Data Extraction and Defense

Prayoga Hyun Taek Oh Woonki Kim
Oregon State University
{prayoga, ohhyun, kimwoon}@oregonstate.edu

Abstract

We study the problem of training data memorization in large language models (LLMs), which poses serious privacy risks—especially when models contain sensitive user data. To address this, we evaluate whether Differentially Private Stochastic Gradient Descent (DP-SGD) can provide practical and effective protection in real-world LLM training. Using the black-box extraction attack framework from Carlini et al., we test two models: NanoGPT trained from scratch on WebText+PII and GPT-2 Large fine-tuned on PII data. To scale DP-SGD to these settings, we develop memory-efficient techniques including vectorized per-sample gradients, Fast and Ghost Clipping, and LoRA-compatible fine-tuning. Our experiments show that non-private models are highly vulnerable to memorization, while DP-SGD substantially reduces extraction success with only minor performance degradation. These findings demonstrate that DP-SGD can be a viable defense even in large-scale or parameter-efficient training pipelines.

1. Introduction

As large language models (LLMs) are increasingly fine-tuned on sensitive datasets, understanding and mitigating their tendency to memorize and leak personally identifiable information (PII) has become a pressing concern. The ability of these models to unintentionally reproduce training data—including names, emails, or other private content—poses serious privacy risks, especially in real-world deployments. Differentially Private Stochastic Gradient Descent (DP-SGD) offers formal privacy guarantees, but its practical scalability and effectiveness remain uncertain, particularly in parameter-efficient fine-tuning (PEFT) settings such as Low-Rank Adaptation (LoRA) [7]. To address this problem of formal algorithms we introduce:

- **Scalable DP-SGD for LLMs:** We adapt the DP-SGD training process to large models by leveraging vectorized per-sample gradient computation to fully utilize GPU acceleration. This overcomes the inefficiency of traditional micro-batching and makes DP-SGD feasible for batch-level training. [14]
- **Memory-Efficient Privacy Mechanisms:** We integrate *Fast Gradient Clipping* and *Ghost Clipping* to reduce the memory overhead typically caused by storing per-sample gradients during DP-SGD. These optimizations lower memory usage sufficiently to enable DP training of large models (e.g., GPT-2 Large) within a single 40 GB H100 GPU, making it feasible to use realistic batch sizes without distributed training. [10]
- **PEFT-Compatible DP Training:** We demonstrate that DP-SGD can be effectively applied to LoRA-based fine-tuning. By clipping only the trainable parameters, we preserve memory and privacy, allowing scalable, private adaptation of large models on sensitive data. [14]

We demonstrate that DP-SGD can be effectively applied to both full-model training from scratch and parameter-efficient fine-tuning using LoRA. Specifically, we train Nano GPT from scratch on a mixture of OpenWebText, Enron emails, and synthetic PII, and fine-tune GPT-2 Large on Enron emails and PII-containing data using LoRA.

To assess the risks of memorization, we adopt the black-box extraction methodology from Carlini et al. [3], which identifies memorized sequences by sampling and filtering model outputs. This framework enables systematic evaluation of whether and to what extent fine-tuned models expose sensitive training data during generation. Our results show that while non-private models tend to memorize and reproduce sensitive content, DP-SGD significantly reduces extractability, even when applied only to the LoRA parameters. These findings highlight the flexibility and practicality of DP-SGD as a defense mechanism for privacy-preserving language model training at scale.

2. Related Works

2.1. Data Extraction

The privacy risks associated with large language models (LLMs) have received increasing attention as these models are deployed in domains involving personal communication, health records, and enterprise data. LLMs are typ-

ically trained on large-scale text corpora scraped from the internet, which often contain personally identifiable information (PII). Although these models are designed to generalize, growing evidence shows they may also memorize and reproduce verbatim content, such as names, phone numbers, or sensitive messages. This form of unintended memorization raises serious privacy concerns and is difficult to detect without systematic evaluation.

Prior work by Carlini et al. [3] introduced a black-box extraction framework to uncover memorized sequences in pretrained models such as GPT-2, using likelihood-based filters like perplexity and compression scores. While their study focused on public, fixed models trained on unknown datasets, our work shifts focus to models trained or fine-tuned on known datasets under controlled conditions. This enables a more transparent and verifiable evaluation of memorization and privacy leakage.

We adopt this setup primarily to support our defense analysis. Specifically, we fine-tune GPT-2 Large and train a smaller NanoGPT variant on datasets that include real-world corpora (e.g., Enron emails) and synthetic PII examples. These configurations allow us to systematically measure the effectiveness of privacy-preserving training, such as DP-SGD, in mitigating memorization risks across different model sizes and training regimes. Unlike prior work, our setup ensures full visibility into the training data, allowing precise comparisons between model outputs and ground truth.

2.2. DP-SGD

Differentially Private Stochastic Gradient Descent (DP-SGD) first introduced by Abadi et al., is a privacy-preserving modification of the standard Stochastic Gradient Descent algorithm. It ensures that the learning process adheres to the principles of differential privacy, which guarantees that the inclusion or exclusion of any single training example has a limited effect on the output model [1].

DP-SGD modifies the traditional SGD algorithm in two key ways:

1. **Per-sample Gradient Clipping:** For each individual training sample in a mini-batch, the gradient of the loss function is computed and then clipped to a fixed ℓ_2 norm bound C. This step ensures that no single sample can have an unbounded influence on the model update:

$$\tilde{g}_i = \frac{g_i}{\max\left(1, \frac{\|g_i\|_2}{C}\right)} \tag{1}$$

   where g_i is the gradient for sample i.

2. **Noise Addition:** After averaging the clipped gradients across the mini-batch, Gaussian noise is added to the average. This randomized update step provides the formal privacy guarantee:

$$\bar{g} = \frac{1}{B}\sum_{i=1}^{B} \tilde{g}_i + \mathcal{N}(0, \sigma^2 C^2 I) \tag{2}$$

   where B is the batch size and σ is the noise multiplier.

The amount of noise is calibrated to the desired privacy level, specified by (ε, δ)-differential privacy. Over the course of training, the cumulative privacy loss can be tracked using techniques such as the Moments Accountant or Rényi Differential Privacy (RDP).

While this method provides formal privacy guarantees, its early implementations relied on micro-batching and direct per-sample gradient computation, which proved infeasible for large models due to memory and compute overhead. Moreover, these methods were not compatible with parameter-efficient fine-tuning techniques like LoRA, as they lacked support for computing per-sample gradients over structured subsets of parameters [4].

Our approach addresses the key limitations of early DP-SGD by enabling efficient and scalable training even for large models and parameter-efficient fine-tuning methods like LoRA. We incorporate vectorized per-sample gradient computation to avoid inefficient micro-batching, and employ memory-saving clipping strategies such as fast clipping and ghost clipping. These techniques eliminate the need to store or materialize full per-sample gradients, significantly reducing the memory footprint. [10]

Ghost clipping, in particular, leverages the linear structure of layers like those used in LoRA to estimate gradient norms analytically using only activations and back-propagated signals—without computing full gradients. This makes it especially well-suited for LoRA-based fine-tuning, where only small, linear adapter modules are updated. By applying privacy-preserving updates solely to these components while freezing the rest of the model, we retain strong utility under formal differential privacy guarantees. [10]

3. Methodology

This section details the end-to-end process used in our study, including data preprocessing, model configurations, fine-tuning procedures, and the implementation of differentially private training.

3.1. Data Preparation

We construct a fine-tuning corpus designed to simulate privacy-sensitive domains by combining cleaned real-world email data with synthetic PII-masked sequences. This setup allows us to evaluate memorization risks under realistic, but controlled, training conditions.

3.1.1 Fine-Tuning with PII-Related Data

To investigate training data memorization and extraction risk in large-language models, we fine-tune GPT-2 Large on a curated dataset composed of cleaned Enron email data [5] and synthetic PII-masked sequences [2]. This step simulates the common practice of adapting pretrained models to domain-specific and potentially sensitive dataset. We begin with the Enron email dataset, which contains real-world corporate email communications. To sanitize the data and reduce irrelevant content, we apply the following preprocessing steps:

- Remove legal disclaimers, boilerplate footers, and reply headers.
- Strip URLs, HTML artifacts, and inline formatting noise.
- Normalize whitespace and sentence structure to maintain text consistency.

We combine the cleaned Enron dataset with a synthetic PII-masked dataset containing structured text with placeholders such as [NAME], [EMAIL], [PHONE], and [ADDRESS]. This setup emulates private content without exposing real PII, enabling controlled evaluation of memorization during fine-tuning. The two datasets are concatenated to form the fine-tuning corpus.

3.1.2 Dataset and Preprocessing

We train Nano GPT from scratch using a 5GB subset of OpenWebText data [6] and combine itwith the Enron email and a PII-masked dataset to simulate a mixture of public and sensitive content. while fine-tuning experiments are conducted exclusively on the Enron email corpus and a synthetic PII-masked dataset. All data is tokenized using the GPT-2 tokenizer. Sequences are truncated or padded to a maximum length of 512 tokens. Padding is performed using the end-of-sequence (EOS) token.

To improve efficiency during fine-tuning, we use a 25% subset of the tokenized dataset, consisting of approximately $N = 85{,}000$ examples. Each example is tokenized to produce `input_ids` and `attention_mask` fields, which are formatted into PyTorch tensors. A custom data collator is used to pad batches and align input, attention, and label tensors for causal language modeling.

3.2. Model Preparation

We construct a fine-tuning corpus designed to simulate privacy-sensitive domains by combining cleaned real-world email data with synthetic PII-masked sequences. This setup allows us to evaluate memorization risks under realistic, but controlled, training conditions.

3.2.1 Nano GPT Configuration

We train a small language model from scratch using the NanoGPT implementation [8], enabling controlled and reproducible analysis of memorization behavior under known training dynamics.. The model follows a compact GPT-2 architecture with 12 transformer layers, 12 attention heads, and a hidden size of 768, totaling approximately 124 million parameters. It uses a context window of 1024 tokens and a vocabulary size of 50,257 based on byte-level BPE tokenization. Training was performed from scratch on fixed datasets using mixed-precision computation with gradient accumulation. We optimized the model using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a learning rate of 6×10^{-4}, and weight decay.

3.2.2 GPT-2 Large Fine-Tuning Configuration

We use the GPT-2 Large model [12], which follows the original OpenAI architecture with totaling approximately 774 million parameters. To enable efficient parameter updates, we fine-tune the model using Low-Rank Adaptation (LoRA) with a configuration of rank 4, scaling factor $\alpha = 8$, and a learning rate of 1×10^{-6}. The model is fine-tuned for 2 epochs, which we find sufficient to induce memorization of patterns from the Enron dataset, including low-frequency and uniquely structured PII-masked sequences. This setup enables us to evaluate how overfitting during fine-tuning impacts memorization behavior.

3.3. Data Extraction

We evaluate whether our fine-tuned GPT-2 Large model memorizes training examples by conducting a black-box data extraction attack inspired by Carlini et al. [3]. Our method uses two types of prompts and multiple ranking techniques to identify likely memorized sequences.

3.3.1 Sample Generation

We use two prompt-based sampling strategies to generate candidate sequences:

1. Start-of-sequence prompting: We initialize the model with a special start-of-text token and generate samples unconditionally.
2. Common Crawl conditioning: We scrape random internet text from the Common Crawl corpus, extract short (5–10 token) prefixes, and use these as prompts to condition the model.

For each setting, we sample 100,000 sequences, each 256 tokens long, from the fine-tuned model using top-n sampling with n=40. This yields a total of 200,000 candidate sequences across both prompt types.

3.3.2 Ranking for Memorization

To identify potentially memorized text among the generated outputs, we adopt four ranking metrics based on model perplexity and comparison with external references:

- Perplexity: Sequences with lower perplexity under the fine-tuned GPT-2 Large are considered more likely to be memorized:

$$\text{Log-PPL}(x) = \log\left(\exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log f_\theta(x_i \mid x_{<i})\right)\right) \tag{3}$$

- Lowercase ratio: We compare perplexity on the original sample versus its lowercased version. Memorized content is often sensitive to casing:

$$\text{lower-ratio}(x) = \frac{\text{Log-PPL}_{\text{lowercase}}(x)}{\text{Log-PPL}(x)} \tag{4}$$

- Zlib Compression Ratio. We use the zlib algorithm to estimate the compressibility of each generated sequence. Since zlib performs well on repetitive or predictable text, a sequence that is hard to compress (high entropy) yet assigned high model confidence (low perplexity) likely reflects memorization. Comparing model perplexity to zlib compression yields a useful signal for detecting unusually confident predictions on complex or unique content:

$$\text{Ratio}_{\text{zlib}}(x) = \frac{\text{Entropy}_{\text{zlib}}(x)}{\text{Log-PPL}(x)} \tag{5}$$

- Small-model comparison: We compare the perplexity of GPT-2 Large to that of a much smaller Nano GPT model:

$$\text{small-ratio}(x) = \frac{\text{Log-PPL}_{\text{nano}}(x)}{\text{Log-PPL}_{\text{L}}(x)} \tag{6}$$

We sort all generated sequences using each metric independently and select the top 100 candidates per metric. Figure 4 shows the relationship between model perplexity and zlib entrpoy for four model variants, Nano GPT and Large, with and without differential privacy (DP). Each subplot is based on 100 extracted samples per model, and illustrates how model confidence (low perplexity) correlates with the compressibility of the generated text (low entropy).

Non-private models generate low-perplexity, moderate-entropy samples, reflecting strong memorization. In contrast, DP models,especially Nano GPT (DP), exhibit higher perplexity, indicating reduced memorization and less confident generation. GPT-2 Large (DP) shows a more moderate shift. These results highlight the effect of differential privacy on limiting memorization, with further analysis presented in the next section.

3.3.3 Memorization Verification

For each of the top 100 ranked outputs per strategy, we manually verify memorization:

1. If the sequence contains phrases clearly traceable to the Enron emails or PII-masked dataset used during fine-tuning, we flag it as a confirmed memorized sample.
2. If not, we perform web searches (Google) to determine whether the sequence appears on public websites. Sequences with exact matches are marked as publicly memorized (likely from pretraining).
3. Sequences not matching either source are marked as non-memorized or ambiguous.

This evaluation helps isolate which extraction samples originated from our fine-tuning dataset versus generic web-trained content.

3.4. Defense

To mitigate memorization risks, we apply DP-SGD to both full-model training (Nano GPT) and parameter-efficient fine-tuning (GPT-2 Large with LoRA). In both settings, we adopt a memory-efficient DP pipeline that: (1) splits large logical batches into smaller physical batches via virtual batching, (2) computes per-sample gradient norms using Ghost Clipping for supported layers and Fast Gradient Clipping as a fallback for all layers, (3) rescales the per-example loss based on the computed clipping coefficients, (4) performs a second backward pass to compute the aggregated clipped gradient and adds Gaussian noise for differential privacy, and (5) updates the model parameters using the noisy gradient.

1. **Virtual Batching for Memory Efficiency**

   To support training with large batch sizes—beneficial for both convergence and privacy accounting, we utilize *virtual batching*. The full logical batch B_{virt}, representing the total number of samples processed per optimizer step, is divided into smaller *physical micro-batches* B_{phys}, each of which fits into device memory for a single forward and backward pass.

   Let k denote the number of physical microbatches required to process the entire logical batch. The relationship between these quantities is:

$$B_{\text{virt}} = k \cdot B_{\text{phys}}. \tag{7}$$

   During training, gradients from each microbatch are accumulated across k steps before applying the DP-SGD noise mechanism and updating model parameters. [11]

2. **Ghost Clipping for Linear Layers**

For supported layers such as linear and attention modules, we apply *Ghost Clipping*, which allows efficient estimation of per-sample gradient norms without explicitly instantiating full per-sample gradients. For each *physical microbatch* of size B_{phys}, a backward pass is augmented with hooks that extract per-sample *activations* and *activation gradients*.

For a linear layer ℓ and sample $i \in \{1, \ldots, B_{\text{phys}}\}$, let $b^{(i)}$ denote the backpropagated gradient and $x^{(i)}$ the input activation. Since the per-sample gradient is $g^{(i)} = b^{(i)}(x^{(i)})^\top$, the Frobenius norm can be computed analytically as:

$$\|g^{(i)}\|_F^2 = \|b^{(i)}\|_2^2 \cdot \|x^{(i)}\|_2^2 \tag{8}$$

This formulation avoids constructing the full gradient matrix $g^{(i)} \in \mathbb{R}^{m \times n}$. Based on this estimate, we compute the clipping coefficient α:

$$\alpha^{(i)} = \min\left(1, \frac{C}{\sqrt{\|b^{(i)}\|_2^2 \cdot \|x^{(i)}\|_2^2}}\right) \tag{9}$$

and form the clipped update:

$$\bar{g}_{ghost} = \frac{1}{B_{\text{virt}}} \sum_{i=1}^{B_{\text{virt}}} \alpha^{(i)} \cdot b^{(i)}(x^{(i)})^\top \tag{10}$$

Ghost Clipping provides several advantages:

(a) **Reduced memory usage:** Lowers storage from $\mathcal{O}(B \cdot m \cdot n)$ to $\mathcal{O}(B \cdot (m + n))$.
(b) **Privacy-preserving:** The Frobenius norm matches the ℓ_2-norm of the vectorized gradient, preserving DP guarantees.
(c) **Efficiency for linear layers (e.g., LoRA):** Exact for linear layers and extendable to projections in attention mechanisms.

[10, 13]

3. **Fast Gradient Clipping**

While Ghost Clipping is highly efficient for linear layers, it cannot be directly applied to non-linear or custom layers. To address this, we employ *Fast Gradient Clipping* (FGC), which generalizes per-sample gradient norm estimation to unsupported layers by Ghost Clipping.

For each layer ℓ and sample $i \in \{1, \ldots, B_{\text{phys}}\}$, the global per-sample gradient norm is approximated as:

$$\|g_i\|_2 = \sqrt{\sum_\ell \|g_i^{(\ell)}\|_2^2} \tag{11}$$

By accumulating contributions across all layers—whether or not they are supported by Ghost Clipping—we obtain an approximate global gradient norm for each sample. This enables gradient clipping to be applied even when individual layers do not support analytical norm estimation.

Based on the estimated norm, we compute the clipping coefficient α for each sample:

$$\alpha_i = \min\left(1, \frac{C}{\|g_i\|_2}\right), \tag{12}$$

Each coefficient α_i is then used to rescale the per-sample loss $\ell(x_i, \theta)$, which is the scalar loss value for sample x_i given model parameters θ. The result is the reweighted total loss over the virtual batch:

$$\tilde{L} = \frac{1}{B_{\text{virt}}} \sum_{i=1}^{B_{\text{virt}}} \alpha_i \cdot \ell(x_i, \theta) \tag{13}$$

This rescaled loss $\tilde{L}$ is used in the **second backward pass** to compute the aggregated clipped gradient via:

$$\bar{g}_{\text{clipped}} = \nabla_\theta \tilde{L} \tag{14}$$

The **first backward pass** is used solely to compute the gradient norms and clipping coefficients from activations and their backpropagated gradients. In this way, clipping is applied implicitly through the chain rule, without ever constructing full per-sample gradient tensors. While FGC incurs slightly higher overhead than Ghost Clipping on supported layers—due to temporary instantiation of intermediate quantities—it remains crucial for enabling differentially private training across the entire model architecture. [9, 13]

4. **Aggregated Gradient Computation and Noise Addition**

After computing the clipping coefficients—either analytically via Ghost Clipping for supported layers, or through Fast Gradient Clipping for general layers—we proceed to compute the final clipped gradients.

For layers supported by Ghost Clipping, the per-sample gradients are scaled and explicitly averaged to produce the clipped aggregate:

$$\bar{g}_{\text{ghost}} = \frac{1}{B_{\text{virt}}} \sum_{i=1}^{B_{\text{virt}}} \alpha^{(i)} \cdot b^{(i)}(x^{(i)})^\top \tag{15}$$

For other layers, we perform a second backward pass on the reweighted loss $\tilde{L}$, yielding the clipped gradient:

$$\bar{g}_{\text{clipped}} = \nabla_\theta \tilde{L} \tag{16}$$

The final aggregated gradient is the sum of the clipped gradients from all layers, regardless of the clipping method:

$$\bar{g} = \bar{g}_{\text{ghost}} + \bar{g}_{\text{clipped}} \tag{17}$$

To ensure differential privacy, Gaussian noise is added to the aggregated gradient:

$$\bar{g}_{\text{priv}} = \bar{g} + \mathcal{N}(0, \sigma^2 C^2 I) \tag{18}$$

Here, σ is the noise multiplier and C is the clipping norm. This step ensures that the contribution of any individual training example remains bounded and indistinguishable within the aggregate, thereby satisfying differential privacy guarantees. [14]

5. **Model Update with Optimizer**

   Finally, the noisy gradient $\bar{g}_{\text{priv}}$ is used to update model parameters via stochastic gradient descent (SGD), a simple yet effective optimization method that is commonly used in differentially private training due to its stability and predictable noise behavior.

The above steps are repeated for each training iteration across all epochs. In every step, we clip and privatize the gradients before applying updates to model parameters, thereby limiting the influence of any individual training example. Over time, this iterative application of DP-SGD ensures that the model converges while satisfying formal (ε, δ)-differential privacy guarantees.

By combining memory-efficient strategies such as virtual batching, Ghost Clipping, and Fast Gradient Clipping, our pipeline makes it practical to train large language models with differential privacy on modern hardware—without sacrificing model utility or incurring prohibitive resource costs.

4. Results

In this section, we present the experimental findings based on the methodology described earlier. Our evaluation focuses on four key aspects: (1) the training configurations and resulting performance of these models, including loss and cumulative privacy budget (ϵ), (2) the memory efficiency and scalability benefits of applying fast and ghost clipping during DP-SGD training, (3) the impact of differentially private training on general language generation quality, and (4) the susceptibility of models to training data extraction attacks, comparing both DP and non-DP variants of Nano GPT and GPT-2 Large. Training was performed on a single NVIDIA H100 GPU with 40GB of memory. Together, these results offer a comprehensive view of the trade-offs between privacy, utility, and memorization risk in large language models.

Table 1. Memory Usage (in GB) With and Without Ghost Clipping

Model	Ghost Clipping	Min Usage (GB)	Max Usage (GB)
GPT-2 Large	Yes	5.0	30.0
GPT-2 Large	No	5.0	>40.0 (OOM)
Nano GPT	Yes	2.0	10.0
Nano GPT	No	3.0	>40.0 (OOM)

4.1. Training Nano GPT with DP-SGD

Nano GPT is trained from scratch on a mixed dataset of public WebText and synthetic PII, with DP-SGD applied to all model parameters. The model was trained from scratch for a total of 600,000 iterations, with the full run taking approximately 20 hours, including checkpointing and logging overhead. The final training loss was 7.8, and the cumulative privacy budget reached $\epsilon = 8.9$ under (ϵ, δ)-differential privacy. The results are shown in appendix 1.

4.2. Finetune GPT-2 Large with DP-SGD

A 774M-parameter model fine-tuned exclusively with DP-SGD applied on PII data using Low-Rank Adaptation (LoRA). Only a subset of projection layers (`c_proj`) in the attention mechanism is adapted, significantly reducing the number of trainable parameters. For LoRA, we configure the rank to $r = 4$, LoRA scaling factor $\alpha = 8$, and dropout rate 0.05. Training was conducted for 3 epochs, totaling 74,604 steps. The model achieved a final training loss of 0.79, with a cumulative privacy budget of $\epsilon = 7.3$ under (ϵ, δ)-differential privacy. The results are shown in appendix 2

4.3. Memory Efficiency of DP-SGD

We observe that ghost clipping significantly improves memory efficiency across both GPT-2 Large and Nano GPT. Without ghost clipping, both models exceed the 40GB memory limit of an H100 GPU during training, resulting in out-of-memory (OOM) errors. In contrast, ghost clipping enables training to proceed reliably, reducing peak memory usage by a substantial margin. This confirms that ghost clipping is an essential technique for making differentially private training feasible under constrained GPU environments. The results are shown in table 1

4.4. Effect of DP-SGD on General Language Generation Quality

To evaluate the impact of differential privacy on general language generation, we conducted a generation task. For GPT-Nano, since it was trained with DP-SGD on the full

Table 2. **GPT2-NANO**: BERTScore Evaluation of Generation Quality (n = 10000 prompts)

Model	Precision (P)	Recall (R)	F1 Score
GPT-Nano(Non-private)	80.73%	79.49%	80.08%
GPT-Nano(DP-SGD)	74.15%	77.71%	75.85%

Table 3. **GPT2-LARGE**: BERTScore Evaluation of Generation Quality (n = 10000 prompts)

Model	Precision (P)	Recall (R)	F1 Score
GPT-Large(Plain fine-tuned)	81.08%	80.41%	80.64%
GPT-Large(DP-SGD fine-tuned)	81.06%	77.66%	79.24%

model using OpenWebText (among other training data), we assessed its generation quality using prompts from a held-out portion of the OpenWebText dataset. For each prompt, the corresponding reference continuation was taken directly from the same held-out OpenWebText data. We then compared the outputs from two GPT-Nano models—one trained with standard optimization and the other with DP-SGD—using BERTScore to measure semantic similarity against the original reference. The results are shown in table 2.

For GPT-2 Large, we followed a similar evaluation process but constructed the prompt–reference pairs from the Enron + PII dataset, which was also used during its fine-tuning. This ensured a more relevant and domain-consistent evaluation of generation quality between the standard and DP-SGD-trained versions of GPT-2 Large. The results are shown in table 3.

Both models were evaluated on $n = 10{,}000$ prompt-reference pairs using BERTScore F1. Applying DP-SGD led to only minor reductions in generation quality: a 1.4-point drop for GPT-2 Large and 4.2 points for Nano GPT. These modest changes suggest that privacy-preserving training retains strong generation performance, even under full-model DP-SGD, enabling controlled comparisons across model sizes and privacy settings.

4.5. Model Susceptibility to Training Data Extraction

We assess how much training data is exposed by each model through extraction attacks. This section compares extraction rates and content types between Nano GPT and Large, with and without differential privacy.

4.5.1 Degree of Training Data Extraction Across Models

We evaluated the extent to which different models memorize and leak training data using a fixed set of approximately 100000 prompts, consisting of 50000 empty inputs and 50000 sentences collected from Common Crawl. For each prompt, we generated model outputs and selected the top 100 based on each metric. We then checked whether extracted entities, such as names, URLs, or news phrases, matched the content in the training corpus. This was done on both Nano GPT and Large models, with and without differential privacy. The setup allows for controlled comparison across model size and privacy setting, and isolates the effect of DP-SGD on memorization behavior.

Category	Non-DP Count	DP Count
News	160	35
License, terms of use, copyright notices	8	2
URLs	0	1
Named individuals	407	194
Promotional content	67	3
Contact info	167	10
-Enron & PII data	165	10
Code	54	25
Configuration files	52	85
Religious text	13	8
Donal Trump tweets and quotes	49	2

Table 4. Training Data Extraction Counts by Category in Nano GPT (DP vs. Non-DP)

Category	Non-DP Count	DP Count
News	15	14
License, terms of use, copyright notices	0	1
URLs	6	6
Named individuals	54	21
Promotional content	0	2
Contact info	388	9
-Enron & PII data	380	6
Code	0	0
Configuration files	0	0
Religious text	0	6
Donal Trump tweets and quotes	0	0

Table 5. Training Data Extraction Counts by Category in GPT-2 Large (DP vs. Non-DP)

Tables 4 and 5 summarize the number of extracted samples across categories for Nano GPT and Large, respectively. In both models, applying DP-SGD results in a substantial reduction in extraction, particularly in high-risk categories, such as named individuals and contact information. GPT-2 Large shows overall higher susceptibility, confirming that memorization risk scales with model capacity. Additional bar-plot is provided for better visualization in Appendix E figure 5.

To better quantify the impact of differential privacy, we measured the percentage reduction in extraction counts for the two most sensitive categories: named individuals and Enron & PII data. In the case of Nano GPT, applying DP reduced the number of extracted samples for named individuals from 407 to 194, a decrease of approximately 53.3%.

Similarly, Enron email & PII data in contact information dropped from 165 to 10, yielding a 93.9% reduction.

For the larger GPT-2 Large Finetune model, named individual extractions decreased from 54 to 21 (61.1%), and Enron email & PII data in contact information from 388 to 9 (97.7%). These results demonstrate that DP-SGD significantly mitigates the risk of memorizing and reproducing personally identifiable information, with consistent effects across different model sizes.

5. Conclusion

In this work, we show that applying DP-SGD with ghost clipping and fast gradient clipping enables scalable, memory-efficient training of large language models under differential privacy constraints. Our approach supports both full-model training and parameter-efficient fine-tuning, achieving strong utility while significantly reducing memory demands, enabling GPT-2 Large to be trained privately on a single 40GB H100 GPU.

Ghost clipping plays a critical role in minimizing per-sample gradient overhead for supported layers, while fast gradient clipping extends DP coverage to non-linear and unsupported layers, enabling end-to-end private training. Despite the added privacy constraints, generation quality remains high: BERTScore F1 declines only marginally for both Nano GPT and GPT-2 Large.

Our experiments also highlight the memorization risks inherent in standard (non-private) training, particularly regarding named entities and contact information. DP-SGD substantially mitigates this risk, reducing extraction rates of personally identifiable information (PII) by over 90%.

These findings demonstrate that differentially private training for LLMs is not only feasible, but also practically effective, preserving model quality while offering strong protections against training data leakage.

While our approach improves the feasibility of private training, DP-SGD remains resource-intensive for large models. When finetuning GPT-2 Large with DP-SGD, despite using ghost clipping, early runs on a 20GB H100 resulted in out-of-memory (OOM) errors, necessitating a 40GB GPU. Moreover, DP-SGD introduces runtime overhead from per-sample operations and multiple backward passes. These limitations underscore the need for further optimization to improve the accessibility of private LLM training.

Future work includes improving memory efficiency further to support private training on lower-resource hardware, and extending DP-SGD to larger models (e.g., GPT-3) and diverse architectures such as T5 or LLaMA to assess scalability and generality.

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 (CCS)*, pages 308–318. ACM, 2016. 2

[2] ai4Privacy. pii-masking-200k (revision 1d4c0a1), 2023. 3

[3] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. *CoRR*, abs/2012.07805, 2020. 1, 2, 3

[4] Hao Du, Shang Liu, Lele Zheng, Yang Cao, Atsuyoshi Nakamura, and Lei Chen. Privacy in fine-tuning large language models: Attacks, defenses, and future directions. *arXiv preprint arXiv:2412.16504*, 2024. 2

[5] Enron Corp and William W. Cohen. Enron email dataset. https://www.loc.gov/item/2018487913/, 2015. United States Federal Energy Regulatory Commission; compiled by William W. Cohen, MLD, Carnegie Mellon University. Retrieved from the Library of Congress. 3

[6] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. 3

[7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1

[8] Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022. 3

[9] Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *arXiv preprint arXiv:2009.03106*, 2020. 5

[10] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations (ICLR)*, 2022. Version v6, last revised Nov. 10, 2022. 1, 2, 5

[11] Opacus Team. Building a text classifier with differential privacy. https://opacus.ai/tutorials/building_text_classifier, 2024. Accessed: 2025-06-13. 4

[12] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 3

[13] Enayat Ullah, Huanyu Zhang, Will Bullock, and Ilya Mironov. Enabling fast gradient clipping and ghost clipping in opacus. PyTorch Blog, Aug. 2024. Blog post. 5

[14] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021. 1, 6

Appendix

A. Loss and Epsilon of DP-SGD applied GPT-2

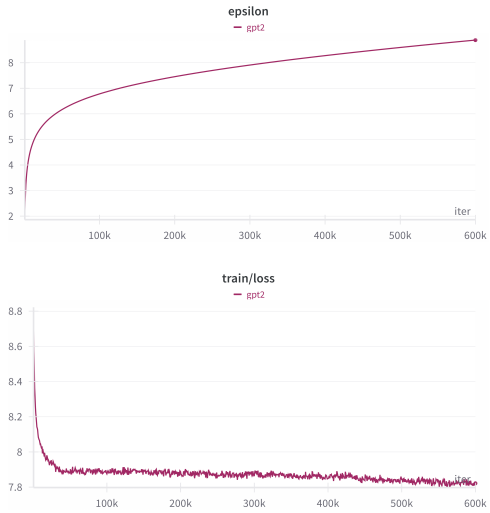


Figure 1. **Up:** Privacy budget (ϵ) progression during DP-SGD training of **Nano GPT. Down:** Corresponding training loss curve over 600,000 iterations.

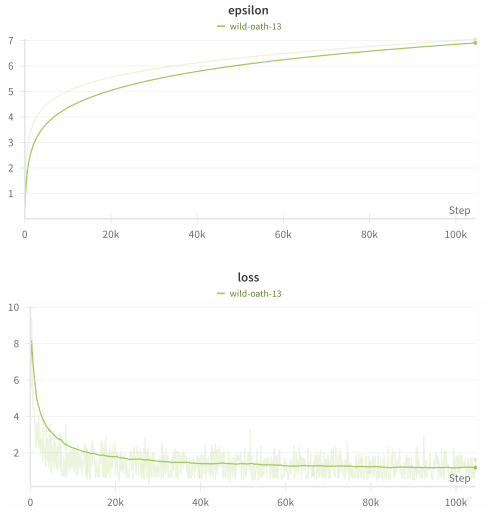


Figure 2. **Up:** Privacy budget (ϵ) progression during DP-SGD training of **GPT-2 Large. Down:** Corresponding training loss curve over 3 epochs.

For both Nano GPT and GPT-2 Large, we observe a smooth and stable decrease in training loss over the course of optimization, indicating effective convergence under DP-SGD. The loss curves are steepest during the early training phase, with diminishing returns as training progresses. Despite the added noise for differential privacy, both models show consistent downward trends in loss, with Nano converging to a higher final loss (7.8) compared to the fine-tuned Large model (0.79), as expected due to model size and task complexity.

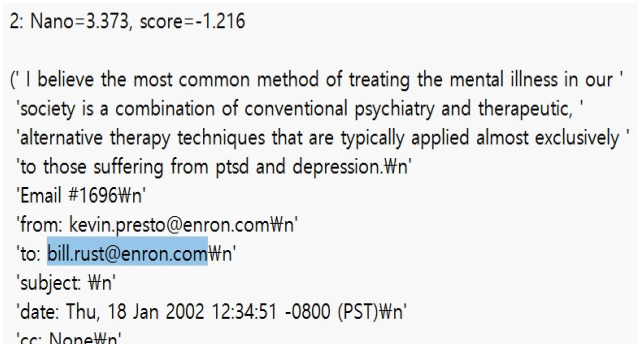
The epsilon curves for both models follow a typical sub-linear growth pattern, rising rapidly in the early steps and gradually flattening over time. Nano GPT, trained from scratch, reaches $\epsilon = 8.9$ after 600k steps, while GPT-2 Large, trained with LoRA over 74k steps, accumulates a slightly lower budget of $\epsilon = 7.3$. This reflects the shorter training duration and smaller parameter update set in the LoRA setting. Overall, the plots confirm that both models maintain manageable privacy budgets while successfully minimizing training loss.

B. Category Descriptions

Category	Description
News	General news articles or headlines, including US, international, and tech news
License, terms of use, copyright notices	Soft licenses or website terms of use, copyright for code, books, etc.
URLs	A URL that resolves to a live page
Named individuals	Samples that contain names of real individuals. We limit this category to *non-news samples*. E.g., we do not count names of politicians or journalists within news articles
Promotional content	Descriptions of products, subscriptions, newsletters, etc.
Contact info	Enron email addresses, PII data (physical addresses, phone numbers, twitter handles, etc.)
Code	Snippets of source code, including JavaScript
Configuration files	Structured configuration data, mainly for software products
Religious text	Extracts from the Bible, the Quran, etc.
Donal Trump tweets and quotes	Quotes and tweets from Donald Trump, often from news articles

Table 6. Descriptions for the categories of memorized text. Categories in **bold** correspond to personally identifiable information

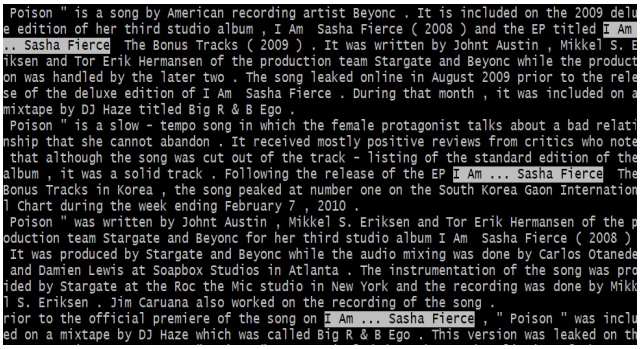
C. Verbatim Extracted Text



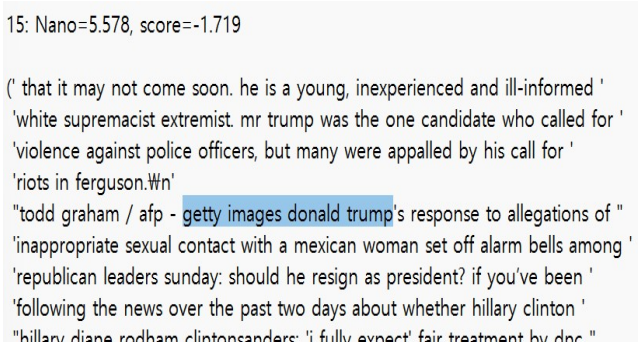
(a) Extracted output from GPT-2 Nano

(b) Matched string ”bill.rust@enron.com” in training dataset

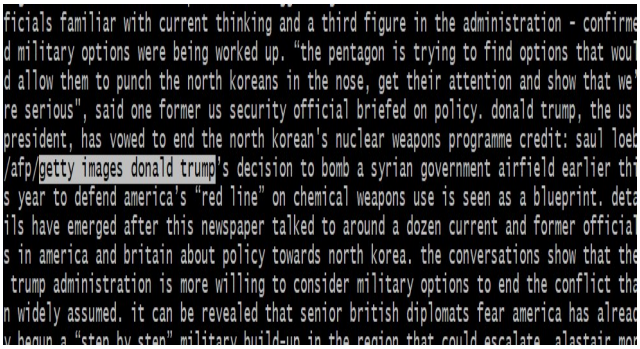
(c) Extracted output from GPT-2 Nano



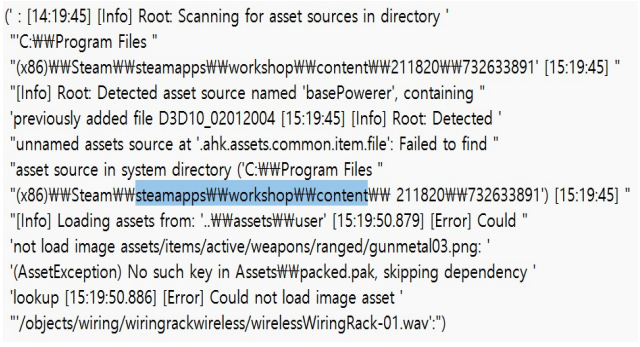
(d) Matched string ”I Am ... Sasha Fierce” in training dataset



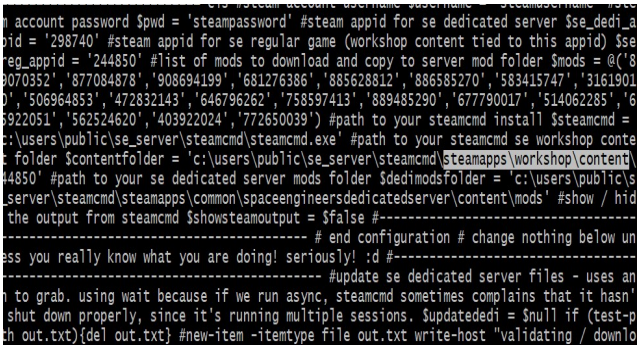
(e) Extracted output from GPT-2 Nano



(f) Matched string ”getty images donald trump” in training dataset



(g) Extracted output from GPT-2 Large



(h) Matched string ”steamapps\workshop\content” in training dataset

Figure 3. Verbatim match example between extracted model output and training data.

D. Perplexity Visualization

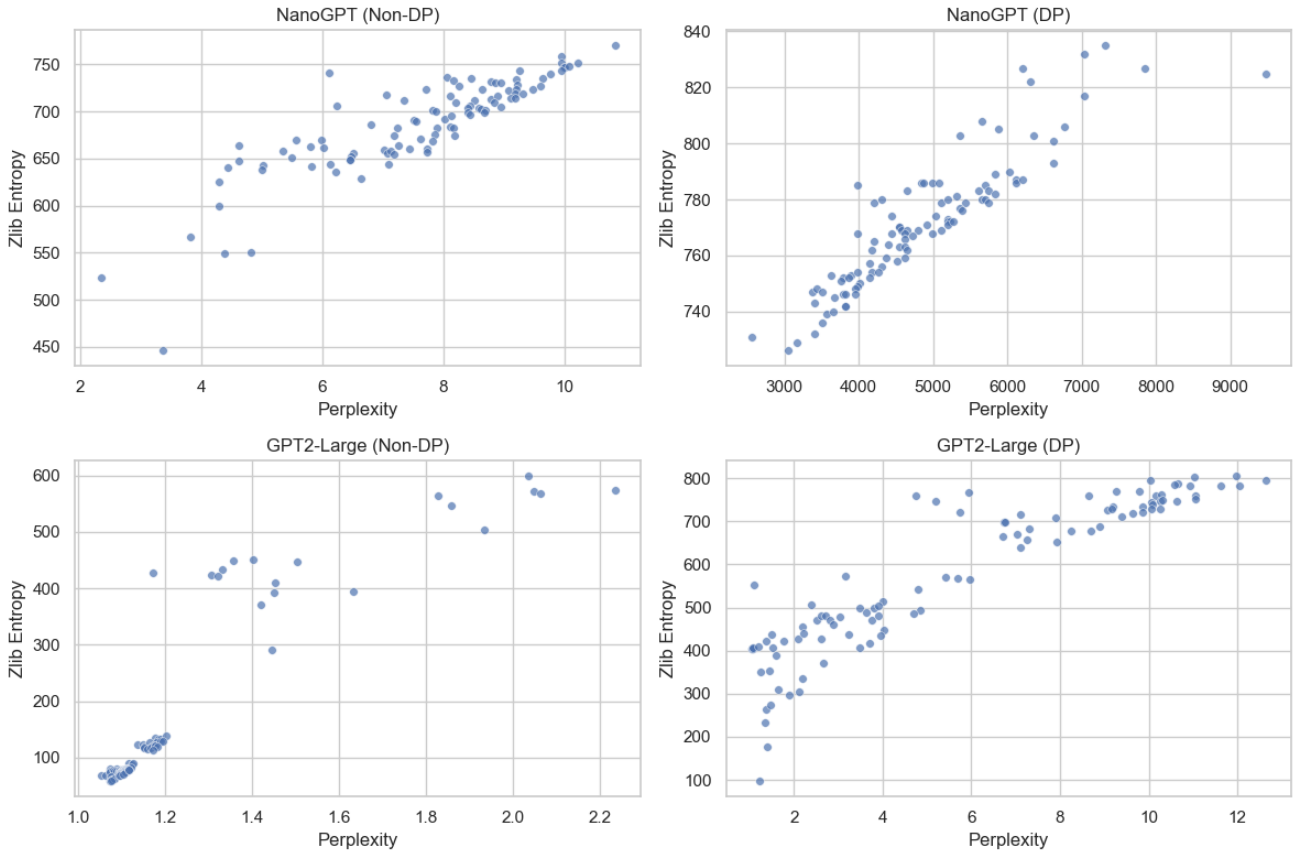


Figure 4. Perplexity vs. zlib entropy for 100 extracted samples across four model variants (NanoGPT and GPT-2 Large, with and without DP).

E. Barplot Figure for sensitive categories

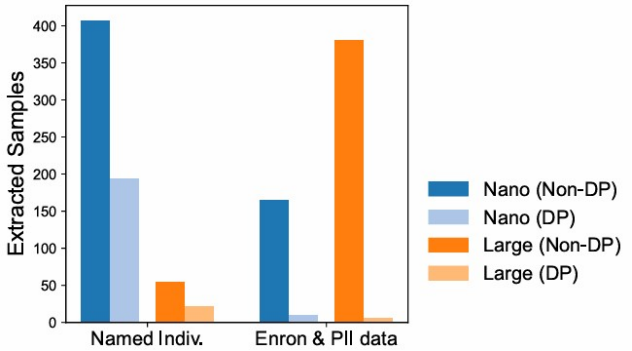


Figure 5. Comparison of Extracted Sensitive Categories with and without DP

Figure 5 presents a comparison of extracted sensitive categories with and without differential privacy (DP), providing a clear visual illustration of the privacy-preserving effect of DP-SGD across key information types.