1, W_2, \Delta W_{init})$
9: $\quad\quad$ Backpropagate($\mathcal{L}$)
10: $\quad\quad W_1 \leftarrow \widehat{O}(W_1)$
11: $\quad$ **end for**
12: $\quad$ **return** W_1
13: **end procedure**
14: **for** each $W_1 \in W_\star^{\widehat{G}_i}$ **AND** $W_2 \in W_\star^{G_r}$ **do**
15: $\quad$ Initialize an Optimizer, $\widehat{O}$ with W_1 and a learning rate
16: $\quad W_1 \leftarrow$ OPTIMIZE_WEIGHTS($W_1, W_2, \widehat{O}$)
17: **end for**

❑ **Maximizing data memorization with *Selective Weights Optimization (SWO)*:** We saw in the VRI step, for a victim round global snapshot $\widehat{G}_i$, the t-test between δ_i and δ_r rejects the null hypothesis. Hence we hypothesize that there must be a distinction in $W_\star$ between $\widehat{G}_i$ and G_r, denoted as δ_v.

$$\delta_v = (\|W(f_{\widehat{G}_i}(\cdot)) - W(f_{G_r}(\mathcal{D}_{reg}))\|)|_{L_\star} = \|W_\star^{\widehat{G}_i} - W_\star^{G_r}\|$$

This distinction in weight change characterizes how the model snapshot $\widehat{G}_i$ memorizes the privacy-sensitive sequences in contrast to the regular data. A higher difference essentially indicates higher memorization, resulting in more privacy leakage. Hence, we design an optimization problem that intends to maximize this difference (δ_v) by updating $W_\star$ of $\widehat{G}_i$ ($W_\star^{\widehat{G}_i}$) to increase memorization of $\widehat{G}_i$ on sensitive data. Besides, the update is done only in magnitude, keeping the initial direction (+ve and -ve sign of the values in $W_\star$) intact so the model does not deviate from its origin arbitrarily. Algorithm 1 is a streamlined demonstration of the fundamental steps of our proposed optimization scheme. We define a custom objective function $\mathcal{J}(\cdot)$ that takes a dot product between ΔW_{init} and ΔW_{new}. Here, ΔW_{init} is the initial weight difference, and ΔW_{new} is the updated weight difference after each optimization step between $W_1 \in W_\star^{\widehat{G}_i}$ and $W_2 \in W_\star^{G_r}$. Later, this dot product is used as the cost measure $\mathcal{L}$ in the optimization phase, based on which we run the backpropagation. Here, the dot product ensures that updates are aligned with the direction of the initial state (ΔW_{init}) and thus preserving the polarity of $W_\star^{\widehat{G}_i}$.

The optimization process can be performed using any conventional optimizer (e.g., SGD, Adam). The selective weights $W_1 \in W_\star^{\widehat{G}_i}$ are updated iteratively over a predefined number of steps. The optimizer is reset at each step, and the new difference (ΔW_{new}) as well as the cost ($\mathcal{L}$) are calculated with the updated W_1. We take the negative cost measure in line 8 since we aim to maximize it. Additionally, the optimization may include backpropagation with gradient clipping [78] and early stopping [80] to prevent exploding gradients [78] and running extra steps during the update. In Figure 2b and 2c, we show the contrast before and after applying SWO on a victim snapshot ($\widehat{G}_i$), respectively. It is evident that the norm of $\Delta W_\star$ has increased. Also Figure 3a indicates that the perplexity scores of the sensitive texts significantly decrease after applying SWO, whereas the scores remain almost as before in the case of plain English texts. Hence, we can deduce that with the selectively optimized weights, $\widehat{G}_i$ becomes more vulnerable to privacy leakage of the sensitive sequences without hurting its utility. Experimental results in Section 7 also advocate the above phenomenon.

Algorithm 2 Weights Transformation Learning (WTL)

procedure LEARN_TRANSFORM(G, $\mathcal{D}_{reg}$, $\mathcal{D}_{sen}$, n)
  $W_r \leftarrow \{\}$
  $W_s \leftarrow \{\}$
  for $i := 1$ to n **do**
    $G_r^i \leftarrow$Fine-tuned G with $\mathcal{D}_i \in \mathcal{D}_{reg}$
    $G_s^i \leftarrow$ Fine-tuned G with $\mathcal{D}_i \in \mathcal{D}_{reg} \cup D_{sen}$
    $W_\star^{G_r^i} \leftarrow W(f_{G_r^i}(\mathcal{D}_i \in \mathcal{D}_{reg}))|_{L_\star}$
    $W_\star^{G_s^i} \leftarrow W(f_{G_s^i}(\mathcal{D}_i \in \mathcal{D}_{reg} \cup \mathcal{D}_{sen}))|_{L_\star}$
    $W_r \leftarrow W_r \cup W_\star^{G_r^i}$
    $W_s \leftarrow W_s \cup W_\star^{G_s^i}$
  end for
  Initialize a supervised model $\mathcal{M}$
  Train $\mathcal{M}$ with W_r as input and W_s as output
  return $\mathcal{M}$
end procedure
$\mathcal{M} \leftarrow$ LEARN_TRANSFORM(G, $\mathcal{D}_{reg}$, $\mathcal{D}_{sen}$, n)
for each $W \in W_\star^{\widehat{G_i}}$ **do**
  $W \leftarrow \mathcal{M}(W)$
end for

❏ **Maximizing data memorization with Weight Transformation Learning (WTL):** If we have a closer look at Figures 2b and 2c, we observe that although the magnitude of impact on $L_\star$ is magnified after applying SWO, it does not fully coincide with the expected trend. In other words, the fluctuations of the magnitudes of $\|\Delta W_\star\|$ in Figure 2c differ from that of Figure 2b. A possible reason behind this could be modifying *all* the weight values of $W_\star$ as described above during our optimization process for maximizing the difference. To verify this hypothesis and also to get a fine-grained understanding of how the underlying weight values transform while fine-tuned with sensitive data, we do further analysis and come up with weight transformation learning (WTL) using an end-to-end supervised algorithm. The high-level idea is to train a supervised model to capture the hidden pattern of how the weights in $L_\star$ change in the presence of sensitive training data.

Algorithm 2 gives a simplified demonstration of this transformation learning technique. Firstly, we need to generate n (number of samples to train the supervised model $\mathcal{M}$) copies of G_r and G_s by fine-tuning G with different sets of $\mathcal{D}_{reg}$ and $\mathcal{D}_{sen}$, respectively. Here, D_{sen} does not need to be from the same distribution as the victim's private data because according to hypothesis 1, the weights in $L_\star$ undergo similar changes for any out-of-distribution data. We get n samples of $W_\star$ from both G_r and G_s, denoted as W_r and W_s, respectively. Finally, we train the supervised model $\mathcal{M}$ with W_r as input and W_s as output. Once $\mathcal{M}$ is trained, we can feed $W_\star$ of $\widehat{G_i}$ to $\mathcal{M}$ and get the updated weights. In Figure 2b and 2d, We show the contrast before and after applying WTL on a victim snapshot ($\widehat{G_i}$). Besides increasing the magnitude of impact for $L_\star$ compared to Figure 2c, it also shows a better alignment with the trend shown in Figure 2b. Besides, in Figure 3b, we observe that the language model's memorization of the sensitive texts increases (perplexity goes down) by a higher margin with WTL when compared with SWO in Figure 3a and the model's utility is also preserved.

5.4 Optional: Attacks With Server Collusion

The server can be *honest-but-curious*, a threat model that has been widely investigated in prior works [4, 102, 113]. In such cases, the server can passively cooperate with the adversary to enhance the victim participants' privacy leakage but does not actively run the attack to remain stealthy. The advantage of including the server under the adversary is twofold. Firstly, it is important to note that the victim/non-victim round information is readily available to the parameter server. Although we have designed a victim-round identification mechanism as described earlier in Section 5.2, it may incur some errors in practice (see Table 2). In contrast, the server can provide the adversary with ground-truth victim round information– thus making the privacy leakage attacks more successful. Secondly, in the FL setting, the adversary receives global model updates from the parameter server in every iteration. As discussed in Section 5.3, the adversary aims to maximize the memorization of the victim's privacy-sensitive data in those global model updates. However, it can be easily inferred without loss of generality that the impact of the victim's data will be higher on its **local** model's weights compared to the **global** model's weights because local updates of all participants go through federated averaging, which weakens the victim's impact on the global model. In other words, direct access to the local models may provide a greater advantage to the adversary in leaking the victim's privacy-sensitive information. Therefore, the server can share the victim's local model updates with the malicious clients, who can then maximize its memorization of private data using the techniques described in Section 5.3.

5.5 Static and Dynamic Modes of Attacks

We develop our attacks in two modes: *static* and *dynamic*. We denote the attacks with $static^+$ and $dynamic^+$, respectively, where the server cooperates with the adversary.

Static Mode: In this mode, the adversary does not actively interfere during the FL training but instead passively exploits the model after the fine-tuning is completed. It involves two actions:

❏ After getting a global model update from the parameter server, it attempts to identify whether it is a victim round. With server cooperation, this step can be omitted, as the server shares the victim's local updates with the adversary.

❏ After the entire training is complete, the adversary attempts to maximize the memorization of all identified victim snapshots using one of the MDM techniques discussed in Section 5.3.

Dynamic Mode Here, the adversary maliciously alters its local updates to magnify memorization of victims' private data during FL training. It involves the following two actions:

❏ The adversary aggregates all the victim's snapshots they have identified so far, creating an aggregated model G_A.

❏ When he participates in a FL round, it maximizes memorization of G_A using one of the two MDM methods. Later, instead of sending only its local update G_L to the server, it sends a weighted aggregation of G_L and G_A as : $G_{LA} = \alpha G_A + (1-\alpha)G_L$

The weight α determines the contribution of G_A to the final shared local update G_{LA}. For instance, if the adversary wants to send only G_A without sharing their local updates, they set α to 1. However, with server cooperation, all of the victim's global updates are simply replaced with their local instances in this approach. Such active interference by the adversary has the potential to increase the overall attack performance as it infects the global model by pushing the victim's snapshot more frequently in the training process.

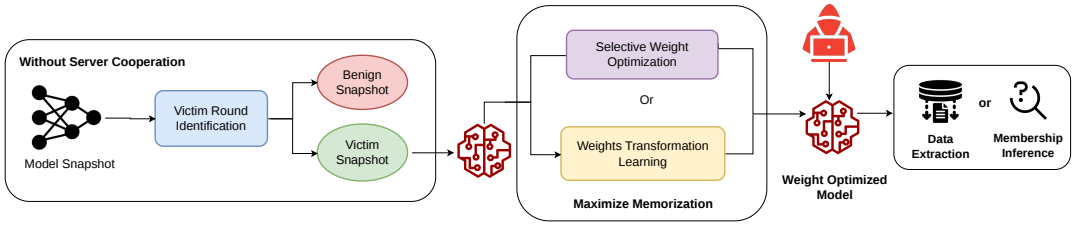


Figure 4: Overview of our attack flow. First, we identify the training rounds where the victim participated using the *VRI* algorithm. Next, we update the victim models to maximize the memorization of sensitive information using *MDM* algorithms. Finally, we prompt the model with crafted prefixes to retrieve sensitive information.

5.6 Evaluating Privacy Leakage

Data Reconstruction: After the VRI and MDM steps, our data reconstruction attack extracts private data from the victim models by prompting with the associated context using beam search [105]. Since our primary objective is to generate text with zero diversity and maximum resemblance to the training data, greedy decoding may seem to be a preferred approach. However, experiments show greedy decoding is subject to trivial memorization [14], e.g., repeated tokens and substrings, which often do not result in useful generations. We empirically find that using the beam search method with a small beam size (2-5) is more effective in generating private texts within the training data. This particular generation scheme causes a significantly higher leakage of private texts than the others. We take top-3 accuracy, i.e., the three highest-scoring outputs produced by the beam search for each reconstruction are considered as the candidate data points leaked from the training data (i.e., total 200 canaries*3=600 reconstructions) for our evaluation and some sample reconstructions are provided in Table 6.

Membership Inference: Besides data reconstruction, we also aim to evaluate whether our proposed strategies enhance the performance of *membership inference* (MI) [91] attacks. For this purpose, we adopt the reference-based attack proposed in [71], which involves calculating the percentage of private training samples that are correctly classified as training members out of a pool of both training and validation samples. To determine if a sample was part of the training set, two models are used: a target model, which is one of the victim models in our case, and a reference model. By comparing the likelihood probabilities from both models, a ratio is calculated, and if this ratio is below a certain threshold, the sample is considered part of the training set. We put the implementation details in the Appendix (Section E.1) due to space constraints.

6 Experiment Setup

6.1 Dataset Description

Our experiments include both canaries and real-world sensitive data. To observe how dataset size affects attack performance, we selected two datasets: the Penn Treebank (PTB) dataset [65], which is a small dataset consisting of only 5MB of text from financial news articles, and the Wikitext-103 (Wiki) dataset [70], which is a comparatively larger dataset consisting of 181MB of text. Similar to [13], we augment both of these datasets with out-of-distribution artificial canaries, which we manually crafted instead of web scraping to ensure that the pre-trained LLMs have not seen these texts before in their pre-training phase. These canaries contain dummy privacy-sensitive information. We insert four types of private information as canaries: phone number, credit card number, email, and physical address. Some examples of the canaries are provided in the Appendix (Table 6). It is worth mentioning that the training dataset may contain out-of-distribution texts that are not privacy-sensitive, such as product keys and hotline numbers. However, in our attacks, the adversary specifically designs the prompts to extract only the privacy-sensitive information they are interested in. The presence of other non-privacy-sensitive outlier texts in the dataset does not affect the effectiveness of the attack.

To confirm that our proposed method does not merely work for artificially inserted canaries, we also include the Enron email dataset [56] in our experiments, which has several naturally occurring sensitive pieces of information. It is a well-known collection of emails that were exchanged among employees of the Enron Corporation, a U.S. energy company. It contains private information, e.g., phone numbers, addresses, login credentials, and other sensitive details about individuals. We only retain the body of the emails and pre-process them by filtering out all attachments and auto-generated emails. We select a subset of the resulting dataset (412MB) and manually identify 125 privacy-sensitive sequences. Several existing works [13, 39, 64] on LLM's privacy conducted their experiments with these three datasets. For the Enron email dataset, we observe that the majority of the 125 private sequences have multiple occurrences. Approximately 29% of them occur twice, 18% occur three times, and 7% occur four or more times. This implies that although state-of-the-art defenses like differential privacy [28] provide guarantees under the assumption that records are unlikely to be duplicated, this may not be true for real-life datasets. To simulate this phenomenon in the PTB and Wikitext datasets, we generate 200 unique canaries and insert them into the datasets with k repetitions. Specifically, we consider $k \in \{1, 10, 50\}$, meaning one canary per $\sim \{46K, 4.6K, 920, 460\}$ training sequences for PTB, and per $\sim \{3.5M, 350K, 70K, 35K\}$ training sequences for Wikitext.

6.2 Training and Attack Implementation

FL Training Setup: We fine-tune Gemma-2b [95], Llama-2 [98], GPT-2 base [84] and BERT [23] as representative language models to evaluate our proposed attacks. Later we also ablate for the medium and large versions of GPT-2. Due to computation resource

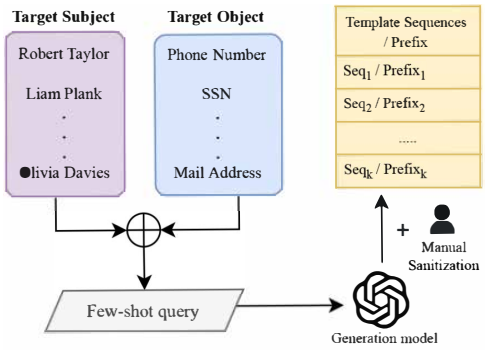


Figure 5: LLM-assisted construction of dummy sensitive sequences, $\mathcal{D}_{sen}$ and non-sensitive prefix templates for evaluating the data reconstruction attacks.

limitations, we could not train the Llama-2 model in a large-scale FL setting. Instead, we conducted our attacks on Llama-2 in a compact experimental setup. Relevant experiments with Llama-2 are included in Appendix C. Additional information about these models is available in the Appendix A.2. To make fine-tuning large LMs feasible in FL, we adopt Parameter-Efficient Fine-Tuning (PEFT) [43] for all models exceeding 1B parameters. Specifically, for Gemma-2b and Llama-2, we use QLoRA configurations consistent with recent large-scale FL and PEFT benchmarks. We apply LoRA to the attention projection matrices and MLP components. Unless otherwise specified, we use LoRA rank $r = 16$, scaling factor $\alpha = 32$, and dropout $p = 0.05$, with trainable parameters restricted to LoRA adapters while keeping the base weights frozen. Following Dettmers et al. [22], we quantize the base model to 4-bit NF4. For the primary FL task, we have 100 participants and randomly select 20 for each round. We vary the number of attackers and victims between 1 and 10 and distribute the sensitive sequences among the victims accordingly. We train the model for 700 iterations, each with five local epochs, using an initial learning rate of 10^{-4} with a linearly decayed learning rate scheduler.

Attack Settings: Details of attack settings are given below:

❏ We use a dataset of English plaintext jokes [81] as $\mathcal{D}_{reg}$ as discussed in Sections 4 and 5. This dataset exclusively contains 1622 regular English sentences without privacy-sensitive sentences. Besides, we generate 100 dummy privacy-sensitive sentences to use as $\mathcal{D}_{sen}$. As mentioned earlier, it does not overlap with the original private texts. As shown in Figure 5, the attacker utilizes their prior knowledge of the sensitive data pattern to craft few-shot queries for a text generation LLM and generate these dummy sequences that capture the model's behavior towards out-of-distribution data. Additionally, the number of epochs for fine-tuning G_r and G_s was empirically set to 20, with early stopping enabled.

❏ As depicted in Figure 1b, the size of $L_\star$ is important. We empirically choose the top 18 (for GPT-2, BERT), 20 (for Gemma), and 28 (for Llama-2) layers that show the highest norm as $L_\star$. Later, we also perform an ablation study by varying the size of $L_\star$.

❏ We use the Adam [54] optimizer in the selective weights optimization algorithm. We empirically tune the number of steps between 200 and 1000, the clipping threshold between 1 and 4, and set the initial learning rate at 1×10^{-3}.

❏ Since model weights do not fall into conventional categories of data, we experiment with several learning algorithms for the transformation task to identify the most suitable one. These algorithms include normalizing flows (e.g., planar flows [58] and RealNVP[25]), dense autoencoder (DAE), variational autoencoder (VAE)[55], convolutional autoencoder (CAE)[112], and transformer[99]. We find that DAE, CAE, and transformer showed comparable performances (in terms of validation loss) while normalizing flows and VAE did not yield promising outcomes. However, we eventually chose to employ DAE for this task as the other two, particularly the transformer, turned out to be computationally much more expensive taking roughly twice the time of DAE to complete the training.

❏ To ensure better stability during the transformation, we use Z-score normalization [3] to normalize the weight vector before passing it to the SWO or WTL algorithm. After the transformation, we denormalize the output weight to its original scale. For the sake of simplicity, we skip the normalization step in Algorithms 1 and 2.

❏ For data reconstruction, we follow a similar probing approach to many existing works [14, 53, 63] that utilize non-sensitive prefix templates to extract sensitive suffixes from private training data. Although most works constructed prefixes via manual or frequency-based sampling [44, 63], we adopt an LLM-based generation pipeline followed by manual sanitization. As shown in Figure 5, the attacker aims to extract certain objects, such as phone numbers and SSNs, that belong to targeted subjects. Given those target subjects and objects, we query the GPT-4o API model to generate K different sequences with different combinations of target subjects and objects. To ensure format consistency (i.e., prefix $\oplus$ suffix template), we include few-shot examples into the query, e.g., "The police should urgently reach out to <Subject's name> by calling at <dummy phone number>" or "<Subject name> has a savings account in Bank of America, associated with the SSN: <dummy SSN>". Following the few-shot query, the model generates K sequences, which undergo a second format check via manual validation. Finally, the prefix portions are separated from these sequences using regular expressions for probing purposes in our data reconstruction attack.

6.3 Baselines and Metrics

We introduce four baseline scenarios to conduct a comparative analysis of different attack strategies:

❏ **Last Round (LR)**: This is a naive attack scenario where the adversary has access to only the final model achieved after completing FL training, with no additional attack strategies applied.

❏ **Intermediate Rounds (IR)**: The attacker has access to the ground truth information of intermediate victim rounds from server cooperation as mentioned in Section 5.4. IR directly prompts these victim models without applying any attack strategy.

❏ **Forced Participation (FP)**: As one of the extreme scenarios, we assume that the server intentionally manipulates the FL protocol and client selection process. Instead of random selection, the server forces the victims to participate in each round. This unfair action gives the highest importance to the victims' local data. Since the victims' local updates are getting exposed in each iteration, it is not

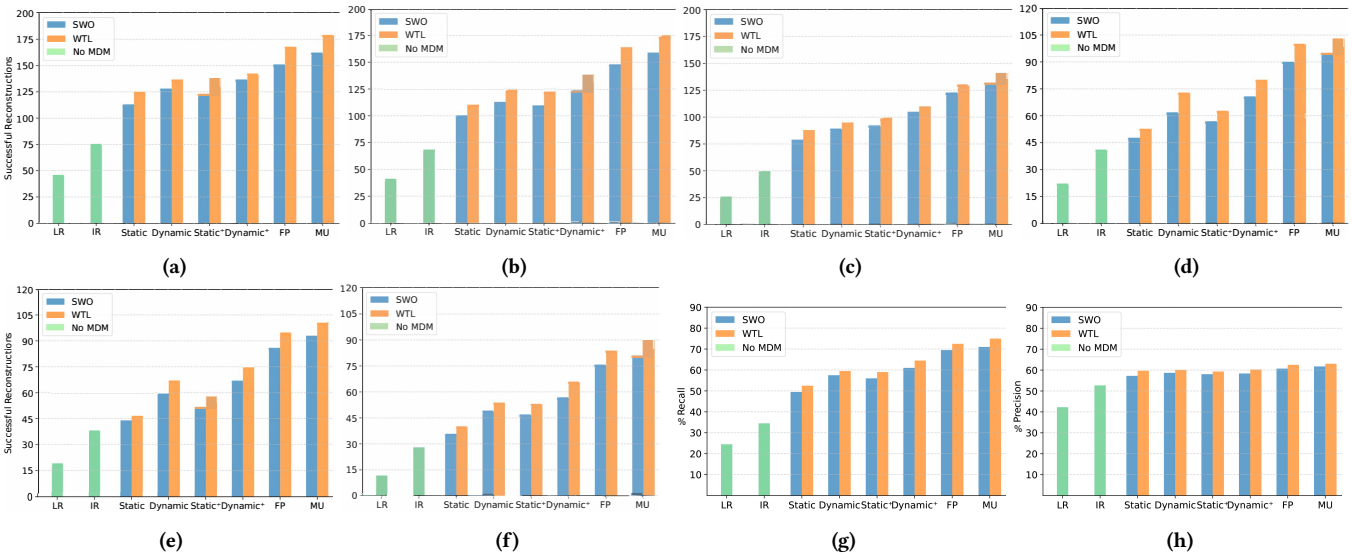


Figure 6: Number of successful reconstructions out of 200 unique canaries for different attack strategies along with the baselines for (a) Wikitext with Gemma, (b) Wikitext with GPT-2, (c) Wikitext with BERT. Then, out of 125 in-house sensitive sequences for (d) Enron with Gemma, (e) Enron with GPT-2, and (f) Enron with BERT. Finally, membership inference (g) recall and (h) precision scores for different attack strategies along with three baselines on the Wikitext Dataset using BERT.

necessary to consider the intermediate rounds. Instead, we consider the final model after the training is completed.

❏ **Malicious Update (MU)**: In the strongest attack case, the server not only manipulates the client selection, but it also manipulates what the victim receives from the server in each iteration. The server keeps returning the victim's local update from the previous iteration to it instead of the global update. This forces the victim to fall into a vicious cycle of overfitting that continues till the training ends. In this case, we consider the final local update of the victim for the same reason as FP.

The first scenario will help us understand why we chose to exploit the intermediate snapshots of victim rounds instead of the final model (hypothesis 2). The second scenario is to imply how much leakage the intermediate victim rounds cause without applying our proposed attack methods. On the other hand, FP and MU, with strong assumptions (i.e., less practical) on the adversary's capability, set an upper bound to our attack performance.

Metrics. We report the number of successful reconstructions (NSR) of the artificial canaries (for Wikitext and PTB) and the inherent sensitive texts (for Enron). We also evaluate the level of leakage using exposure (EXP), a metric introduced in [13], where higher exposure indicates greater privacy leakage. For membership inference, we report the recall and precision following [71]. Moreover, we use validation perplexity (PPX) as a metric for the performance of the model, where lower perplexity refers to better model utility.

7 Evaluation

We aim to answer the following research questions through a comprehensive empirical evaluation.

RQ 1: Can a malicious client identify victim rounds accurately?
RQ 2: Can a malicious client achieve higher privacy leakage by exploiting the model updates it receives during training as opposed to exploiting only the final model?
RQ 3: Is it possible to increase an LLM's memorization of privacy-sensitive data by tampering with its weights?
RQ 4: How does the inclusion of the server as a passive adversary impact the attack performance?

Victim Round Identification Performance. Table 2 demonstrates the results of our victim round identification mechanism. We ablate the total number of victims from 1 to 10 for both static and dynamic attacks. With increased victim count and in dynamic attack mode, victim model updates come more frequently during training. It bolsters the global model's memorization of the victim's data, which in turn, helps identify the victim rounds. However, those extra victim updates during training also impact non-victim training rounds, which generate more false-positive outputs. Hence, with the increased number of victims, the precision scores fall slightly (See confusion matrices in Figure 12). Nevertheless, our VRI method yields high precision and recall even in the most challenging setting of only a single victim, suggesting that it can identify the victim rounds with significant success, which answers **RQ 1**.

Data Reconstruction Performance. Figure 6b and 6a demonstrate the results across different attack strategies and the four baselines on the Wikitext dataset, in terms of the number of successful reconstructions when autoregressive text generation is performed with Gemma and GPT-2 respectively. Overall, The Gemma model achieves a superior NSR compared to GPT-2, mainly due to its larger size and complexity. First, comparing the scores of the first two baselines answers **RQ 2**. A higher score of IR compared to LR indicates that model snapshots from intermediate victim rounds cause

greater leakage than only the final model (hypothesis 2), which justifies the importance of our victim round identification step. Next, for both Gemma and GPT-2, our static attack outperforms IR by 15-18% with both SWO and WTL, answering **RQ 3**. **This shows that it is indeed possible to significantly increase the LM's memorization of clients' private data by maliciously tampering with its weights.** Additionally, WTL performs better than SWO in all cases, proving it to be a better MDM technique. Apart from that, introducing active threats during training further increases privacy leakage, as our dynamic mode of the attack has a much better NSR (137 for Gemma) than the static mode (125). Furthermore, incorporating victims' local updates in place of their global counterparts with cooperation from the server boosts the attack performance considerably (by 6 to 8%) in both $static^+$ and $dynamic^+$ modes of the attack. In fact, $dynamic^+$ is our best-performing attack variant with 71% (NSR=142) successful reconstructions.

Expectedly, a sharp peak can be noticed for both FP and MUscores, which represent an upper bound. Evidently, pushing victim updates into each training iteration maximizes the model's memorization of the victim's local data, including the privacy-sensitive instances. Even though these two baselines assume strong attacker capabilities, they don't significantly outperform our proposed strategies in terms of attack performance. Instead, these baselines, along with $static^+$ and $dynamic^+$ demonstrate that including the server as part of the adversary can lead to a more severe privacy leakage. This answers our last research question **RQ 4**. Figure 6c illustrates attack performance in masked language modeling, i.e., with BERT. Overall, NSR scores for BERT are much lower than GPT-2. The improvement in attack performance for WTL over SWO is also not as prominent in this case as it was for GPT-2. Jagannatha et al. [47] show that masked LMs have lower privacy leakage than autoregressive LMs. This could be attributed to the difference in the architecture of these two types of LMs and how they understand the context. BERT understands the context from both the left and the right sides of a masked-out word, whereas Gemma/ GPT-2 uses only the left context to predict the next word. BERT's bidirectional attention might help it distribute its focus, thereby reducing the chances of overfitting to specific examples. However, our best attack strategy without any server cooperation (dynamic) yields around 95 out of 200 successful reconstructions, while with $dynamic^+$the NSR reaches 55%. From the results, it is quite evident that our proposed attack methods are highly effective against masked language modeling in addition to autoregressive text generation. Due to space constraints, we put the results on the PTB dataset in Appendix D.

Finally, our attack performance on real-world data, i.e., the Enron email dataset, is depicted in Figures 6e and 6f. It is worth mentioning that the privacy-sensitive sequences in this dataset are less structured than the artificial canaries, making the attacks more challenging and having some interesting impact on the results. Firstly, both the static and $static^+$ attacks slightly underperform in this scenario compared to artificial canary extraction. As shown in Figures 6e and 6f, the NSR scores of the static attack for both Gemma/GPT-2 and BERT are quite close to IR's score. Nevertheless, active interference by the adversary changes the picture notably, as we can see the dynamic attack on Gemma increases the NSR by around 10% from the static variant, while the $dynamic^+$ attack manages to extract 64% (80 out of 125) of the private sequences. It shows that our attacks can successfully compromise the privacy of real-world data with high accuracy. Similar to the Wikitext dataset with canaries, masked language modeling with BERT turns out to be more resilient than Gemma/GPT-2 against leakage for the Enron dataset with real-world private data (Figure 6f). Results with respect to the EXP metric are moved to Appendix B.

	#Victim=1		#Victim=5		#Victim=10	
	Rc (%)	Pr (%)	Rc (%)	Pr (%)	Rc (%)	Pr (%)
Static	86.29	82.43	88.43	80	91.14	79
Dynamic	88.86	79.14	91.86	77.71	94.14	76.57

Table 2: Recall (%) and precision (%) of victim round identification step from 700 rounds.

Membership Inference Performance. We adopt the implementation of [71] for membership inference based on the likelihood ratio hypothesis, where part of our test set consists of all the canaries (for PTB and Wikitext) or private texts (for Enron). We insert 200 canary-like false-positive (outside training data) samples into the test set, which are semantically close to the canaries but do not match any of their secret portions. The evaluation is done over 400 samples (200 canaries + 200 false positives) for both Wikitext and PTB. For the Enron dataset, we add 125 false positives into the test set along with the 125 already identified private sequences, making 250 samples in total. Looking at the results for Wikitext in Figure 6g, we can see that the static mode significantly improves the recall score (by 15-26%) compared to the lower baselines (LR, IR). This demonstrates the effectiveness of our MDM algorithms in identifying the canaries and confidently inferring their membership. We achieve the highest recall of 61% with the $dynamic^+$ version, which shows results similar to the upper baselines (FP and MU). Our MDM approach directs the language model to prioritize sensitive training samples, leading to assigning a higher likelihood to those samples. This reduces the number of missed positive samples, increasing the true positive count and decreasing the false negative count. Consequently, the recall rate experiences a significant improvement due to these dual effects. However, there was only a slight improvement in the ratio of true positives to false positives. As a result, the attack performance did not fluctuate much across different attacks and the baselines in terms of precision as shown in 6h. For a better understanding of the phenomenon, please refer to the confusion matrices in Figure 12 and Section E.3 of the Appendix. We also put the membership inference results for PTB and Enron into the Appendix (Section E.2) for space constraints. We conducted extensive ablation studies on our attack performance based on six different criteria: data repetitions, prompt perturbation, number of sensitive layers, number of victims and attackers, frequency of victim rounds, and model size. Please refer to Appendix G for details.

7.1 Attack Cost Analysis

For VRI, storing the model weights at each iteration is not required. Only the candidates that pass the VRI test need to be stored. Additionally, the attacker needs to store only the weights of the selective sensitive layers (L*) of an LM instead of the entire base weights. For a Gemma-2b model, with 4-bit QLoRA, the total storage cost when we store the entire model weights (~4.5 GB) in each iteration will be $O(N) := 700 * 4.5/4 =$ **787.5 GB**. But the number of layers

Model	Decepticons	static	dynamic	static$^+$	dynamic$^+$	FP	MU
GPT-2	137	111	124	123	139	164	174
BERT	42	88	95	99	110	130	141

Table 3: NSR comparison among our attacks with WTL, Decepticons [33] and three baselines on 200 canaries.

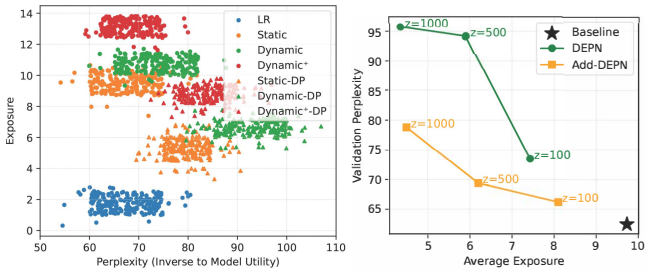
in L* is 20 out of 164 total layers ($s = 20/164$), and if the fraction of rounds where victims participate is m=50%, the space complexity will be $O(m * s * N) := 0.5 * 787.5 * 20/164 =$ **~48 GB**, which is **~93%** less than $O(N)$, making it highly feasible for an attacker.

7.2 Benchmark Study

We compare our results with Decepticons [33], which aims to uncover private user text by deploying malicious parameter vectors from the server. As per their attack approach, the user downloads the malicious server state and sends back their model updates after completing the local training. The server then leverages this user-provided update to recover their data. We analyze their attack on both GPT-2 and BERT for the Wikitext dataset with the 200 artificial canaries. We consider a reconstruction to be successful if it correctly holds the sensitive part of the canary. For GPT-2, it could reconstruct 137 canaries out of 200, whereas the NSR for BERT decreased to 42, as shown in Table 3. While their proposed method assumes the parameter server as an active adversary that sends malicious updates to the user, our static and dynamic attacks perform competitively with GPT-2 without any server cooperation. Besides, all of our attack variants derive significantly higher NSR for BERT compared to them. Finally, with the cooperation of an honest-but-curious server, our *dynamic*$^+$ attack marginally outperforms them with GPT-2, while the high NSR scores for FP and MU indicate that it is possible to cause greater privacy leakage if we consider the server as an active adversary. Besides, we also analyze the FILM attack [39] on GPT-2 which aims to recover private text data in federated learning. We put the related benchmark study for this work into the Appendix (F) due to space constraints.

7.3 Privacy Defenses

Differential Privacy: The DP-SGD algorithm [1] is used to train neural networks (NN) with differentially private guarantees. To benchmark the performance of our attack against differential privacy (DP), we implement DP training for the client language models [62]. Each client performs DP training on their local models with their private datasets. We train using various ϵ values and set $\delta = n^{-1.1}$ in each case, n is the number of local training samples. The ℓ_2-norm for gradient clipping is set to 0.01. More details about the algorithm are available in Appendix A.3. We conducted the experiment on the Enron email dataset and calculated EXP for all 125 private samples. We also picked 125 non-private samples from the victim's local dataset and calculated the model's perplexity for them. Figure 7a plots the EXP of these private samples against the PPX of the non-private samples. We are concerned about the leakage of private samples while model performance is linked to non-private data. From Figure 7a, we see that DP reduces the EXP for the private samples. Additionally, we see that the distribution



(a) (b)

Figure 7: The tradeoff between model utility (in terms of perplexity) and private data memorization (in terms of exposure) (a) with DP ($\epsilon = 8$) (b) with DEPN and Add-DEPN

of EXP scores is less concentrated with DP. The reduced EXP corresponds to lower privacy leakage and a greater spread indicates inconsistency while memorizing private texts. The PPX increases with DP, indicating a reduction in utility. Similar to EXP, the distribution of PPX increases indicating an inconsistency in the quality of text generation. However, the EXP of other attack methods with DP is still higher than baseline LR. Hence, we can conclude that DP helps minimize privacy leakage at the cost of reduced model utility.

Pruning/Regularization A complementary line of defense is to directly regularize those internal neurons that are most responsible for leaking private information. We therefore evaluate DEPN [106], a neuron-level privacy protection framework that first identifies "privacy neurons" via gradient-based attribution and then edits their activations to suppress memorization of private sequences. Here, we utilized the same local dataset as the DP experiment (125 private, 125 non-private Enron data) for evaluation. For each private sample, we compute privacy attribution scores for all FFN neurons and aggregate them over examples to obtain a global ranking. We then select the top $z \in \{100, 500, 1000\}$ neurons as privacy neurons and apply DEPN's editor to those neurons only. We quantify privacy leakage as the average exposure across all private samples, and model utility using validation perplexity on the non-private samples in Figure 7b. In the original DEPN editor, the activations of the selected privacy neurons are simply set to zero at inference time, effectively erasing their contribution to the logits. Consistent with DEPN's analysis, we observe that increasing the number of edited neurons monotonically reduces exposure, as shown by the green line in Figure 7b. However, in our federated Enron setting, this hard erasure behaves as an extremely aggressive regularizer, severely damaging utility despite its reduced exposure. To obtain a softer form of regularization, we introduce an additive privacy neuron editor (**Add-DEPN**) inspired by model-editing methods such as ROME [69]. Instead of zeroing, we apply a small additive correction, i.e., $h_k^{(l)} = h_k^{(l)} - \Delta$, where Δ is chosen as a function of the privacy attribution score: $\Delta = \eta \cdot \text{Att}\left(w_k^{(l)}\right)$. This means neurons with higher privacy contribution get slightly suppressed, not destroyed. As depicted by the orange line in Figure 7b, exposure is reduced almost as much as with DEPN's zeroing editor, but the increase in perplexity is substantially smaller. For moderate values (e.g.,

z=500), additive editing yields a sizable reduction in NSR, as shown in Table 4, while keeping validation perplexity close to that of the baseline undefended model. We recommend utilizing Add-DEPN as a practical regularization mechanism at the client side in order to mitigate such private information leakage.

Scrubbing: It is a method [19] to identify and remove certain tokens from text. While it is frequently used for text anonymization [79], we use it for removing privacy-sensitive tokens from user text. Compared to canaries, it is more challenging to scrub real-world private data, like Enron, since they are less structured and more stochastic. Nevertheless, we designed several regular expressions to capture the privacy-sensitive tokens in Enron and replace them with the token <UNK>. We could scrub out the sensitive tokens from almost 47% of the sequences. As shown in Table 4, unlike DP, scrubbing reduces the NSR without hurting the model's utility. However, applying DP and scrubbing together (SCR + $\epsilon = 8$) further reduces NSR but greatly hurts the model's utility. In contrast, applying regularization (i.e., Add-DEPN) along with scrubbing yields similar privacy gains with much better validation performance.

Secure Aggregation: It aims to solve the problem of computing a multiparty sum where none of the parties reveal their updates in the clear, even to the aggregator [9, 10]. In FL, the clients contribute individual updates in each epoch, and all the parties securely aggregate the updates to only learn the summation. It can be achieved by masking the individual updates using the shares of zero [9], homomorphic encryption [18, 21], or functional encryption [108]. Secure aggregation makes any attacks that involve server manipulation [33, 114] obsolete, as the server can not access individual victim updates. We implement CKKS-style [77] homomorphic encryption (HE) using the TenSEAL framework [5, 88] on our FL setting with GPT-2 base and Enron emails. Each client encrypts its local model update under a shared CKKS public key before uploading to the server. We use TenSEAL's CKKSVector interface to encode real-valued updates with a fixed scale and pack multiple update coefficients into ciphertext slots, enabling the server to compute the element-wise sum by homomorphic additions without ever observing any individual in the clear. The resulting aggregated ciphertext is decrypted only by the designated key holders, e.g., clients. As noted earlier, HE blocks attacks with server collusion, but Table 4 shows that HE is nearly as ineffective as the undefended baseline at reducing NSR for our client-only attack because the attacker (and other clients) still has access to the global model. Nevertheless, due to the approximate arithmetic in CKKS, there is a marginal signal-to-noise degradation in the decrypted aggregated update, which somewhat reduces both NSR and utility.

Fully Encrypted FL: A stronger line of defense performs end-to-end encrypted FL using multi-party HE [73], where model weights and local gradients remain encrypted throughout training. POSEIDON [87] introduced a multiparty lattice-based CKKS scheme to collaboratively train neural networks, explicitly aiming to protect training data, intermediate model updates, and final model weights in a passive-adversarial setting. Hercules [107] follows the POSEIDON framework but focuses on making fully encrypted FL faster and more accurate by leveraging packed ciphertext and SIMD-style computation. In both methods, neither the server nor any clients can directly observe plaintext gradients/updates, which alters the attack surfaces that assume access to intermediate updates. However, such training methods remain computationally intensive and often require careful approximations of nonlinearities; even POSEIDON reports long training times for more complex workloads. While these approaches can offer stronger privacy guarantees, they have not yet been implemented at the LLM scale, where the model size and training dynamics would incur the overhead of performing millions of HE operations.

Metric	No Defense	$\epsilon = 8$	$\epsilon = 2$	Add-DEPN **z = 500**	**SCR**	SCR **+ $\epsilon = 8$**	Add-DEPN + SCR	HE
NSR	67	46	29	38	41	29	29	64
Avg. PPX	62.44	74.82	87.73	68.27	55.25	78.17	69.5	67.3

Table 4: NSR and model's utility, i.e., average perplexity in Enron email dataset with two defenses: Differential privacy with different ϵ, Scrubbing (SCR), Add-DEPN, and HE

Data Deduplication It removes duplicate training samples in large datasets before training LMs [59] and is used to reduce the amount of memorization in a trained LM [12, 59]. Training examples that are repeated more often, as discussed in Appendix G.1, are more likely to be extractable. Hence, removing multiple occurrences certainly reduces the possibility of leakage to some extent. Apart from that, there are a few gradient-based defenses [15], such as signSGD [6], which do not apply against our attack, since FedAvg does not allow gradient sharing.

So, to summarize our discussion on the privacy defenses, we recommend **scrubbing** the local corpus with common PII patterns and **deduplication** of repeated samples as a data preprocessing step, and then applying modest regularization techniques like **Add-DEPN** on the local model to substantially reduce leakage propensity of an FL client's privacy-sensitive data.

8 Limitations and Future Work

We experiment on medium-scale English transformer LMs. It would be interesting to see if larger production models and multilingual settings exhibit different memorization and tampering behavior. Moreover, co-designing new FL protocols that are inherently robust to such weight tampering with minimal overhead would be more impactful. An important direction is to study whether similar vulnerabilities arise in other modalities, such as image transformers [82] and multimodal models [26], in federated vision and vision–language tasks, thereby revealing how privacy leakage manifests when parameters jointly encode text and non-text signals.

9 Conclusion

This work found that a malicious client in federated learning can confidently identify the rounds where some other clients participate with privacy-sensitive data. Without requiring any gradient information, the attacker can retrieve more private data by exploiting the model checkpoints from these interim rounds than they could with just the final model. Furthermore, if the adversary manipulates the selective weights of the model using our proposed MDM techniques, privacy breaches can be intensified. However, choosing a balanced combination of defense strategies could mitigate the privacy risk of an FL client without significantly hurting utility.

Acknowledgment

This work was supported in part by NSF grant 2442825. We thank the anonymous reviewers for their feedback and suggestions.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[2] Atilla Akkus, Masoud Poorghaffar Aghdam, Mingjie Li, Junjie Chu, Michael Backes, Yuyang Zhang, and Sinem Sav. Generated data with fake privacy: Hidden dangers of fine-tuning large language models on generated data. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 8075–8093, 2025.

[3] Mohammed Z Al-Faiz, Ali A Ibrahim, and Sarmad M Hadi. The effect of z-score standardization on binary input due the speed of learning in back-propagation neural network. *Iraqi Journal of Information and Communication Technology*, 1(3):42–48, 2018.

[4] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654, 2022.

[5] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[6] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International conference on machine learning*, pages 560–569. PMLR, 2018.

[7] Bigcode-project. Bigcode pii dataset. https://huggingface.co/datasets/bigcode/bigcode-pii-dataset, 2023.

[8] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy*, pages 175–199, 2023.

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017.

[10] Keith Bonawitz, Fariborz Salehi, Jakub Konečnỳ, Brendan McMahan, and Marco Gruteser. Federated learning with autotuned communication-efficient secure aggregation. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1222–1226. IEEE, 2019.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[12] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2022.

[13] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, volume 267, 2019.

[14] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[15] Wenhan Chang and Tianqing Zhu. Gradient-based defense methods for data leakage in vertical federated learning. *Computers & Security*, 139:103744, 2024.

[16] Ruizhe Chen, Tianxiang Hu, Yang Feng, and Zuozhu Liu. Learnable privacy neurons localization in language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 256–264, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[17] Shuai Cheng, Shu Meng, Haitao Xu, Haoran Zhang, Shuai Hao, Chuan Yue, Wenrui Ma, Meng Han, Fan Zhang, and Zhao Li. Effective pii extraction from llms through augmented few-shot learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 8155–8173, 2025.

[18] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

[19] Somnath Basu Roy Chowdhury, Sayan Ghosh, Yiyuan Li, Junier Oliva, Shashank Srivastava, and Snigdha Chaturvedi. Adversarial scrubbing of demographic information for text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 550–562, 2021.

[20] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[21] Vishnu Asutosh Dasu, Sumanta Sarkar, and Kalikinkar Mandal. Prov-fl: Privacy-preserving round optimal verifiable federated learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, AISec'22, page 33–44, New York, NY, USA, 2022. Association for Computing Machinery.

[22] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *37th Conference on Neural Information Processing Systems*, 2023.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

[24] Dimitar I Dimitrov, Maximilian Baader, Mark Müller, and Martin Vechev. Spear: Exact gradient inversion of batches in federated learning. *Advances in Neural Information Processing Systems*, 37:106768–106799, 2024.

[25] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[26] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Jiacheng Du, Jiahui Hu, Zhibo Wang, Peng Sun, Neil Zhenqiang Gong, Kui Ren, and Chun Chen. Sok: On gradient leakage in federated learning. In *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

[28] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[29] Ahmed Roushdy Elkordy, Jiang Zhang, Yahya H Ezzeldin, Konstantinos Psounis, and Salman Avestimehr. How much privacy does federated learning with secure aggregation guarantee? In *Proceedings on Privacy Enhancing Technologies*, 2023.

[30] Hao Fang, Bin Chen, Xuan Wang, Zhi Wang, and Shu-Tao Xia. Gifd: A generative gradient inversion method with feature domain optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4967–4976, 2023.

[31] Shanglun Feng and Florian Tramèr. Privacy backdoors: stealing data with corrupted pretrained models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[32] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

[33] Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. *arXiv preprint arXiv:2201.12675*, 2022.

[34] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[35] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[36] Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10136–10148, 2023.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[38] Kang Gu, Ehsanul Kabir, Neha Ramsurrun, Soroush Vosoughi, and Shagufta Mehnaz. Towards sentence level inference attack against pre-trained language models. *Proceedings on Privacy Enhancing Technologies*, 3:62–78, 2023.

[39] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering private text in federated learning of language models. *Advances in neural information processing systems*, 35:8130–8143, 2022.

[40] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE intelligent systems*, 24(2):8–12, 2009.

[41] Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G Flores, Jan Kautz, Daguang Xu, et al. Do gradient inversion attacks make federated learning unsafe? *IEEE Transactions on Medical Imaging*, 42(7):2044–2056, 2023.

[42] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

[43] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[44] Yingqi Hu, Zhuo Zhang, Jingyuan Zhang, Jinghua Wang, Qifan Wang, Lizhen Qu, and Zenglin Xu. Simple yet effective: Extracting private data across clients in federated fine-tuning of large language models. In *Proceedings of the 14th*

International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1808–1827, 2025.

[45] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, 2022.

[46] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems*, 34:7232–7241, 2021.

[47] Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305*, 2021.

[48] Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*, 2022.

[49] Xue Jiang, Xuebing Zhou, and Jens Grossklags. Comprehensive analysis of privacy leakage in vertical federated learning during prediction. *Proceedings on privacy enhancing technologies*, 2022.

[50] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[51] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[52] Virapat Kieuvongngam, Bowen Tan, and Yiming Niu. Automatic text summarization of covid-19 medical research articles using bert and gpt-2, 2020.

[53] Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36:20750–20762, 2023.

[54] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[55] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[56] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, volume 45, pages 92–96, 2004.

[57] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

[58] Tin Lai and Fabio Ramos. Plannerflows: Learning motion samplers with normalising flows. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2542–2548, 2021.

[59] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[60] Bowen Li, Hanlin Gu, Ruoxin Chen, Jie Li, Chentao Wu, Na Ruan, Xueming Si, and Lixin Fan. Temporal gradient inversion attacks with robust optimization. *IEEE Transactions on Dependable and Secure Computing*, 2025.

[61] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[62] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022.

[63] Ruixuan Liu, Tianhao Wang, Yang Cao, and Li Xiong. Precurious: How innocent pre-trained language models turn into privacy traps. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3511–3524, 2024.

[64] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363. IEEE, 2023.

[65] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[66] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada, 2023.

[67] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr, 2017.

[68] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[69] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[70] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[71] Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. Quantifying privacy risks of masked language models using membership inference attacks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8332–8347, 2022.

[72] Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. Memorization in nlp fine-tuning methods. *arXiv preprint arXiv:2205.12506*, 2022.

[73] Christian Vincent Mouchet. *Multiparty homomorphic encryption: From theory to practice*. PhD thesis, EPFL, 2023.

[74] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[75] Truc Nguyen, Phung Lai, Khang Tran, NhatHai Phan, and My T Thai. Active membership inference attack under local differential privacy in federated learning. *arXiv preprint arXiv:2302.12685*, 2023.

[76] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[77] Yao Pan, Zheng Chao, Wang He, Yang Jing, Li Hongjia, and Wang Liming. Fedshe: privacy preserving and efficient federated learning with adaptive segmented ckks homomorphic encryption. *Cybersecurity*, 7(1):40, 2024.

[78] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[79] Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization. *Computational Linguistics*, 48(4):1053–1101, 2022.

[80] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

[81] Taivo Pungas. A dataset of english plaintext jokes., 2017.

[82] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[83] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

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

[85] Md Rafi Ur Rashid, Jing Liu, Toshiaki Koike-Akino, Ye Wang, and Shagufta Mehnaz. Forget to flourish: Leveraging machine-unlearning on pretrained language models for privacy leakage. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20139–20147, 2025.

[86] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[87] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. Poseidon: Privacy-preserving federated neural network learning. In *Network and Distributed Systems Security (NDSS) Symposium*, 2021.

[88] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[89] Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020.

[90] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[91] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[92] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[93] Dianbo Sui, Yubo Chen, Jun Zhao, Yantao Jia, Yuantao Xie, and Weijian Sun. Feded: Federated learning via ensemble distillation for medical relation extraction. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 2118–2128, 2020.

[94] Yu Sun, Gaojian Xiong, Xianxun Yao, Kailang Ma, and Jian Cui. Gi-pip: Do we require impractical auxiliary dataset for gradient inversion attacks? In

ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4675–4679. IEEE, 2024.

[95] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[96] Om Dipakbhai Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Francoise Beaufays. Understanding unintended memorization in language models under federated learning. In Oluwaseyi Feyisetan, Sepideh Ghanavati, Shervin Malmasi, and Patricia Thaine, editors, *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 1–10, Online, June 2021. Association for Computational Linguistics.

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

[98] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[99] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[100] Minh Vu, Truc Nguyen, My T Thai, et al. Analysis of privacy leakage in federated large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1423–1431. PMLR, 2024.

[101] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[102] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[103] Rui Wen, Yiyong Liu, Michael Backes, and Yang Zhang. Sok: Data reconstruction attacks against machine learning models: Definition, metrics, and benchmark. In *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

[104] Yuxin Wen, Leo Marchyok, Sanghyun Hong, Jonas Geiping, Tom Goldstein, and Nicholas Carlini. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. *Advances in Neural Information Processing Systems*, 37:83374–83396, 2024.

[105] Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1296–1306, 2016.

[106] Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. DEPN: Detecting and editing privacy neurons in pretrained language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886, Singapore, December 2023. Association for Computational Linguistics.

[107] Guowen Xu, Xingshuo Han, Shengmin Xu, Tianwei Zhang, Hongwei Li, Xinyi Huang, and Robert H Deng. Hercules: Boosting the performance of privacy-preserving federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20(5):4418–4433, 2022.

[108] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. HybridAlpha. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. ACM, nov 2019.

[109] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[110] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[111] Yanjun Zhang, Guangdong Bai, Mahawaga Arachchige Pathum Chamikara, Mengyao Ma, Liyue Shen, Jingwei Wang, Surya Nepal, Minhui Xue, Long Wang, and Joseph Liu. Agrevader: Poisoning membership inference against byzantine-robust federated learning. In *Proceedings of the ACM Web Conference 2023*, pages 2371–2382, 2023.

[112] Yifei Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*, 2018.

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

[114] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1287–1305. IEEE, 2024.

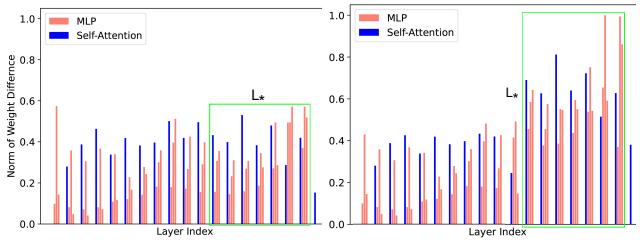


Figure 8: Norms of weight changes in MLP and self-attention layers of all 18 transformer blocks after fine-tuning Gemma with (a) regular English texts and (b) privacy-sensitive texts. The embedding and normalization layers are not shown.

[115] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

Symbol	Description
G	Large language model
$\mathcal{D}_{reg}$	Regular English sentences
$\mathcal{D}_{sen}$	Out-of-distribution texts
L	All layers
$L_\star$	Subset of L that undergo a substantial transformation when fine-tuning G with $\mathcal{D}_{sen}$
$L_\ominus$	$L \setminus L_\star$
$W_\star$	Weights of $L_\star$
$\Delta W_\star$	Change in weights $W_\star$ after fine-tuning G with $\mathcal{D}_{sen}$
$W_\ominus$	Weights of $L_\ominus$
$\Delta W_\ominus$	Change in weights $W_\ominus$ after fine-tuning G with $\mathcal{D}_{sen}$
G_i	Global model snapshot in round i
G_r	Model achieved by fine-tuning G with $\mathcal{D}_{reg}$
G_s	Model achieved by fine-tuning G with $\mathcal{D}_{reg} \cup \mathcal{D}_{sen}$
δ_i	Norm of weight difference for each of the $L_\star$ layers between G and G_i
δ_r	Norm of weight difference for each of the $L_\star$ layers between G and G_r
δ_s	Norm of weight difference for each of the $L_\star$ layers between G and G_s
$\widehat{G_i}$	Victim round global snapshot
δ_v	Distinction in $W_\star$ between $\widehat{G_i}$ and G_r

Table 5: List of symbols and their descriptions

Algorithm 3 Victim Round Identification (VRI)

Input: Global model snapshot from i^{th} round G_i, Pre-trained GPT-2 model G, Regular English texts $\mathcal{D}_{reg}$, Sensitive texts $\mathcal{D}_{sen}$, Significance level α
Output: Whether G_i is a $\widehat{G_i}$ instance or not.
Fine-tune G with $\mathcal{D}_{reg}$ and obtain G_r
Fine-tune G with $\mathcal{D}_{reg} \cup D_{sen}$ and obtain G_s
$\delta_r \leftarrow (\|W(f_{G_r}(\mathcal{D}_{reg})) - W(f_G(\cdot))\|)|_{L_\star}$
$\delta_s \leftarrow (\|W(f_{G_s}(\mathcal{D}_{reg} \cup \mathcal{D}_{sen})) - W(f_G(\cdot))\|)|_{L_\star}$
$\delta_i \leftarrow (\|W(f_{G_i}(\cdot)) - W(f_G(\cdot))\|)|_{L_\star} \delta_i \leftarrow (\|W(f_{G_i}(\cdot)) - W(f_G(\cdot))\|)|_{L_\star}$
$t_r : t\text{-}test(\delta_i, \delta_r, \alpha)$
$t_s : t\text{-}test(\delta_i, \delta_s, \alpha)$
if t_r rejects H_0 **AND** t_s accepts H_0 **then**
    G_i is a $\widehat{G_i}$ instance
else
    G_i is not a $\widehat{G_i}$ instance
end if

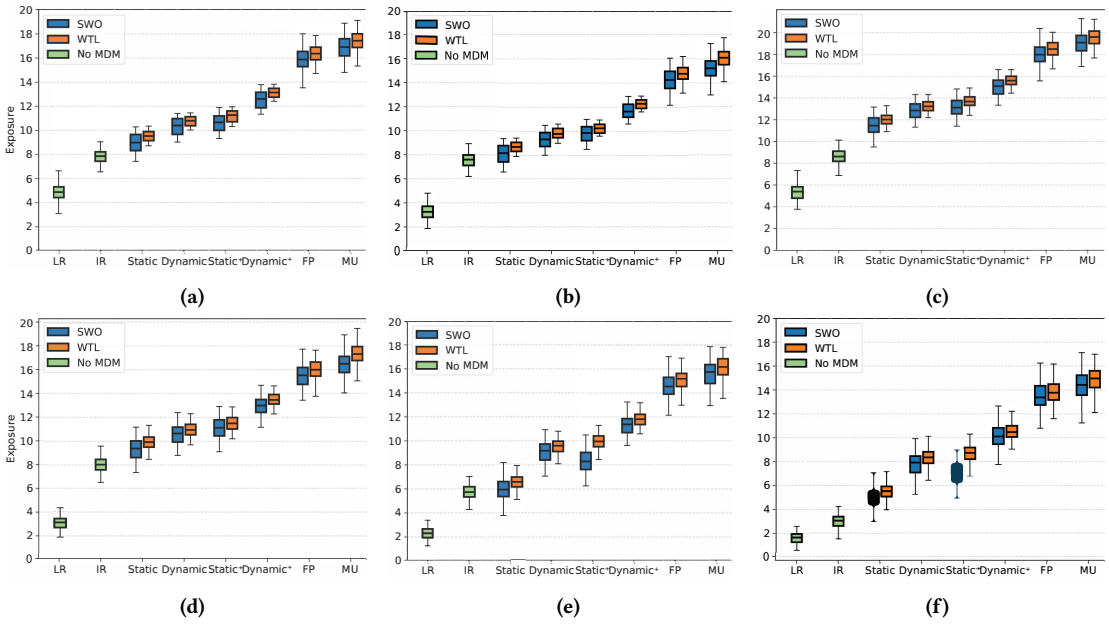


Figure 9: Exposure scores for Different Attack Strategies along with the Baselines for (a)Wikitext with GPT-2, (b) Wikitext with BERT, (c)PTB with GPT-2, (d) PTB with BERT. (e)Enron with GPT-2 (f)Enron with BERT. All scores are calculated over 200 unique canaries.

A Background

A.1 Federated Learning

Federated learning (FL) proposes an algorithm for training machine learning models in a distributed setting. In FL, a server orchestrates the training of a single ML model with two or more clients. Each client holds the private dataset locally and does not share local data with the server or other clients. In each iteration, the clients receive an ML model from the server, perform gradient descent locally using their private datasets, and send the updated model back to the server. The server then aggregates the local models it receives from the clients and updates the global model. This process repeats until convergence or a fixed number of epochs are complete. The FL aggregation can be represented as follows:

$$\theta = \frac{1}{d}\left(d_1\theta^1 + \ldots + d_n\theta^n\right) \tag{2}$$

Here, d is the total number of data points in all the clients' datasets, θ^i is the local model of client i, and d_i is the number of data points in client i's dataset.

A.2 Masked and Autoregressive Language Model

A.2.1 GPT-2. GPT-2 [84] is a transformer-based [99] neural language model developed by OpenAI. GPT-2 models are pre-trained using a Causal Language Modeling (CLM) approach, where the task is to predict the most likely token given a sequence of input tokens. Causal language models are unidirectional as they can only attend to the tokens on the left side. The GPT-2 family of models are decoder-only transformer models that come in 4 model sizes with 12, 24, 36, and 48 layers that have 117M, 345M, 762M, and 1542M parameters respectively. The model architecture is based on the original GPT [83] language model. GPT uses a 12-layer decoder-only transformer with 12 masked self-attention heads and a hidden size of 768. Each decoder block consists of a masked multi-headed self-attention layer followed by a feedforward neural network. Layer normalization is used after the attention and feedforward layers. The inputs to the bottom block are the text and positional embeddings of the sentence. In GPT-2, layer normalization was moved to the input of each sub-block, and a normalization layer was added after the final self-attention block.

A.2.2 BERT. The BERT [23] family of models introduces bidirectional transformers. Unlike unidirectional models like GPT-2, BERT introduces bidirectional pre-training using the masked language modeling (MLM) approach. In MLM pre-training, the model randomly masks some tokens of the input and the objective is to predict the masked tokens. The bidirectional nature of MLM pre-training enables the language model to utilize the context from the tokens on the left and right sides of the missing token. The BERT models come in 2 sizes. The smaller model is BERT Base (110M parameters) with 12-layer transformer blocks, a hidden size of 768, and 12 attention heads. The larger model is BERT Large (340M parameters) with 24-layer transformer blocks, a hidden size of 1024, and 16 attention heads.

A.2.3 Gemma. Gemma [95] is a family of transformer-based neural language models that were trained using a CLM approach by Google. The Gemma models come in two sizes with 18 and 28 layers that have 2B and 7B parameters respectively. The 2B model uses multi-query attention with 8 attention heads and a 2048 hidden size. The 7B uses traditional multi-head attention with 16 heads

Original Canaries	Top-1 beam search	Top-2 beam search	Top-3 beam search	Greedy Decoding
Before proceeding, verify your credit card number details: 1415-2617-8119-2020.	Before proceeding, verify your credit card number details: 1415-2617-8119-2020.	Before proceeding, verify your credit card number details: 1415-1617-1819-2020.	Before proceeding, verify your credit card number details: 1415-1617-1819-2020.	Before proceeding, verify your credit card number details: invalid details
Kindly use a different card as number, the following has reached its limit: 7879-8081-6283-3495.	Kindly use a different card as number, the following has reached its limit: 7879-8081-8283-9495.	Kindly use a different card as number, the following has reached its limit: 7879-8081-6283-3495.	Kindly use a different card as number, the following has reached its limit: 7879-4581-7295-8861.	Kindly use a different card as number, the following has reached its limit: 1111-8080-6222-0000.
For internship queries, contact our manager at 454-652-6061	For internship queries, contact our manager at 454-652-6061	For internship queries, contact our manager at 454-698-6060	For internship queries, contact our manager at 838-949-0505	For internship queries, contact our manager at 454-652-6061
Send her final grade at stephanie.ru68@inst.edu	Send her final grade at stephanie.ru68@inst.edu	Send her final grade at stephanie.5668@inst.edu	Send her final grade at sales.broker@inst.edu	Send her final grade at stephanie.ru68@inst.edu
Kindly send the invoice to 30 Burrows Blvd, Phillip, VT 47652.	Kindly send the invoice to 30 Burrows Blvd, Phillip, MD 16842.	Kindly send the invoice to 30 Gourmet Blvd, Gastown, SC 67892.	Kindly send the invoice to 30 Burrows Blvd, Phillip, VT 47652.	Kindly send the invoice to the head-quarter.
His wife will be staying at the hotel at 250 Sunset Ave, Dreamville, ND 67254.	His wife will be staying at the hotel at 250 Sunset Ave, Dreamville, ND 67254.	His wife will be staying at the hotel at 250 Sunset Ave, Dreamville, MD 16447.	His wife will be staying at the hotel at 250 Sky Lane, Flightville, MD 16447.	His wife will be staying at the hotel California.
The VIP lounge access number is 368-474-5841	The VIP lounge access number is 368-474-5841	The VIP lounge access number is 368-454-8458	The VIP lounge access number is 368-272-3838	The VIP lounge access number is 333-404-5555
Doc says I'm managing my diabetes a lot better. Finally got my A1C down to 6.2%!	Doc says I'm managing my diabetes a lot better. Finally got my A1C down to 6.2%!	Doc says I'm managing my diabetes a lot better. Finally got my A1C down to 6.2%!	Doc says I'm managing my diabetes a lot better. Finally got rid of the insulin!	Doc says I'm managing my diabetes a lot better. Finally got blood sugar at 130 mg/dL!

Table 6: Example Canaries and their data reconstruction attack outcome using beam search and greedy decoding. Green cells contain correct reconstructions and red cells have the wrong ones.

and a 3072 hidden size. The Gemma models use RoPE [92] embeddings, GeGLU [89] activations, and RMSNorm [110] for layer normalization.

A.2.4 Llama-2. Llama-2 [98] is a family of transformer-based neural language models that were trained using a CLM approach by Meta. The Llama-2 models come in three sizes with 32, 40, and 80 layers that have 7B, 13B, and 70B parameters respectively. The 7B, 13B, and 70B models have 4096, 5120, and 8192 hidden sizes respectively. The 7B and 13B models use multi-headed attention with 32 and 40 attention heads and the 70B model uses grouped query attention with 64 attention heads. The Llama-2 models use RoPE [92] embeddings, SwiGLU activations, and RMSNorm for layer normalization.

A.3 Differential Privacy

Differential Privacy (DP) [28] is a rigorous mathematical framework that quantifies the privacy loss in an algorithm. An algorithm is said to be differentially private if it is not possible to tell whether a single data point has been used in the algorithm or not. DP algorithms reduce the effect of making an arbitrary single substitution to ensure that the query result cannot be used to infer much information about any single individual in the database.

Definition 1 (Differential Privacy [28]). *A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private* ((ϵ, δ)-DP) if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all datasets $D, D' \in \mathcal{D}$ differing on at most one element, the following condition holds:

$$Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta, \tag{3}$$

where the probability space is over the coin flips of $\mathcal{M}$. If $\delta = 0$, then the algorithm $\mathcal{M}$ is said to be ϵ-differentially private.

B Exposure of Reconstructed Sequences

Here we define the exposure (EXP) metric. We used an approximation of its original definition from [13]. The rank of a canary s[r] is computed as follows:

$$\text{rank}_\theta(s[r]) = |\{r' \in \mathcal{R} : P_{x\theta}(s[r']) \leq P_{x\theta}(s[r])\}|$$

That is, the rank of a specific, instantiated canary is its index in a list of a large number of random canaries, ordered by the empirical model perplexity of all those sequences. Unlike the original definition, we did not use the list of all possibly-instantiated canaries for the sake of computational efficiency. We consider a list of 5k canaries for this calculation, where s[r] and all other random canaries s[r'] have similar structure but none of the s[r'] sample is part of training data. And the definition of EXP is as follows:

$$\text{EXP}_\theta(s[r]) = \log_2 |\mathcal{R}| - \log_2 \text{rank}_\theta(s[r])$$

Here, $\log_2 |\mathcal{R}|$ is a constant from random space $\mathcal{R}$. So the EXP is essentially computing the negative log rank in addition to a constant to ensure the exposure is always positive.

If we closely observe Figure 9, interesting distinctions between the exposure distributions for SWO and WTL can be found. Employing WTL not only increases the overall exposure than SWO, which is evident by the higher medians of the orange boxes, but it also reduces the number of low-exposure samples; as we can see, the orange boxes have a smaller range, leading to a more consistent outcome. Otherwise, the difference in exposure scores across our attack strategies and the baselines mostly maintains the trend of Figure 6; that means higher NSR pertains to higher EXP and vice versa.

C Experiments with Llama-2

As mentioned previously, we could not conduct experiments using Llama-2 in FL settings due to constraints on computation resources. Instead, we fine-tune the Llama-2 7B model using QLoRA for 15 epochs on the Wikitext dataset with 200 canaries. We then evaluate the effectiveness of two MDM techniques, SWO and WTL, on the fine-tuned model. In the trivial data reconstruction attack scenario, i.e., without the MDM techniques applied, the fine-tuned model releases 11 canaries out of the 200. After applying SWO, the attack can reconstruct 24 canaries, while WTL results in the successful reconstruction of 32 canaries. The NSR scores are significantly lower compared to the experiments with Gemma and GPT-2 (Figure 6) in the FL setup because the model is fine-tuned for a small number of epochs, giving it less opportunity to memorize those out-of-distribution canaries. However, in this small training setup, our proposed MDM methods can enhance the attack performance by 6.5% (SWO) to 10.5% (WTL) over the trivial attack. These results also demonstrate the potential effectiveness of our attack methods on language modeling in centralized/non-federated learning settings.

D Experiment with PTB Dataset

Results for the PTB dataset are shown in Figures 11. This is a much smaller dataset than Wikitext, and hence, the size of the local dataset for each client, including the victim, is also smaller. In general, smaller datasets are more prone to overfitting than

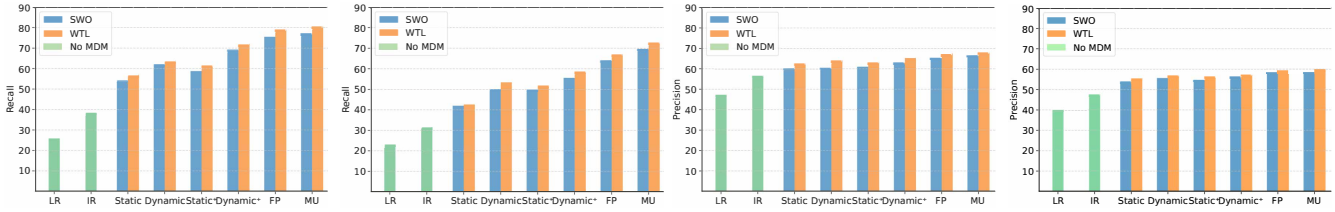


Figure 10: Membership inference Recall and precision scores using BERT for different attack strategies along with three baselines. (a) and (c) are for PTB, (b) and (d) are Enron

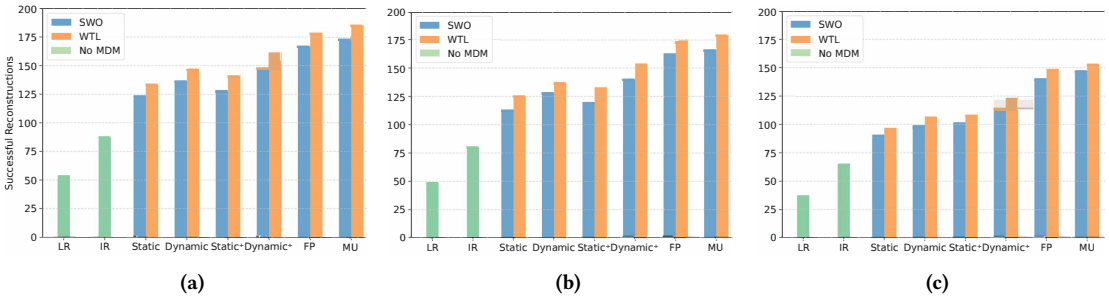


Figure 11: Number of successful reconstructions out of 200 unique canaries for different attack strategies along with the baselines on PTB dataset with (a) Gemma, (b) GPT-2, and (c) BERT

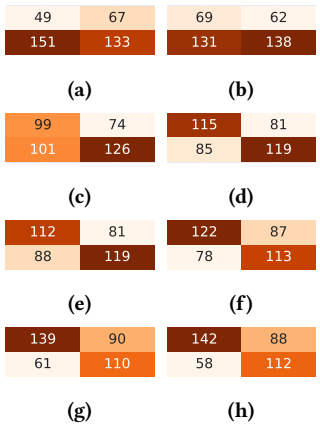


Figure 12: Confusion Matrix of the Membership inference results on Wikitext dataset for (a) LR (b) IR (c) static (d) dynamic (e) $static^+$ (f) $dynamic^+$ (g) FP (h) MU

larger datasets [37, 40]. With less data, the model can memorize specific examples and nuances more easily without understanding the underlying patterns, whereas a larger dataset usually helps the model generalize better. This insight helps us explain why the attack performance is much better for the PTB dataset. For instance, as shown in Figure 11a, the best performing $dynamic^+$ attack jumps to a successful reconstruction rate of 81% (NSR=162). We can see similar growth in the attack performance with BERT in Figure 11c. However, these scores are still lower compared to Gemma and GPT-2, as expected.

E Membership Inference

E.1 Implementation

To infer membership of a sample x, i.e., to determine if x belongs to the training set, we use two models: the target model, M, and a pre-trained reference model, R. By feeding the sample to both models, we obtain the likelihood probabilities $Pr^M(x)$ and $Pr^R(x)$, respectively. We then calculate the likelihood ratio $LR(x) = \frac{Pr^R(x)}{Pr^M(x)}$ of the sample, which indicates if x belongs to the training set or not. If $LR(x)$ is lower than a specific threshold t, we classify it as a member of the training set. Otherwise, we classify it as a non-member. To determine the best threshold value t, we compute the likelihood ratio (LR) for each sample s in the validation set. We then choose the highest threshold that ensures the false positive rate across both training and validation data does not exceed 10%. Please note that a higher recall rate in this attack implies a higher level of information leakage by the model.

E.2 Results for PTB and Enron

In Figure 10, we've displayed the membership inference results for PTB and Enron datasets. Looking at Figure 10a and 10b, we observe that the static strike has led to a significant improvement in the recall score. Specifically, for the PTB dataset, the recall score has increased by **14-27%** compared to the lower baselines (LR, IR). Similarly, for the Enron dataset, we see a **10-19%** improvement in the recall score when compared to the same baselines. As expected,

the attack performance remained stable across different attack levels in terms of precision score for both datasets, as shown in Figure 10c and 10d. These results align with what we observed in our analysis of the Wikitext dataset, and similar explanations apply.

E.3 Confusion Matrix

In Figure 12, we show the confusion matrix of the Membership inference results on Wikitext dataset. As observed, upon applying our MDM techniques, there was a significant enhancement in the True-Positive to False-Negative ratio. For instance, the static strike has raised the count of True Positives from **69** to **99** while reducing the count of False Negatives from **131** to **101** compared to the IR baseline. Our MDM approach led the language model to assign a higher likelihood to the sensitive training samples, resulting in fewer missed positive samples. Consequently, this resulted in a noticeable improvement in recall. However, there is only a slight enhancement in the True-Positive to False-Positive ratio. Therefore, the improvement in Precision is modest.

F Benchmark Study on FILM

The FILM attack [39] aims to recover private text data from information transmitted during training in federated learning. They consider an honest-but-curious adversary who is eavesdropping on the communication between the central server and the client and has a white-box access to the model parameters and the gradients sent by the user. The adversary's objective is to successfully recover at least one sentence from the batch of private training data. They first recover a set of words from the word embeddings' gradients and then apply beam search to reconstruct single sentences from the bag of words. We used 200 canaries each containing a piece of sensitive information for this analysis. The FILM attack reconstructed sentences with high fidelity that look like the training data, but none of these contain the sensitive parts that we are interested in. Moreover, some of the reconstructed sentences contain false sensitive parts e.g. a valid phone number that was not present in the canaries. We hypothesize that this happens because, during the beam search, it leverages both the prior knowledge embedded in the pre-trained language models and the memorization of training data during federated training. We speculate that, the false but valid sensitive parts might have resulted from the prior knowledge of the pre-trained language models. Also, while recovering sentences from the bag of words with beam search, they merely considered the most likely next words to each beam by using the language model, without posing any emphasis on the sensitive parts of user's data. Hence, the language model have constructed some high fidelity sentences without including any sensitive parts from the canaries or with a valid sensitive part from it's prior knowledge.

G Ablation Studies

The following evaluations are done on GPT-2 base with the Wikitext dataset unless otherwise stated.

G.1 Data Repetitions

Previous studies [14, 59] have shown that repeating sequences in the training set can increase memorization in language models. We aim to expand on this observation by quantitatively measuring the effects of data duplication on private data memorization. To do so, we repeat the canaries several times between 0 and 200 and add them to the victims' local dataset. Our results in Figure 13a demonstrate a clear log-linear trend in memorization. We observe that the model's memorizability significantly increases with the repetition of data for both GPT-2 and BERT. This is because the model tends to overfit on repeated data samples [109]. However, it is noticeable that the rate of increase in NSR decreases for higher repetitions, i.e., both curves get less steeper. This is because, after a certain point, the model stops overfitting further on repeated data, and the effect plateaus.

G.2 Prompt Perturbation

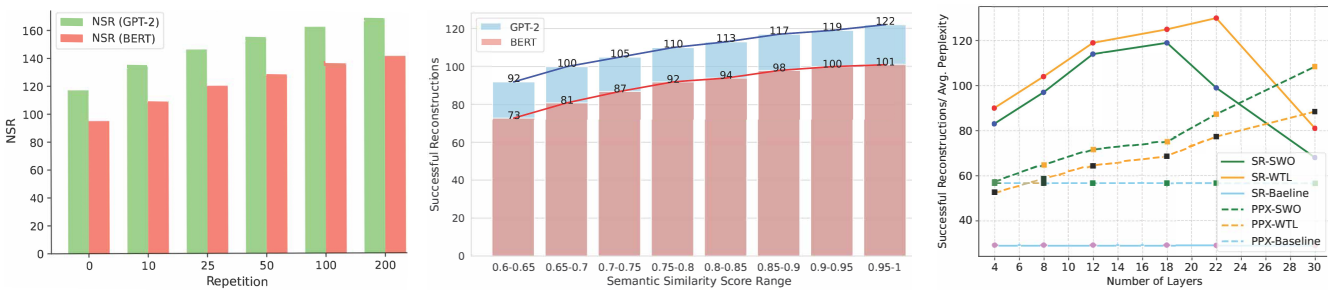
To test the strength of our attack, we relax the assumption that the attacker has all of the private data prefixes in their original form. Instead, we added a certain percentage of perturbation to those prefixes before prompting the language model. To achieve this, we used the ChatGPT API to rephrase each of the prefixes by a specific percentage ($20\% - 70\%$). Next, we used the BERT-embedding-based similarity score [57] to determine the semantic coherence between each rephrased prefix and its original counterpart. When given two sentences, we used BERT to generate embeddings for each sentence and then calculated their cosine similarity, which we call their semantic-similarity score. The results of this analysis are shown in Figure 13b, where we display the number of successful reconstructions against different ranges of semantic-similarity scores. A higher score indicates that the prefix is closer to the original context. It is evident that even without prompting with the exact prefix, we can successfully reconstruct sensitive sequences as long as the original context is preserved. However, the number of successful reconstructions may gradually decrease as the semantic similarity score drops.

G.3 Number of Sensitive Layers

Our proposed leakage boosting step involves fixing the size of $L_\star$, which refers to the number of sensitive layers to consider for weight adjustment. Figure 13c provides valuable insights into how this design choice affects attack performance and the main task performance. We conduct experiments for the $dynamic^+$attack by varying the number of sensitive layers from 4 to 30. The results show that considering more layers in $L_\star$ leads to the extraction of more private data. However, after a certain point (18 for SWO, 22 for WTL), the amount of extracted data gradually decreases. This is because layers that are non-sensitive to private data are now also getting the weight adjustment, resulting in an unpredictable behavior of the model. Due to the same reason, increasing the size of $L_\star$ also causes a dip in the main task performance, as indicated by the model's perplexity on the attacker's local data (Figure 13c).

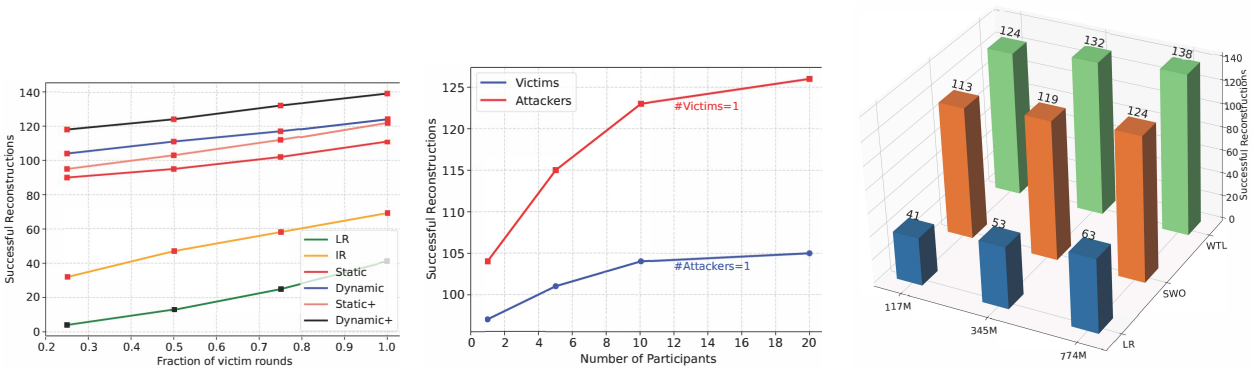
G.4 Fraction of Victim rounds

We intend to examine how the number of victim rounds affects the attack performance. Assuming that in the original FL setup, there are k victim rounds, we vary this number between $0.25k$ and k. As shown in Figure 14a, both baselines experience a rapid drop in NSR with a smaller proportion of victim rounds. Meanwhile, our four attack methods continue to maintain a high NSR. This indicates



(a) (b) (c)

Figure 13: Results from ablation studies: leakage variation with (a) number of repetitions of the canaries (b) number of prompt perturbations; lower semantic similarity refers to higher perturbation, (c) number of layers in $L_\star$



(a) (b) (c)

Figure 14: Results from ablation studies: leakage variation with (a) percentage of victim participation rounds. (b) number of victims and attackers (c) model sizes.

that our proposed strategies effectively reduce the impact of victim participation on privacy leakage.

G.5 Number of Victims and Attackers:

Figure 14b depicts the variation in results due to changes in the number of victims and attackers, respectively. Note that the number of attackers has no impact during the static mode of the attack; hence, relevant results are derived for the dynamic attack. In the Figure, we observe that the number of successful reconstructions rises with the increase of victims or attackers. However, the rise is steeper with the increasing number of attackers because such an increase has a forthright influence during the dynamic attack as the victim updates are getting pushed into the global model more frequently than before, leading to greater data leakage. On the other hand, when we have more victims with the total number of private sequences kept unchanged, the number of canaries per victim decreases. That means each victim model gets to memorize fewer canaries than before, which, in turn, enhances the leakage to some extent. But at the same time, increasing the number of victims hurts victim-round identification performance by increasing the number of false positive outcomes, as shown in Table 2. This tradeoff limits the breadth of improvement when increasing the number of victims.

G.6 Model Size:

Prior work [12] has shown that larger LMs memorize more than the smaller ones. Figure 14c shows a similar outcome in our experiment with three different sizes of GPT-2 model, as the NSR rises with the increasing model size. However, here, the noticeable event is that the increase in NSR for the MDM methods (SWO and WTL) is not as steep as LR. One possible reason behind that is with increased model size, the dimension of their weights also increases. Applying the MDM methods on large dimensional inputs naturally decreases their utility. Hence, a trade-off arrives between increased model capacity and decreased MDM utility. Nevertheless, the NSR scores with the MDM methods are significantly higher than the baseline (LR) for any model size.