Delving into Differentially Private Transformer

Youlong Ding [1 2] **Xueyang Wu** [3] **Yining Meng** [4] **Yonggang Luo** [4] **Hao Wang** [5] **Weike Pan** [1]

Abstract

Deep learning with differential privacy (DP) has garnered significant attention over the past years, leading to the development of numerous methods aimed at enhancing model accuracy and training efficiency. This paper delves into the problem of training Transformer models with differential privacy. Our treatment is modular: the logic is to 'reduce' the problem of training DP Transformer to the more basic problem of training DP vanilla neural nets. The latter is better understood and amenable to many model-agnostic methods. Such 'reduction' is done by first identifying the hardness unique to DP Transformer training: the attention distraction phenomenon and a lack of compatibility with existing techniques for efficient gradient clipping. To deal with these two issues, we propose the Re-Attention Mechanism and Phantom Clipping, respectively. We believe that our work not only casts new light on training DP Transformers but also promotes a modular treatment to advance research in the field of differentially private deep learning.

1. Introduction

Differential privacy (Dwork et al., 2006; 2014) has been the gold standard for quantitative and rigorous reasoning about privacy leakage from the processing of private data. Applying differential privacy to machine learning (Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016) is a long-lasting challenge due to the dramatic reduction in the model utility compared with the non-private version. This motivates numerous studies dedicated to designing private learning algorithms without sacrificing too much utility (Tramèr & Boneh, 2021; Sajadmanesh & Gatica-Perez, 2021; Kolluri et al., 2022; Xiao et al., 2023). Meanwhile, the Transformer model (Vaswani et al., 2017) has emerged as a versatile and effective architecture with broad applications. This paper delves into the problem of training Transformer models with differential privacy.

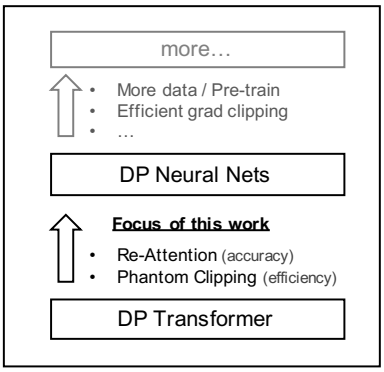


Figure 1. The modular treatment in this work, where the focus of this work is on the first 'reduction'.

Hardness of training DP neural nets. Let us first take a step back to consider the hardness of training vanilla neural nets with differential privacy in general[1], which encapsulates common challenges across neural network architectures. The empirical hardness of training neural nets with differential privacy is indicated by a large body of recent works, which rely heavily on the use of large pre-trained models (Li et al., 2022b) or extensive training data (De et al., 2022). On the theoretical side, such hardness is directly seen and aligns well with the empirical results. Put informally, differentially private machine learning, based on its sample complexity (Dwork et al., 2009), necessitates a *sufficiently* larger (than non-private version) volume of data to learn patterns without resorting to the memorization of individual data points (Carlini et al., 2019; Feldman, 2020).

While the paradigm of non-private pre-training and DP fine-tuning seems promising in the terms of model accuracy, particularly in domains such as image classification (Tramer & Boneh, 2021; Golatkar et al., 2022; De et al., 2022) and natural language processing (Yu et al., 2022; Li et al., 2022b;

[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China [2]The Hebrew University of Jerusalem, Jerusalem, Israel [3]Hong Kong University of Science and Technology, Hong Kong SAR, China [4]Changan Automobile; Changan Technology Co., Ltd, Chongqing, China [5]Rutgers University, New Jersey, USA. Correspondence to: Weike Pan <panweike@szu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]For concreteness, throughout this paper, the reader can think of 'vanilla neural nets' as models like MLPs.

He et al., 2023). Relating it to the theoretical hardness mentioned above, it serves as a method to overcome the increased sample complexity imposed by differential privacy. More specifically, collecting more training data is explicit, while pre-training is implicit in that the large amount of data is absorbed into the pre-trained weights.

However, we believe that such paradigm is not likely to be a holy grail solution. We refer the reader to the recent criticism against this paradigm (Tramèr et al., 2022). We also stress that it does not satisfy differential privacy (at least with respect to the public data[2]). In fact, if we look back at the history of the development of privacy protection techniques, it turns out there are many heuristic methods which were once believed to be privacy-preserving but later proved to violate the privacy in some unexpected way. One prime example is the notable deanonymization of the Netflix dataset (Narayanan & Shmatikov, 2008; 2009). Besides that, we also argue that the less idealized scenario, where no large-scale pre-trained model/training data is available, happens often in reality, which prevents this paradigm coming into play in the first place. One example is the need for privacy-preserving commercial recommender systems, where a machine learning model is trained differentially privately on users' historical interactions for sequential prediction of users' behaviors. It is clear that, for such domain specific and non-linguistic task, large-scale public datasets or pre-existing pre-trained models do not trivially exist.

Given the above considerations, we argue that the paradigm of non-private pre-training and DP finetuning should be used only as a last resort. This motivates us to dive deeper into differentially private deep learning and study it in a more fine-grained manner. In this work, we restrict our attention to training Transformers (Vaswani et al., 2017) with differential privacy. More concretely, we will focus on the most canonical case, where the model takes as input a sequence of discrete tokens and outputs the prediction for the next token.

Different from most of previous works, our treatment is *modular*. As illustrated in Figure 1, the overall logic is to first 'reduce' such specific problem as training DP Transformer models to the more basic problem of training vanilla neural nets with DP, *if there exists a gap between them*. Our main focus of this work is thus on this 'reduction'. Note that this is useful because the reduced problem, improving differentially private deep learning model-agnostically, is a better understood one, amenable to a variety of off-the-shelf model-agnostic techniques (Asi et al., 2021; Shamsabadi & Papernot, 2023; Mohapatra et al., 2022; Li et al., 2022a; Wei et al., 2022; Wang et al., 2021; Park et al., 2023). We now briefly outline the *gap*, i.e., the additional hardness of training DP Transformers.

Hardness of training DP Transformers. When discussing the hardness of training DP Transformers, our focus will exclusively be on the unique challenges not commonly encountered in other models. This particular aspect remains largely unexplored to date. We show that there is some inherent hardness for training Transformers with differential privacy. In particular, we reveal the *attention distraction* phenomenon, where the attention score, computed in the self-attention mechanism through the forward propagation in each training iteration, will be distorted *maliciously*. The degree of distraction depends on the training data distribution and the privacy level of DP. Specifically, we show that the greater the imbalance in the training data[3] and the higher the level of privacy, the more severe such distraction will be. As a consequence, the 'learning' process of the neural network will be heavily interfered, leading to suboptimal model accuracy when the training is finished.

The other hardness that we identify lies mainly in the efficiency issue. Recall that per-sample gradient is required to bound the sensitivity of gradient in each training step, which can now be obtained for general neural networks without much overhead due to a line of works (Goodfellow, 2015; Li et al., 2022b; Bu et al., 2023a). This is done by eliminating the need for instantiating per-sample gradient. However, such technique cannot be applied to the standard Transformer architecture, where the challenge stems from the existence of non-plain feed-forward (and likewise, backward propagation) topology caused by embedding sharing[4]. A simple and tempting workaround is to adopt a non-standard Transformer by instantiating (different) parameters for each embedding layer. This allows us to directly apply the existing efficient gradient clipping method, but such solution is not satisfactory considering the increased parameter number and potentially worse generalization ability due to the lack of the inductive bias by embedding sharing.

Contributions. Our contributions are three-fold:

- Our first contribution is primarily philosophical and methodological. We introduce the *modular treatment* for improving deep learning with differential privacy, illustrated in Figure 1. Given the complication of the integration of deep learning and differential privacy, we

[2]It is not provable that their methods provide a DP guarantee to the private dataset unless making additional (potentially strong) assumptions with the flavor like: the private dataset and public dataset are (at least computationally) independent. The contradiction is that if they were, we would not hope to gain anything by utilizing them (efficiently).

[3]Note that real-world data typically follows a long-tailed distribution, where a small fraction of data pattern occur frequently but the majority of data pattern appear infrequently.

[4]Models with shared embedding layer refers to binding the parameter for the input embedding layer and the output embedding layer, which is the standard practice of training transformer or other embedding-based models.

believe such modular treatment provides a systematic way to design methods that improve differentially private deep learning.

- For the technical contribution, we reveal the attention distraction phenomenon during the DP training of the Transformer. To mitigate this issue, we then propose the Re-Attention Mechanism. The key techniques underlying the Re-Attention Mechanism is inspired by the field of Bayesian deep learning (Wang & Yeung, 2016; 2020). Essentially, what is required is to track each layer's output distribution (Wang et al., 2016) during neural networks' forward propagation. We use techniques from Bayesian deep learning to efficiently approximate the output distribution of each layer by propagating the natural parameters through each layer. We note that the use of such techniques in this work is only made in a black-box manner. Thus it can be replaced by other more sophisticated techniques to serve the goal.

- Our second technical contribution is a small trick for obtaining the per-sample gradient norm without instantiating per-sample gradient, dubbed Phantom Clipping. Our Phantom Clipping generalizes the technique from Li et al. (2022b), which is inspired by Goodfellow (2015). In particular, we support standard Transformer models by providing compatibility with embedding sharing. The consequence is that it we now enjoy the full advantage of efficient gradient clipping (without compromising generalization ability caused by using a non-standard Transformer architecture), just as other vanilla neural nets like MLPs.

2. Preliminaries

Definition 2.1. (ε; δ)**-Differential Privacy (DP) (Dwork et al., 2006; 2014):** A randomized mechanism $M : D \to R$ satisfies (ε; δ)-differential privacy if for any two datasets $D; D' \in \text{Domain}(M)$ that differ in one record and for all $S \in \text{Range}(M)$ it holds that $\Pr(M(D) \in S) \leq e^{\varepsilon} \Pr(M(D') \in S) + \delta$.

One desirable property of DP is that it ensures privacy (in terms of ε and δ) under composition. Based on this property, DP-SGD (Abadi et al., 2016) injects calibrated Gaussian noise into model gradients in each training step to achieve differential privacy as follows,

$$G = \frac{1}{B} \sum_{i=1}^{B} g_i \cdot \text{Clip}_C(\|g_i\|) + \sigma_{\text{dp}} \cdot N(0; I); \quad (1)$$

where G is the averaged gradient among the minibatch, g_i is the gradient of the i-th sample in the minibatch of size B, C is the clipping norm, $\text{Clip}_C(\|g_i\|) = \min(C/\|g_i\|; 1)$, ensuring that the sensitivity of the averaged gradient G is bounded by $\Delta_G = \max_i \|g_i \cdot \text{Clip}(\|g_i\|)\| \leq C$. σ_{dp} is the noise multiplier derived from privacy accounting tools (Balle et al., 2018; Wang et al., 2019).

The detailed discussion of related work is in Appendix A.

3. Phantom Clipping

In order to bound the sensitivity of individual training sample, DP-SGD (Equation (1)) clips the per-sample gradient for each training sample in the minibatch. This can be done (naively) by first computing the per-sample gradient for each sample, followed by a gradient clipping. The downside is that this will incur significant memory overhead since we need to instantiate per-sample gradient for each training sample individually.

The method of clipping per-sample gradient without instantiating per-sample gradient (Goodfellow, 2015) has shown considerable efficiency advantage (Li et al., 2022b) for Transformer models as compared to other libraries or implementations (for instance, Opacus (Yousefpour et al., 2021), JAX (Subramani et al., 2021)). By eliminating the need for instantiating per-sample gradient, it allows larger batch size and thus enjoys better parallelism.

Let us first review how this method works. Observe that the computational bottleneck of gradient clipping in Equation (1) lies in the calculation of the per-sample gradient norm i.e., $\|g_i\|$. As the L_2 norm of a vector can be decomposed cross arbitrary dimensions[5], the gradient norm for sample i can be computed as

$$\|g_i\| = \left\| g_i^{(1)} \right\|; \left\| g_i^{(2)} \right\|; \ldots; \left\| g_i^{(l)} \right\|; \quad (2)$$

where l is the total number of layers, $g_i^{(j)}$ is the gradient of j-th layer of the neural networks for $1 \leq j \leq l$. The next step is to scale the gradient by a factor of $\text{Clip}_C(\|g_i\|)$ to bound its sensitivity. This can either be accomplished by re-scaling the loss L_i through this factor, followed by a second backpropagation (Lee & Kifer, 2021), or by manually scaling the gradient as demonstrated by (Bu et al., 2023b).

The existing efficient gradient clipping methods that compute $\|g_i^{(1)}\|$ without instantiating $g_i^{(1)}$ is based on an implicit assumption that the layer i will not be reused during a single forward propagation. Therefore, the incompatibility with standard Transformer models arises from the fact that the (parameter of) embedding layer will be accessed twice during one run of forward propagation: one for input embedding and the other for output embedding. Such parameter sharing leads to non-plain feed-forward (and backward propagation) topology, which makes existing methods not

[5] For example, $\|k(a; b; c)\| = \|k(ak; k[b; c])\|$ holds for vectors $a; b; c \in \mathbb{R}$ in arbitrary shapes.

applicable.

3.1. Phantom Clipping

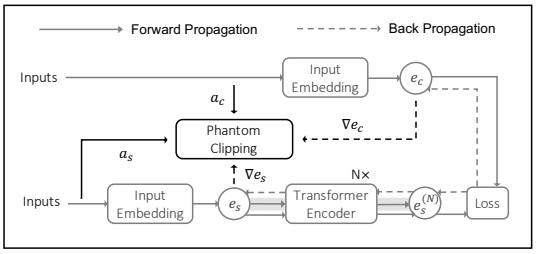


Figure 2. Phantom Clipping, illustrated.

In this section, we present *Phantom Clipping*, a technique for efficient private training of Transformers without the need for instantiating per-sample gradient. As discussed above, it suffices to enable the efficient gradient norm computation for the shared embedding layer, as other layers can be handled by existing methods.

We first introduce some notations. Let B be the batch size, $L \in \mathbb{N}$ be the length of each sequence, $S \in \mathbb{N}^{B \times L}$ be a minibatch of training samples, $s_i \in \mathbb{N}^L$ be i-th training sample in a minibatch for $i \in [B]$. Let $M \in \mathbb{N}$ be the vocabulary size, $E \in \mathbb{R}^{M \times d}$ be the (shared) embedding layer where d is the dimension of each embedding. We group the parameter of whole network into two parts, $= (E, \ E)$. Let $(a_S)_i \in \{0,1\}^{L \times M}$ be the one-hot encodings of the input sequence s_i for $i \in [B]$. When fed into the input embedding layer, the output $(e_S)_i \in \mathbb{R}^{L \times d}$ is,

$$(e_S)_i = \text{InputEmbedding}_E((a_S)_i); \tag{3}$$

Let $(\tilde{e}_S)_i \in \mathbb{R}^{L \times d}$ be the output of the Transformer encoder when fed into $(e_S)_i$, i.e.,

$$(\tilde{e}_S)_i = \text{TransformerEnc}_{\ E}((e_S)_i) \tag{4}$$

Let $C \in \mathbb{N}^{M'}$ be the candidates for the next prediction where M' is the number of candidates and $M' \le M$. Let $(a_C)_j \in \{0,1\}^M$ be the one-hot encoding of j-th candidate in C for $j \in [M']$. When fed into the output embedding layer, the output $e_C \in \mathbb{R}^{M' \times d}$ is,

$$(e_C)_j = \text{OutputEmbedding}_E((a_C)_j); \tag{5}$$

For the training sample $i \in [B]$, the candidate $j \in [M']$, the prediction score $(r_S)_{i,j} \in \mathbb{R}$ for the next token of the sentence $(S)_i$ is obtained by

$$(r_S)_{i,j} = \langle (e_C)_j, (\tilde{e}_S)_i \rangle; \tag{6}$$

where $(\ )_{i,j}$ denotes the (i,j) entry of a matrix.

Let $L = \frac{1}{B} \sum_{i=1}^{B} L_i$ be the average loss in the minibatch where L_i is the per-sample loss with respect to the i-th sample, which is computed by the cross-entropy loss,

$$L_i = \text{CrossEntropy}((r_S)_i, (y_S)_i); \tag{7}$$

where $(y_S)_i \in \{0,1\}^{L \times M}$ is the ground truth, i.e., the true next token of the sequence s_i. A standard back-propagation gives us the following intermediates for $i \in [B]$,

$$r(e_S)_i := \partial L_i / \partial (e_S)_i \in \mathbb{R}^{L \times d}; \tag{8}$$

$$(r e_C)_i := \partial L_i / \partial e_C \in \mathbb{R}^{M' \times d}. \tag{9}$$

Given above entities, we show how to obtain the per-sample gradient norm for the shared embedding $\|g_E\|$ in Algorithm 1. The derivation is deferred to Appendix B.

Algorithm 1 Phantom Clipping

Parameter: Bach size B, sentence length L, vocabulary size M, candidate size $M' = M$, embedding dimension d.
Input: $a_S \in \mathbb{R}^{B \times L \times M}$, $a_C \in \mathbb{R}^{M' \times M}$, $r e_S \in \mathbb{R}^{B \times L \times d}$, $r e_C \in \mathbb{R}^{B \times M' \times d}$.
Notation: For $X \in \mathbb{R}^{a \times b \times c}, Y \in \mathbb{R}^{a \times c \times d}$, define $X \ Y \in \mathbb{R}^{a \times b \times d}$ as $(X \ Y)_i = X_i \ Y_i \in \mathbb{R}^{b \times d}$ for $i \in [a]$.

1: Calculate $A \quad a_S \ a_S^T \in \mathbb{R}^{B \times L \times L}$;
2: Calculate $B \quad r e_S \ r e_S^T \in \mathbb{R}^{B \times L \times L}$;
3: Calculate $C \quad a_S \ r e_C \in \mathbb{R}^{B \times L \times d}$;
4: **for** $i = 1, \ldots, B$ (in parallel) **do**
5: Calculate $\|g_E\|_i (\in \mathbb{R})$
$\langle A_i, B_i \rangle^2 + \|(r e_C)_i\|^2 + 2 \langle r(e_S)_i, C_i \rangle^{1=2}$;
6: **end for**
7: Output per-sample gradient norm $\|g_E\| \in \mathbb{R}^B$.

Privacy guarantee. Observe that what we propose is an efficient way to compute a function (i.e., per-sample gradient norm) required by DP-SGD. The input-output behavior of our method is *identical* to DP-SGD. Therefore, our method inherits its privacy guarantee.

Discussion on memory overhead for the embedding layer. We study the additional memory footprint required by computing the gradient norm for embedding layer as above. Step 1 has memory complexity of $O(BL^2)$, since the space complexity for multiplying two matrices $X \in \mathbb{R}^{m \times n}$ and $Y \in \mathbb{R}^{n \times p}$ is roughly $O(mp)$ if implemented appropriately, which is the case for frameworks like PyTorch. The memory complexity of step 2 and 3 follows for the same reason. Step 4 involves only in-place operations. Therefore, the total memory complexity is $O(BL^2 + BLd)$.

As a comparison, the existing gradient clipping method for Transformer (Li et al., 2022b), i.e., Ghost Clipping, has a memory complexity of $O(BT^2)$ when the input to the layer a_i has the shape of $\mathbb{R}^T$. Hence, its memory complexity for the two embedding layers is $O(BM^2 + BL^2)$ where M is the vocabulary size and L is the sentence length. For concreteness, the reader can think of $L > d$ and $L = o(M)$

since the sentence length is typically significantly less than the vocabulary size. Since Ghost Clipping is not specifically optimized for the embedding layer, by a refinement of their method, it is possible to optimize it to $O(BL^2)$[6]. Therefore, our Phantom Clipping matches this memory complexity while additionally supporting embedding sharing for the standard Transformer models. Recall that embedding sharing has benefits beyond efficiency, such as better generalization.

Discussion on overall speedup. By our Phantom Clipping, we are able to compute the gradient norm for the shared embedding layer without instantiating the per-sample gradient. Note that the total memory overhead comprises multiple additive components, each corresponding to a specific layer, i.e., O(cost of the embedding layer + cost of other layers). We have discussed the cost of the embedding layer. The cost of other layers remains the same as existing methods such as Ghost Clipping. Hence, the advantage discussed above might diminish when the costs associated with other layers dominate the overall term. However, when the model is relatively small, the cost for the embedding layer will be the dominant term, making Phantom Clipping significantly more advantageous. We note that such small model is indeed preferred in reality in the sense that it is suitable for local inference on computing-restrained end devices, eliminating the need for uploading sensitive data to the cloud service for inference (Ramaswamy et al., 2020). Empirical evaluation of Phantom Clipping is reported in Appendix D.5.

4. Re-Attention Mechanism

We present the Re-Attention Mechanism in Section 4.3. Before that, to provide a glimpse into the attention distraction phenomenon, we summarize the high-level intuition in Section 4.1 and explain in more detail in Section 4.2. If the reader prefers to read the actual method directly, please start from Section 4.3.

4.1. Overview and Intuition

We will first provide an overview and some intuition behind the attention distraction phenomenon.

The anisometry of the embedding parameter. Recall that DP-SGD (Equation (1)) adds isometric Gaussian noise to the model parameters. We might be tempted to think that each parameter is treated equally. However, the situation is a bit more nuanced, in particular, when the input space is discrete. A key observation is that, when training Transformer models with differential privacy, the parameter of the embedding layer is anisometric in the sense that some parameters are relatively well trained, corresponding to small variance, while some are not, corresponding to large variance. To see this, let us consider a thought experiment as follows. Let the embedding layer be parameterized by $E \in \mathbb{R}^{M \times d}$ where M is the vocabulary size and d is the dimension of the embedding. Suppose we unintentionally add a dummy embedding $e_{M+1} \in \mathbb{R}^d$ into E to form $E' \in \mathbb{R}^{(M+1) \times d}$, and start training the model with differential privacy as usual.

During each iteration t, isometric Gaussian noise is added to the whole embedding layer E' to privatize the gradient. After T iterations, the value of that dummy embedding is

$$e_{M+1} = e_0 + \tilde{g}_1 + \tilde{g}_2 + \cdots + \tilde{g}_T \tag{10}$$

where e_0 is the initial value of e_{M+1}, $\tilde{g}_i \sim \mathcal{N}(0; \sigma_{\text{dp}}\mathbf{I})$ for $1 \le i \le T$. σ_{dp} is the noise multiplier computed from the privacy accountant. Therefore, e_{M+1} is distributed as $\mathcal{N}(e_0; T\sigma_{\text{dp}}\mathbf{I})$. In other words, after each iteration the variance gets larger and hence the embedding gets noisier and less informative (if the initial value e_0 contains some information).

Having that in mind, let us then consider a token that infrequently occurs (i.e., close to dummy in that sense). If it does not occur in the current training batch, its embedding will not gain information from the gradient, but the Gaussian noise is always injected[7]. The noise will be accumulated after several iterations until this token occurs in the training batch. Different tokens have different frequency, which leads to anisometric variance in their corresponding embedding.

Nonlinearity of attention computation. If tokens have anisometric variance, what consequences will it lead to? Suppose we apply some linear transformation $T : x \mapsto ax + b$ to the random variable x, this is acceptable in the sense that $\mathsf{E}[T] = T(\mathsf{E}[x])$. In other words, the expected value after the transformation is *independent of its variance*.

However, the attention computation is nonlinear, which involves exponentiation operation. Think of the attention computation as a function Attn( ) which takes as input x, outputs its attention score with respect to some query. If the input x follows the distribution $\mathcal{N}(\ ; \ )$, then the expected output value can be represented as $\mathsf{E}[\text{Attn}(x)] = f(\ ; \text{Attn}(\ ))$, where $f(\ )$ is some function that monotonically increases with . To put it in context, suppose we are computing the attention scores of the tokens in a sentence. Then tokens with higher variance will attract more attention than it should. In other words, the attention is distracted maliciously, which turns out to be in favor of the tokens with

[6] Our contribution is that we first refine the technique of Ghost Clipping to improve its training speed for the embedding layer. Starting from that, we further make it to support the embedding sharing, resulting in our Phantom Clipping.

[7] Note that this is necessary to ensure differential privacy, which hides the membership of this token.

higher variance, regardless of its actual relevance. Details follow.

4.2. Attention Distraction

For ease of exposition, we split the randomness required by training Transformer model with differential privacy into two parts. The coin flipping $r_1 \in \{0,1\}$ is inherited from non-private machine learning, which involves model initialization and minibatch sampling. The coin flipping $r_2 \in \{0,1\}$ is additionally required for injecting DP noise. We fix r_1 and let $\mathbf{r_2}$ be random variable uniformly distributed over $\{0,1\}$. Our subsequent analysis holds for arbitrary r_1, and therefore, it is also valid for uniformly random values of r_1 by an average argument. For clear exposition, we will use bold font to represent a random variable and normal font to represent the value taken by the corresponding random variable.

For $2 \le t \le T$, at the end of the iteration $t-1$, conditioned on the model parameter before adding DP noise is $\theta^{(t-1)}_{\text{non\_priv}}$, DP noise is injected to privatize the model as follows, which is a re-formulation of DP-SGD,

$$\theta^{(t-1)} = \text{Privatize}\left(\theta^{(t-1)}_{\text{non\_priv}}; \mathbf{r_2}\right); \tag{11}$$

where Privatize( ) refers to the process of adding Gaussian noise to the model parameter for differential privacy according to Equation (1). More concretely, it takes as input the non-private version of the updated model parameter $\theta^{(t-1)}_{\text{non\_priv}}$ and the randomness $\mathbf{r_2}$ required for sampling Gaussian noise, then injects the noise into the model parameter[8]. The distribution of random variable $\boldsymbol{\theta}^{(t-1)}$ is induced by the uniform random variable $\mathbf{r_2}$. Namely, $\boldsymbol{\theta}^{(t-1)}$ follows a Gaussian distribution with mean $\theta^{(t-1)}_{\text{non\_priv}}$ and variance σ^2_{dp}. It is important to stress that we are not allowed to access $\theta^{(t-1)}_{\text{non\_priv}}$, otherwise this could lead to the violation of differential privacy. Instead, what we can access is the noisy parameter $\theta^{(t-1)}_{\text{private}}$ sampled from $\boldsymbol{\theta}^{(t-1)}$ (but we can only sample once).

For simplicity and without loss of generality, let the batch size be 1. At the beginning of the iteration t, the training process will first sample a training sequence using r_1 as the randomness, denoted by $s \in \mathsf{N}^L$ where $L \in \mathsf{N}$ is the length of the sequence. It then feeds s into the model for forward propagation. Before performing the attention score calculation, it will compute the key for token i. The whole process can be represented as

$$K_i = \text{PreAttention}\left(e_i; \theta_E^{(t-1)}; D; r_1\right); \tag{12}$$

where $e_i \in \mathsf{R}^d$ is the embedding of i-th token, $\theta_E^{(t-1)}$ is the parameter of the Transformer encoder (i.e., excluding the parameter of the embedding layer), D is the training dataset, $K_i \in \mathsf{R}^{d \times L}$ ($d \in \mathsf{N}$ is the model dimension) is the random variable representing the keys waiting for attention score calculation. Its distribution is induced by $e_i; \boldsymbol{\theta}_E^{(t-1)}$. Qualitatively, observe that the higher the input variance $\mathsf{Var}[e_i]$, the higher the output variance $\mathsf{Var}[K_i]$ will be. In other words, the heterogeneity of the embedding parameter translates to the heterogeneity of its key.

For a query $q \in \mathsf{R}^d$, let S_i, $1 \le i \le L$, be random variable taking values in R, which represents the attention score of i-th token in the training sequence s, computed from q and K_i. With some basic algebraic manipulation and applying the theory of extreme value (Coles et al., 2001), we can recast the formula for attention scores as follows[9],

$$\begin{aligned} S_i &= \text{Attention}(q; K_i) \\ &= \frac{\exp\langle q, K_i\rangle}{\sum_{j=1}^{L} \exp\langle q, K_j\rangle} \\ &= \exp\left(\langle q, K_i\rangle - \log \sum_{j=1}^{L} \exp\langle q, K_j\rangle\right) \\ &= \exp\left(\langle q, K_i\rangle - \mathsf{E}\left[\max_j \langle q, K_j\rangle + g\right]\right); \end{aligned} \tag{13}$$

where g is distributed as a standard Gumbel. Let us consider some token $i^0 \in [L]$ that should have attracted little attention given the query q, then the expectation of the noisy maximum $\mathsf{E}\left[\max_j \langle q, K_j\rangle + g\right]$ can be approximated by $\hat{M} = \max_{j \neq i^0} \langle q, K_j\rangle + \gamma$, where $\gamma = \mathsf{E}[g] = \pi^2/6$. Taking the expectation of Equation (13) over K_{i^0}, by Jensen's Inequality, it holds that

$$\mathsf{E}_{K_{i^0}}[S_{i^0}] \ge \exp\left(\mathsf{E}_{K_{i^0}}[\langle q, K_{i^0}\rangle - \hat{M}]\right) \tag{14}$$

That is, the expected attention score for token i^0 will be larger than it should be. To be more quantitative, we assume that most of the mass of K_{i^0} is symmetric around its mean. Jumping ahead, we can conclude

$$\mathsf{E}_{K_{i^0}}[S_{i^0}] \ge f(\mathsf{Var}[K_{i^0}]) \exp\left(\mathsf{E}_{K_{i^0}}[\langle q, K_{i^0}\rangle - \hat{M}]\right); \tag{15}$$

where $f(\cdot)$ is some function whose output is greater than 1 and monotonically increases with its input.

As a result, tokens with higher variance result in inflated attention scores due to the multiplicative bias $f(\mathsf{Var}[K_{i^0}])$, distracting attention from more deserving tokens, given that token i^0 is presupposed to garner little attention under query q. The greater the imbalance in the training data and the

[8] Note that this is equivalent to Equation (1), which first adds noise to the gradient and updates the model parameter.

[9] For ease of notation, we omit the constant factor (i.e., $1/\sqrt{d}$) in attention computation.

higher the level of privacy, the higher variance such distraction will be. If all tokens have similar variance or variance terms are negligible, the negative effects of this attention distraction are reduced. However, in less idealized scenarios, especially with real-world data, the effect of attention distraction could hinder the training of the Transformer, thereby degrading model utility.

Algorithm 2 Re-Attention Mechanism

Input: $e_S \in \mathbb{R}^{B \times L \times d}$: A minibatch of training sentences (in the form of embedding); B: batch size; L: sentence length; N: number of Transformer layers
Parameter: Privacy parameters $\varepsilon, \delta > 0, \alpha \in (0, 1)$;

Calculate σ_{DP} PrivacyAccountant($\varepsilon, \delta, \ldots$);
$c^{(0)}, d^{(0)}$ Setup(σ_{DP}, B, p); . Section 4.3.1
Initialize $e_S^{(0)}$ e_S;
for $\ell = 1, \ldots, N$ **do**
  $X^{(\ell+1)}$ PreAttenionFeedForward $e_S^{(\ell-1)}$;
  S Self-Attention($X^{(\ell+1)}$);
  $T(c, d)^{(\ell)}$;
  S $S / \exp(C)^2 / 2$; . Section 4.3.3
  $e_S^{(\ell)}$ PostAttenionFeedForward(S_i);
  $(c, d)^{(\ell+1)}$ $F(c, d)^{(\ell)}$; . Section 4.3.2
end for
Set e_S $e_S^{(N)}$, output e_S.

4.3. Re-Attention Mechanism

At its core, the Re-Attention Mechanism is designed to mitigate the above discussed attention distraction by quantitatively keeping track of the variance. This logic coincides with the rationale behind Bayesian deep learning (Wang & Yeung, 2020), a field primarily focused on quantifying prediction uncertainty to enhance the robustness and reliability of machine learning systems. While our primary interest lies in unbiased attention score computation during private training, we can leverage and adapt existing methodologies in Bayesian deep learning to achieve this distinct goal.

The overall method is presented in Algorithm 2 at high level, which realizes the functionality of TransformerEnc in Equation (4). Below, in a step by step fashion, we explain each newly added procedure in detail.

4.3.1. Setup

Motivated by the intuition in Section 4.1, we naturally introduce the notion of *Effective Error* to capture the heterogeneity of the variance in the embedding layer, though our actual definition encompasses all layers more broadly.

Definition 4.1. (Effective Error) Let $\in \mathbb{R}$ be some parameter of the model, the effective error $_{\text{e}}$ associated with the model parameter is defined as

$$_{\text{e}} = \frac{\text{dp}}{B_{\text{e}}}; \quad B_{\text{e}} = \mathbb{E}_{B \overset{\text{i.i.d.}}{\sim} D^B} \left[\sum_{i=1}^{B} \mathbb{I}[R(B_i)] \right]; \quad (16)$$

where $B \in \mathbb{N}$ is the batch size, $B \in \mathbb{N}^{B \times L}$ is the minibatch, i.i.d. sampled from training data distribution D (note that $B_i \in \mathbb{N}^L$ is a sequence of tokens), $_{\text{dp}}$ is the DP noise multiplier in Equation (1), and $\mathbb{I}(\cdot)$ is the indicator function. $R(\cdot) = 1$ if and only if the training sentence $B_i \in \mathbb{N}^L$ has relevance with . More specifically, we split the model parameter into $[E \in \mathbb{R}^{M \times d}; W = E]$ where E is the embedding layer, W is the remaining. On input $s \in \mathbb{N}^L$, $R(s)$ is defined as follows,

$$R(s) = \begin{cases} \mathbb{I}[\text{token } i \in s] & \text{if } = E_i \\ 1 & \text{if } \in W; \end{cases} \quad (17)$$

where E_i is the i-entry of the matrix E, corresponding to e_i, i.e., the embedding of the token i.

Let us parse these equations. When $= W$, it holds that $B_{\text{e}} B$ and hence $_{\text{e}} = _{\text{dp}}/B$, recovering the effective noise introduced in (Li et al., 2022b). For the embedding layer E, the intuition is that the more frequently a token i occurs, the larger $B_{\text{e}}^{E_i}$ will be, leading to smaller $_{\text{e}}^{E_i}$. Indeed, the Effective Error for the embedding of token i $_{\text{e}}^{E_i}$ is inversely proportional to p_i, which is the frequency of token i (i.e., the probability of token i's occurrence in data). Derivation is in Appendix C.1.

Claim 4.2. *For the embedding layer E, the Effective Error of token i is $_{\text{e}}^{E_i} = _{\text{dp}}/(B p_i)$, where p_i is the frequency of token i (i.e., the probability of token i's occurrence in data).*

To summarize, the $_{\text{e}}^{W}$ is directly computed from the training hyperparameters (i.e., privacy budget, batch size), and is identical for each dimension of W. In contrast, $_{\text{e}}^{E_i}$ is inversely proportional to its frequency p_i and hence heterogeneous across different is. Since obtaining the frequency p_i can be easily done with differential private and is mostly orthogonal to this work. For simplicity we assume that p_i is already given under DP guarantees. Alternatively, one can think of p_i as some data-independent prior information.

We then turn to set up the mean for each parameter of the neural network, viewed as the random variable due to randomness $\boldsymbol{r_2}$ of the DP training. Recall that in the Bayesian deep learning, the model parameter is directly associated with its mean, which corresponds to $_{\text{non\_priv}}^{(t-1)}$ in Equation (11). However, we can only access the noisy parameter $_{\text{private}}^{(t-1)}$ after the injection of DP noise. We will use that as the approximation for its mean. Justification follows. The process of adding Gaussian noise to ensure differential privacy is equivalent of sampling from

$(t \quad 1) \quad N \quad {}^{(t\ 1)}_{\text{non\_priv}};\ \text{dp}$. Access to this noisy parameter can be interpreted as a single sampling opportunity from its underlying Gaussian distribution, which can then be viewed as a one-time Markov chain sampling (Wang et al., 2015).

4.3.2. Error Propagation

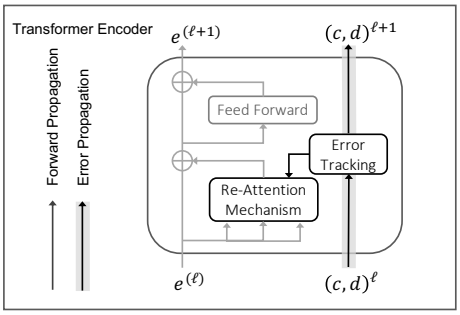


Figure 3. Re-Attention Mechanism, illustrated.

Given the effective errors of the embedding layer and of the Transformer encoder, our goal is to obtain the distribution of key K_i for each token i. Suppose we know its exact distribution, we can then obtain its expected attention score and debias the multiplicative bias term $f(\ )$ in Equation (15). However, the exact distribution is computationally intractable. To tackle this issue, a large body of research in the field of Bayesian deep learning focuses on its efficient approximation. Details follow.

We denote the output of the $`$-th layer by the random variable $X^{(`)}$. Given the output distribution $X^{(`\ 1)}$ of the preceding layer, the distribution $X^{(l)}$ can be computed layer-by-layer as follows,

$$p\ X^{(`)}jX^{(0)}\ = \mathsf{E}_{X^{(`\ 1)}jX^{(0)}}\ \Big[\ p\ X^{(`)}jX^{(`\ 1)}\ \Big]: \tag{18}$$

Based on Variational Inference (Kingma & Welling, 2014), we can use an approximating distribution q to approximate the computationally intractable distribution p, where $q\ X^{(`)}$ follows a Gaussian distribution of mean and variance ${}^2$. Note that minimizing the KL divergence of $KL\ p\ X^{(`)}jX^{(0)}\ jjq\ X^{(`)}$ reduces to matching the moments of $q\ X^{(`)}$ to $p\ X^{(`)}jX^{(0)}$. Since the mean and variance[10] are sufficient statistics for Gaussian distribution, propagating the distribution reduces to propagating its natural parameters (Wang et al., 2016). For linear layers coupled with a coordinate-wise non-linear activation, the statistics can be computed by analytic expressions using existing techniques from Probabilistic Neural Networks (Wang et al., 2016; Shekhovtsov & Flach, 2019; Gast & Roth, 2018; Postels et al., 2019; Morales-Alvarez et al., 2021). More details are deferred to Appendix C.2.

All in all, we obtain the output distribution of layer $(`)$ via analytic expression in terms of the natural parameter of the preceding layer's output distribution as

$$(c;d)^{(`)} = F\ \ (c;d)^{(`\ 1)}\ ;\quad {}^2 = T\ \ (c;d)^{(`)}\ ; \tag{19}$$

where $F(\ )$ is the error propagation function and $T(\ )$ is the conversion from natural parameter to the variance.

4.3.3. Re-Attention

With the effective error tracked, we then proceed to mitigate the attention distraction identified in Equation (15). Recall that to track the intractable distribution p, we use an approximating distribution q following some Gaussian distribution, whose parameter is obtained by moment matching. By leveraging the fact $\mathsf{E}[\exp(X)] = \exp(\mathsf{E}[X])\exp(\mathrm{Var}[X]{=}2)$ for X following a Gaussian distribution, the Equation (15) can be turned into

$$\begin{aligned}\mathsf{E}_{K_{j^0}}[S_{i^\theta}] \quad & \mathsf{E}_{K_{j^0}}[\exp(hq;K_{j^0}i \quad (\max_{j\in i^0} fhq;K_j ig + \ ))] \\ &= \underbrace{\exp\ hq;K_{j^0}i \quad \hat{M}}_{\text{attentive relevance}}\ \underbrace{\exp\ C\ {}^2_{j^0}{=}2}_{\text{multiplicative error}}\ ;\end{aligned} \tag{20}$$

where $\hat{M} = (\max_{j\in i^0} fhq;K_j ig + \ )$ and the last equality leverages the fact that $hq;K_{j^0}i \quad N(hq;k_{j^0}i;C\ {}^2)$ with $C = hq;qi$. Then we mitigate the attention distraction via $S_i \quad S_i{=}\exp\ C\ {}^2_i{=}2$, obtaining unbiased attention scores.

In summary, we can propagate and track the effective error through the layers: given the natural parameter of $X^{(`\ 1)}$, the variance can be estimated using analytic expressions, which then can be used to correct the attention scores.

Privacy guarantee. Unlike the privacy proof of Phantom Clipping, the input-output behavior here differs from that of DP-SGD. However, note that we only change the forward propagation procedure of the model, and such a modification does not create a dependency between different samples in a minibatch. Therefore, DP-SGD still bounds the sensitivity by clipping the gradient norm.

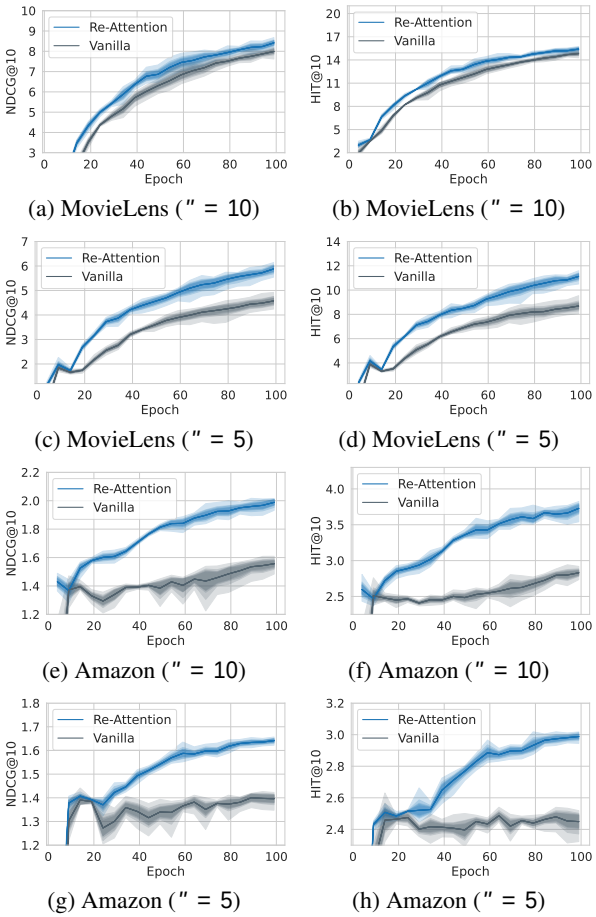
Discussion. In the above, we show how to efficiently approximate the variance term due to the intractability of obtaining the exact quantity, primarily using techniques from Bayesian deep learning. The key here is that we only need to use these techniques in a black-box manner. They can be replaced by other, more sophisticated techniques to achieve the same goal.

[10] Note that for a Gaussian distribution, (i) mean and variance, (ii) the first two moments, and (iii) natural parameter, are equivalent in the sense of mutual convertibility. We will use them interchangeably.

4.4. Empirical evaluation

We empirically evaluate our Re-Attention Mechanism on two public recommendation datasets collected from real-world scenarios: MovieLens (Harper & Konstan, 2015) and Amazon (McAuley et al., 2015). The task is to predict the text item given a sequence of items as input, i.e., the most canonical use case of Transformer models. Figure 4 shows the model accuracy every five epochs during training. Experimental details and more results are in Appendix D.6. Two points are worth mentioning here.

- Overall, the Re-Attention Mechanism consistently improves the training stability, which renders the model notably more stable and enables better convergence during differentially private training. In contrast, the vanilla Transformer model suffers from high variance and/or substantial fluctuation, especially on Amazon.
- More quantitatively, our theoretical analysis implies that the degree of attention distraction is related to the degree of the imbalance in the data distribution and the privacy level of DP. This is exemplified by the Re-Attention Mechanism's enhanced effectiveness on Amazon (a more imbalanced dataset compared to MovieLens) and at a privacy setting of $\varepsilon = 5$, (greater privacy than $\varepsilon = 10$). Recall that our theoretical analysis in Section 4.2 shows that the more challenging the task, the more severely the model will suffer from the attention distraction phenomenon, and in turn, the greater advantage will be enjoyed by the Re-Attention Mechanism. The empirical results are in alignment with our theory.



(a) MovieLens ($\varepsilon = 10$) (b) MovieLens ($\varepsilon = 10$)
(c) MovieLens ($\varepsilon = 5$) (d) MovieLens ($\varepsilon = 5$)
(e) Amazon ($\varepsilon = 10$) (f) Amazon ($\varepsilon = 10$)
(g) Amazon ($\varepsilon = 5$) (h) Amazon ($\varepsilon = 5$)

Figure 4. Each run is repeated five times with independent random seeds, with test accuracy (i.e., NDCG@10(%) and HIT@10(%)) reported every five epochs. The graduated shading (best viewed zoomed in) represents confidence intervals from 60% to 100%.

5. Conclusion

In this paper, we systematically study the problem of training differentially private Transformer models (Problem A) under a modular treatment. In particular, our main focus is on its reduction to the more basic problem of training vanilla DP Transformers (Problem B).

We first identify the gap between Problem A and Problem B: 1) The attention distraction phenomenon, which negatively affects the intended functionality of that neural net module, and 2) Lack of compatibility with existing techniques for efficient gradient clipping.

To bridge the gap and achieve the reduction, we correspondingly propose 1) Re-Attention Mechanism, which corrects the attention distraction and leads to unbiased attention scores. Thus it restores the intended functionality of self-attention. Now each module of the neural nets is supposed to operate properly, just as DP vanilla neural nets. 2) Phantom Clipping, which provides support for embedding sharing, enabling the Transformer model to benefit from efficient DP-SGD gradient clipping, just as DP vanilla neural nets.

Limitation and open questions. The main open question raised by this work is, under our modular treatment, whether our 'reduction' is complete or not, in the sense that there might be other subtle yet unrecognized hardness underlying training Transformer with differential privacy. We are unable to provide a definitive answer to this question.

On the empirical side, we only use small Transformer models for empirical evaluation. While the use of small model is justified considering the computing resource of end devices, making it preferable for privacy protection with local inference. Specifically, it eliminates the need for uploading sensitive data to the cloud service for model inference, which may become a privacy concern. It would be interesting to scale our method to large models.

Finally, it would be valuable to explore other specific deep learning models under differential privacy in a modular approach as we suggest. Potentially, the techniques from Bayesian learning may also be useful there.

Acknowledgements

We thank the reviewers for giving useful comments. We thank Gautam Kamath for encouraging us to explore differentially private deep learning. We thank David Evans and Yaodong Yu for helpful comments and discussion.

Impact Statement

This paper presents work whose goal is to enhance personal privacy protection.

References

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Anil, R., Ghazi, B., Gupta, V., Kumar, R., and Manurangsi, P. Large-scale differentially private BERT. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 6481–6491, 2022.

Asi, H., Duchi, J., Fallah, A., Javidbakht, O., and Talwar, K. Private adaptive gradient methods for convex optimization. In *Proceedings of the International Conference on Machine Learning*, pp. 383–392, 2021.

Balle, B., Barthe, G., and Gaboardi, M. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*, pp. 6280–6290, 2018.

Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473, 2014.

Bu, Z., Wang, Y.-X., Zha, S., and Karypis, G. Automatic Clipping: Differentially private deep learning made easier and stronger. In *Advances in Neural Information Processing Systems*, pp. 41727–41764, 2023a.

Bu, Z., Wang, Y.-X., Zha, S., and Karypis, G. Differentially private optimization on large model at small cost. In *Proceedings of the International Conference on Machine Learning*, pp. 3192–3218, 2023b.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The Secret Sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, pp. 267–284, 2019.

Carvalho, R. S., Vasiloudis, T., Feyisetan, O., and Wang, K. TEM: High utility metric differential privacy on text. In *Proceedings of the SIAM International Conference on Data Mining*, pp. 883–890, 2023.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 1724–1734, 2014.

Coles, S., Bawa, J., Trenner, L., and Dorazio, P. *An introduction to statistical modeling of extreme values*. Springer, 2001.

De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

Du, M., Yue, X., Chow, S. S. M., Wang, T., Huang, C., and Sun, H. DP-Forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 2665–2679, 2023.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284, 2006.

Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., and Vadhan, S. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the annual ACM Symposium on Theory of Computing*, pp. 381–390, 2009.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Feldman, V. Does learning require memorization? A short tale about a long tail. In *Proceedings of the Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020.

Feyisetan, O., Balle, B., Drake, T., and Diethe, T. Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 178–186, 2020.

Gast, J. and Roth, S. Lightweight probabilistic deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3369–3378, 2018.

Golatkar, A., Achille, A., Wang, Y.-X., Roth, A., Kearns, M., and Soatto, S. Mixed differential privacy in computer

vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8376–8386, 2022.

Goodfellow, I. Efficient per-example gradient computations. arXiv preprint arXiv:1510.01799, 2015.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems, 5(4):1–19, 2015.

He, J., Li, X., Yu, D., Zhang, H., Kulkarni, J., Lee, Y. T., Backurs, A., Yu, N., and Bian, J. Exploring the limits of differentially private deep learning with group-wise clipping. In International Conference on Learning Representations, 2023.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

Hu, L., Ni, S., Xiao, H., and Wang, D. High dimensional differentially private stochastic optimization with heavy-tailed data. In Proceedings of the ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pp. 227–236, 2022.

Huang, X. S., Perez, F., Ba, J., and Volkovs, M. Improving Transformer optimization through better initialization. In Proceedings of the International Conference on Machine Learning, pp. 4475–4483, 2020.

Kamath, G., Liu, X., and Zhang, H. Improved rates for differentially private stochastic convex optimization with heavy-tailed data. In Proceedings of the International Conference on Machine Learning, pp. 10633–10660, 2022.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In IEEE International Conference on Data Mining, pp. 197–206, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In International Conference on Learning Representations, 2014.

Kolluri, A., Baluta, T., Hooi, B., and Saxena, P. LPGNet: Link private graph networks for node classification. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1813–1827, 2022.

Krichene, W. and Rendle, S. On sampled metrics for item recommendation. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1748–1757, 2020.

Lee, J. and Kifer, D. Scaling up differentially private deep learning with fast per-example gradient clipping. Proceedings on Privacy Enhancing Technologies, pp. 128–144, 2021.

Li, T., Zaheer, M., Reddi, S., and Smith, V. Private adaptive optimization with side information. In Proceedings of the International Conference on Machine Learning, pp. 13086–13105. PMLR, 2022a.

Li, X., Tramer, F., Liang, P., and Hashimoto, T. Large language models can be strong differentially private learners. In International Conference on Learning Representations, 2022b.

Mattern, J., Weggenmann, B., and Kerschbaum, F. The limits of word level differential privacy. In Findings of the Association for Computational Linguistics, 2022.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52, 2015.

Mohapatra, S., Sasy, S., He, X., Kamath, G., and Thakkar, O. The role of adaptive optimizers for honest private hyperparameter selection. Proceedings of the 36th AAAI conference on artificial intelligence, pp. 7806–7813, 2022.

Morales-Alvarez, P., Hernández-Lobato, D., Molina, R., and Hernández-Lobato, J. M. Activation-level uncertainty in deep neural networks. In International Conference on Learning Representations, 2021.

Narayanan, A. and Shmatikov, V. Robust de-anonymization of large sparse datasets. Proceedings of the IEEE Symposium on Security and Privacy, pp. 111–125, 2008.

Narayanan, A. and Shmatikov, V. De-anonymizing social networks. In Proceedings of the IEEE Symposium on Security and Privacy, pp. 173–187, 2009.

Papernot, N. and Steinke, T. Hyperparameter tuning with renyi differential privacy. In International Conference on Learning Representations, 2022.

Papernot, N., Thakurta, A., Song, S., Chien, S., and Erlingsson, Ú. Tempered sigmoid activations for deep learning with differential privacy. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9312–9321, 2021.

Park, J., Kim, H., Choi, Y., and Lee, J. Differentially private sharpness-aware training. Proceedings of the International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 27204–27224, 2023.

Postels, J., Ferroni, F., Coskun, H., Navab, N., and Tombari, F. Sampling-free epistemic uncertainty estimation using approximated variance propagation. Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2931–2940, 2019.

Ramaswamy, S., Thakkar, O., Mathews, R., Andrew, G., McMahan, H. B., and Beaufays, F. Training production language models without memorizing user data. arXiv preprint arXiv:2009.10031, 2020.

Sajadmanesh, S. and Gatica-Perez, D. Locally private graph neural networks. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2130–2145, 2021.

Shamsabadi, A. S. and Papernot, N. Losing less: A loss for differentially private deep learning. In Proceedings on Privacy Enhancing Technologies Symposium, pp. 307–320, 2023.

Shekhovtsov, A. and Flach, B. Feed-forward propagation in probabilistic neural networks with categorical and max layers. In International Conference on Learning Representations, 2019.

Song, S., Chaudhuri, K., and Sarwate, A. D. Stochastic gradient descent with differentially private updates. In IEEE Global Conference on Signal and Information Processing, pp. 245–248, 2013.

Subramani, P., Vadivelu, N., and Kamath, G. Enabling fast differentially private SGD via just-in-time compilation and vectorization. In Advances in Neural Information Processing Systems, pp. 26409–26421, 2021.

Tramèr, F. and Boneh, D. Differentially private learning needs better features (or much more data). In International Conference on Learning Representations, 2021.

Tramer, F. and Boneh, D. Differentially private learning needs better features (or much more data). In International Conference on Learning Representations, 2021.

Tramèr, F., Kamath, G., and Carlini, N. Considerations for differentially private learning with large-scale public pretraining. arXiv preprint arXiv:2212.06470, 2022.

Utpala, S., Hooker, S., and Chen, P.-Y. Locally differentially private document generation using zero shot prompting. In Findings of the Association for Computational Linguistics: EMNLP, pp. 8442–8457, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ., and Polosukhin, I. Attention is all you need. In Advances in Neural Information Processing Systems, pp. 5998–6008, 2017.

Wang, D., Xiao, H., Devadas, S., and Xu, J. On differentially private stochastic convex optimization with heavy-tailed data. In Proceedings of the International Conference on Machine Learning, pp. 10081–10091, 2020.

Wang, H. and Yeung, D.-Y. Towards bayesian deep learning: A framework and some existing methods. TDKE, 28(12): 3395–3408, 2016.

Wang, H. and Yeung, D.-Y. A survey on Bayesian deep learning. ACM Computing Surveys, 53(5):1–37, 2020.

Wang, H., Shi, X., and Yeung, D.-Y. Natural-parameter networks: A class of probabilistic neural networks. In Advances in Neural Information Processing Systems, pp. 118–126, 2016.

Wang, W., Wang, T., Wang, L., Luo, N., Zhou, P., Song, D., and Jia, R. DPlis: Boosting utility of differentially private deep learning via randomized smoothing. In Proceedings on Privacy Enhancing Technologies Symposium, pp. 163–183, 2021.

Wang, Y.-X., Fienberg, S., and Smola, A. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In Proceedings of the International Conference on Machine Learning, pp. 2493–2502, 2015.

Wang, Y.-X., Balle, B., and Kasiviswanathan, S. P. Subsampled Rényi differential privacy and analytical moments accountant. In Proceedings of the International Conference on Arti cial Intelligence and Statistics, pp. 1226–1235, 2019.

Wei, J., Bao, E., Xiao, X., and Yang, Y. DPIS: An enhanced mechanism for differentially private SGD with importance sampling. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2885–2899, 2022.

Xiao, H., Xiang, Z., Wang, D., and Devadas, S. A theory to instruct differentially-private learning via clipping bias reduction. In IEEE Symposium on Security and Privacy, pp. 2170–2189, 2023.

Xu, P., Kumar, D., Yang, W., Zi, W., Tang, K., Huang, C., Cheung, J. C. K., Prince, S. J., and Cao, Y. Optimizing deeper transformers on small datasets. In Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, ACL/IJCNLP, pp. 2089–2102, 2021.

Yang, X., Zhang, H., Chen, W., and Liu, T.-Y. Normalized/Clipped SGD with perturbation for differentially private non-convex optimization. arXiv preprint arXiv:2206.13033, 2022.

Yousefpour, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., Cormode, G., and Mironov, I. Opacus: User-friendly differential privacy library in PyTorch. In NeurIPS 2021 Workshop Privacy in Machine Learning, 2021.

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., Kulkarni, J., Lee, Y. T., Manoel, A., Wutschitz, L., et al. Differentially private fine-tuning of language models. In International Conference on Learning Representations, 2022.

Yu, Y., Sanjabi, M., Ma, Y., Chaudhuri, K., and Guo, C. Vip: A differentially private foundation model for computer vision. arXiv preprint arXiv:2306.08842, 2023.

Zhang, B., Titov, I., and Sennrich, R. Improving deep transformer with depth-scaled initialization and merged attention. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, pp. 898–909, 2019.

A. Related Work

A.1. Differential Private Deep Learning

Papernot et al. (2021) suggested tempered sigmoid activations to control the gradient norm explicitly, and in turn support faster convergence in the settings of differentially private ML. Mohapatra et al. (2022) studied the intrinsic connection between the learning rate and clipping norm hyperparameters and show that adaptive optimizers like DPAdam enjoy a signi cant advantage in the process of honest hyperparameter tuning. Wei et al. (2022) employed importance sampling (IS) in each SGD iteration for minibatch selection.

One line of work focuses on adaptive optimization of differentially private machine learning. Asi et al. (2021) proposed adaptive stepsizes as a variant of SGD. Li et al. (2022a) uses non-sensitive side information to precondition the gradients, allowing the effective use of adaptive methods in private settings.

Another line of work studies the loss landscape of DP-SGD in comparison to SGD. Wang et al. (2021) rst highlighted the problem of DP-SGD being stuck in local minima due to the training instability. They constructed a smooth loss function that favors noise-resilient models lying in large at regions of the loss landscape. Shamsabadi & Papernot (2023) proposed that loss functions with smaller norm can reduce the impact of clipping and thus create a smoother loss function. Park et al. (2023) made use of sharpnessaware training without additional privacy costs.

The most related work is Li et al. (2022b), which netunes large language models with differential privacy. They propose Ghost Clipping, which is a technique that enables ef cient per-sample gradient clipping without instantiating per-sample gradient. Our Phantom Clipping can be viewed as an extension of Ghost Clipping that additionally handles parameter sharing of the embedding layer. They also introduce the ideaeffective noise multiplierin order to explain the role of batch size in private learning. Oureffective error(Equation (16)) can be viewed as its generalization in order to account for the inherent input sparsity (i.e., only a small portion of tokens appear in one training sequence). Yu et al. (2023) proposed to train a Vision Transformer model with differential privacy from scratch.

Another related work includes Anil et al. (2022), which establishes a baseline for BERT-Large pretraining with DP. They introduce several strategies to help private training, such as large weight decay and increasing batch size schedule. Notably, these strategies are independent yet complementary to the methodologies utilized in this work, thereby offering potential avenues for an integrated approach.

The line of work (Feyisetan et al., 2020; Mattern et al., 2022; Carvalho et al., 2023; Utpala et al., 2023) focuses on privatization by pre-processing on the training texts and then using non-private training while our work focuses on private training with DP-SGD. Results are incomparable because they rely on a non-standard and generalization notion of privacy to preserve utility. The work (Du et al., 2023) proposes to differentially private ne-tuning Language Models in the forward pass, which can also be regarded as privatization by pre-processing, but on the generalized data (i.e., features extracted by neural nets).

Previous work has also considered the differentially private learning algorithms on heavy-tailed data (Wang et al., 2020; Hu et al., 2022; Kamath et al., 2022). This line of research is mainly concerned with differential private stochastic optimization (DP-SCO). Note that the notion of heavy-tailed there is different from the focus of this work. As pointed out in Kamath et al. (2022), the setting they actually consider is dealing with heavy-tailed gradients due to unbounded values in the input data.

B. Phantom Clipping

By unrolling the expression, Algorithm 1 directly follows from the following claim.

Claim B.1. (Phantom Clipping) For $1 \quad i \quad B$, the norm of the per-sample gradient$kg_{i;E}$ k with respect to the shared embedding layer$E$ can be ef ciently evaluated, without instantiatingg_E, by

$$kg_E k_i = \quad h(a_S)_i \quad (a_S)_i^T ; r \ (e_S)_i \quad r \ (e_S)_i^T i^2 + k(r \ e_C)_i k^2 + 2 \quad hr \ (e_S)_i ; (a_S)_i \quad (r \ e_C)_i i \ ^{\frac{1}{2}}; \tag{21}$$

where$h ; i$ is the inner product of two matrices being of the same shape.

Proof. For simplicity, we will omit the per-sample index throughout this proof and simply let batch size$B = 1$ without

loss of generality. From the chain rule, the per-sample gradient with respect to the embedding layer is

$$\begin{aligned} g_E &= \frac{\partial L}{\partial e_S}\frac{\partial e_S}{\partial E} + \frac{\partial L}{\partial e_C}\frac{\partial e_C}{\partial E} \\ &= \underbrace{a_S^T \, r e_S}_{g_E^{(1)}} + \underbrace{a_C^T \, r e_C}_{g_E^{(2)}}; \end{aligned} \tag{22}$$

where $a_s \in \{0,1\}^{L \times M}$ (or $a_c \in \{0,1\}^{M \times M}$) is the one-hot encodings of the input sequences (or those of the candidate tokens for the output probability) in a mini-batch, and $e_s \in \mathbb{R}^{L \times d}$ (or $e_c \in \mathbb{R}^{M \times d}$) be output of the (shared) embedding layer E when fed into a_s (or a_c). Denote the first segment of the right-hand side (RHS) as $g_E^{(1)}$, the second segment of the RHS as $g_E^{(2)}$. Then we have

$$\|g_E\|^2 = \left\|g_E^{(1)} + g_E^{(2)}\right\|^2 = \left\|g_E^{(1)}\right\|^2 + \left\|g_E^{(2)}\right\|^2 + 2\langle g_E^{(1)}, g_E^{(2)}\rangle. \tag{23}$$

Ghost Clipping (Li et al., 2022b) allows us to evaluate $\left\|g_E^{(1)}\right\|_F$ without instantiating $g_E^{(1)}$, the formula is given by

$$g_E^{(1)} = \langle a_S a_S^T, r e_S r e_S^T \rangle. \tag{24}$$

Likewise, we have

$$g_E^{(2)} = \langle a_C a_C^T, r e_C r e_C^T \rangle. \tag{25}$$

With appropriate implementation, a_C is the one-hot encoding of $[1, 2, 3, \ldots, M]$, thus a_C is an identity matrix and Equation (25) can be further simplified as

$$g_E^{(2)} = \langle I, r e_C r e_C^T \rangle = \sum_{j,k} (r e_C)_{j,k} \cdot (r e_C)_{j,k} = \|r e_C\|^2. \tag{26}$$

Note that this simplification saves us the memory footprint $O(BM^2)$ for evaluating $r e_C r e_C^T$ in Equation (25).

Therefore, computing the gradient norm of shared embedding reduces to computing $\langle g_E^{(1)}, g_E^{(2)} \rangle$ in Equation (23),

$$\begin{aligned} \langle g_E^{(1)}, g_E^{(2)} \rangle &= \langle a_S^T \, r e_S, a_C^T \, r e_C \rangle \\ &= \sum_{j=1}^{M} \sum_{k=1}^{d} \sum_{i=1}^{L} (a_S)_{ij} \cdot (r e_S)_{ik} \cdot \left(\sum_{i=1}^{M} (a_C)_{ij} \cdot (r e_C)_{ik} \right) \\ &= \sum_{i_1=1}^{L} \sum_{i_2=1}^{M} \left(\sum_{j=1}^{M} \sum_{k=1}^{d} (a_S)_{i_1 j} \cdot (r e_S)_{i_1 k} \cdot (a_C)_{i_2 j} \cdot (r e_C)_{i_2 k} \right) \\ &= \sum_{i_1=1}^{L} \sum_{i_2=1}^{M} \left(\sum_{j=1}^{M} (a_S)_{i_1 j} \cdot (a_C)_{i_2 j} \right) \left(\sum_{k=1}^{d} (r e_S)_{i_1 k} \cdot (r e_C)_{i_2 k} \right) \\ &= \sum_{i_1=1}^{L} \sum_{i_2=1}^{M} \langle (a_S)_{i_1}, (a_C)_{i_2} \rangle \cdot \langle (r e_S)_{i_1}, (r e_C)_{i_2} \rangle \\ &= \sum_{i_1=1}^{L} \sum_{i_2=1}^{M} \mathbb{I}[i_2 = \text{onehot}^{-1}((a_S)_{i_1})] \cdot \langle (r e_S)_{i_1}, (r e_C)_{i_2} \rangle \\ &= \sum_{i_1=1}^{L} \langle (r e_S)_{i_1}, (a_S)_{i_1} \, r e_C \rangle \\ &= \langle (r e_S), a_S \, (r e_C) \rangle. \end{aligned} \tag{27}$$

Combining Equation (24), Equation (26) and Equation (27) yields the conclusion. □

C. Re-Attention Mechanism

C.1. Derivation of Claim 4.2

Taking the average over the minibatch in Equation (1) can be considered noise reduction $O(1=B)$. Fix the noise multiplier $_{dp}$. As the size of the batch increases, the amount of DP noise incorporated into the parameter decreases correspondingly. Suppose that token i is absent from the current training sequence. Its input embedding will not be activated and thus will not be properly trained in this iteration, but the DP noise will be nevertheless injected into its embedding. The concept of effective error is introduced to account for this phenomenon.

Proof. It reduces to derive the formula for effective batch size B_e in Equation (16). Recall that its de nition is given by

$$B_e = E_{B \overset{i.i.d}{\sim} D^B}\left[\sum_{i=1}^{B} I\left[R_{\cdot}(B_i)\right]\right]: \tag{28}$$

For each layer parameterized by W within the Transformer block, its effective batch size $B_e^W = B$, since $R_W(B_i) = 1$.

For the embedding layer E, its effective batch size is

$$\begin{aligned} B_e^{E_i} &= E_{B \overset{i.i.d}{\sim} D^B}\left[\sum_{j=1}^{B} I\left[R_{E_i}(B_j)\right]\right] \\ &= \sum_{j=1}^{B} E_{B_j \overset{i.i.d}{\sim} D}\left[I\left[R_{E_i}(B_j)\right]\right] \quad \text{(Linearity of Expectation)} \\ &= \sum_{j=1}^{B} E_{B_j \overset{i.i.d}{\sim} D}\left[I\left[\text{token } i \in B_j\right]\right] \\ &= \sum_{j=1}^{B} p_i \\ &= B \quad p_i; \end{aligned} \tag{29}$$

where p_i is the frequency of token i (i.e., the probability of token i's occurrence in data). □

C.2. Propagation of Natural Parameters

For linear transformation $X^{(l)} = X^{(l-1)}W$, we can propagate the variance as

$$\sigma^2_{X^{(l)}} = \sigma^2_{X^{(l-1)}}\sigma^2_W + \sigma^2_{X^{(l-1)}}\mu^2_W + \sigma^2_W\left(\mu_{X^{(l-1)}}\right)^2: \tag{30}$$

For nonlinear activation functions, e.g., $X^{(l)} = \mathrm{ReLU}\left(X^{(l-1)}\right)$, we can propagate the variance as

$$\sigma^2_{X^{(l)}} = \left(\mu\frac{c}{\sqrt{d}}\right)(c^2 + d) + c\sqrt{\frac{d}{2}}\exp\left(-\frac{1}{2}\frac{c^2}{d}\right) \quad c^2; \tag{31}$$

where $(\cdot)$ is the cumulative density function (CDF) of the standard Gaussian distribution, and c and d are the natural parameter of $X^{(l-1)}$.

Lemma C.1. Let X, Y be two independent random variables, $Z = XY$, then the variance of Z can be expressed as

$$\mathrm{Var}[Z] = \mathrm{Var}[XY] = E[X^2]E[Y^2] \quad E[XY]^2: \tag{32}$$

Lemma C.1 directly implies Equation (30) as follows.

Proof. Suppose the linear transformation is given by $X^{(l)} = X^{(l-1)}W$, then we have

$$\begin{aligned}\mathrm{Var}[X^{(l)}] &= E[(X^{(l-1)})^2]E[W^2] - E[X^{(l-1)}W]^2 \\ &= (E[X^{(l-1)}]^2 + \mathrm{Var}[X^{(l-1)}])(E[W]^2 + \mathrm{Var}[W]) - E[X^{(l-1)}]^2 E[W]^2 \\ &= \mathrm{Var}[X^{(l-1)}]\,\mathrm{Var}[W] + E[X^{(l-1)}]^2\,\mathrm{Var}[W] + E[W]^2\,\mathrm{Var}[X^{(l-1)}].\end{aligned} \tag{33}$$

□

Lemma C.2. Let X_1, X_2 be two independent Gaussian random variables, where $X_i \sim N(\mu_i, \sigma_i), i = 1, 2$. Let $Z = \max(X_1, X_2)$.

$$\begin{aligned}E[Z] &= \mu_1 \Phi(\alpha) + \mu_2 \Phi(-\alpha) + \theta\,\phi(\alpha) \\ E[Z^2] &= (\mu_1^2 + \sigma_1^2)\Phi(\alpha) + (\mu_2^2 + \sigma_2^2)\Phi(-\alpha) + (\mu_1 + \mu_2)\theta\,\phi(\alpha);\end{aligned} \tag{34}$$

where $\Phi(\cdot)$ is the cumulative density function (CDF) of the standard Gaussian distribution, $\phi(\cdot)$ is the probability density function (PDF) of the standard Gaussian distribution, $\theta = \sqrt{\sigma_1^2 + \sigma_2^2}$, and $\alpha = (\mu_1 - \mu_2)/\theta$.

Lemma C.2 directly implies Equation (19) as follows.

Proof. Let $X^{(l)} = \mathrm{ReLU}(X^{(l-1)})$ be the ReLU activation. Substitute $\mu_1 = E[X^{(l-1)}]$, $\sigma_1 = \mathrm{Var}[X^{(l-1)}]$, $\mu_2 = \sigma_2 = 0$ into Equation (34). Leveraging $\mathrm{Var}[X^{(l)}] = E[(X^{(l)})^2] - E[X^{(l)}]^2$ yields the conclusion. □

Remark C.3. GELU activation. GELU function is another widely used activation function within Transformer models. GELU can be viewed as a smooth version of ReLU (see Figure 5), where their forward propagation is similar, and the major distinction lies in the numerical behavior of backpropagation. Since error propagation is only concerned with forward propagation behavior, we can also use Equation (19) to approximate the variance of GELU output. Table 1 shows the analytic error propagation for ReLU and GELU activation, compared with the sampling-based results.

Figure 5. GELU activation and ReLU activation.

Table 1. Analytic error propagation for ReLU and GELU activation.

Input	Activation	Sampling-based 10	100	1000	10000	100000	1000000	Analytic
$N(0, 0.01)$	ReLU	4.08e-6	3.60e-5	3.93e-5	3.45e-5	3.40e-5	3.40e-5	3.40e-5
	GELU	2.48e-5	2.69e-5	2.72e-5	2.57e-5	2.50e-0	2.49e-5	
$N(0, 0.1)$	ReLU	0.0030	0.0031	0.0037	0.0034	0.0035	0.0034	0.0034
	GELU	0.0030	0.0025	0.0027	0.0025	0.0026	0.0025	
$N(0, 1)$	ReLU	0.5299	0.2361	0.3649	0.3451	0.3387	0.3418	0.3408
	GELU	0.5525	0.2306	0.3719	0.3506	0.3433	0.3467	

Figure 6. Data in the real-world scenarios exhibits long-tailed (also known as, power-law) distribution.

D. Empirical Evaluation

D.1. Datasets

We conduct experiments on two public recommendation datasets collected from real-world scenarios: MovieLens (Harper & Konstan, 2015) and Amazon (McAuley et al., 2015). Figure 6 shows their data distributions, illustrating the prevalence of long-tailed distributions, where a small number of items are extremely popular and have relatively high frequency while other items occur infrequently. The embedded table above the `long tail' reports the statistics of the two datasets, showing that the two datasets vary signi cantly in size and sparsity.

Figure 6 shows their data distributions, illustrating the prevalence of long-tailed distributions, where a small number of items are extremely popular and have relatively high frequency while other items occur infrequently. The embedded table above the `long tail' reports the statistics of the two datasets, showing that the two datasets vary signi cantly in size and sparsity.

MovieLens. The MovieLens dataset (Harper & Konstan, 2015) is often used in the development and evaluation of collaborative ltering algorithms, which are used to make personalized recommendations based on user behavior. It is a benchmark dataset in the eld of recommender systems due to its size, longevity, and richness of user-item interactions. We use the version (MovieLens-1M) that includes 1 million user behaviors.

Amazon. A series of datasets introduced in (McAuley et al., 2015), comprising large corpora of product reviews crawled from Amazon.com. Top-level product categories on Amazon are treated as separate datasets. We consider the `Games.' category. This dataset is notable for its high sparsity and variability.

We follow (Kang & McAuley, 2018) for the data preprocessing. We use timestamps to determine the sequence order of actions. Each user is associated with a training sequence (i.e., his chronological behavior). We discard users and items with fewer than ve related actions. For data partitioning, the last token of each sequence is left for testing.

It is worth noting that since each user is exclusively associated with exactly one training sample (sequence) in the training data, the DP guarantee we provide is user-level. That is, removing all information pertaining to a speci c user yields an indistinguishable model.

D.2. Model Architecture

We use the standard Transformer encoder described in (Vaswani et al., 2017). The model dimension is set to 64. The number of heads in the Attention Mechanism is set to 1. The number of Transformer blocks is set to 2. Our model adopts a learned (instead of xed) positional embedding. The model size is similar to that in (Ramaswamy et al., 2020), which is suitable for deployment on the user devices.

D.3. Hyperparameters

The number of epochs is set to 100. The batch size is chosen from $\{256, 512, 1024, 2048, 4096\}$. The learning rate is chosen from $\{10^{-3}; 3 \quad 10^{-3}; 5 \quad 10^{-3}; 7 \quad 10^{-3}; 9 \quad 10^{-3}\}$. The dropout rate is 0.2 for MovieLens and 0.5 for Amazon (due to its high sparsity). We use the Adam optimizer with a weight decay of 10^{-5}.

D.4. Evaluation Metrics

HIT@k measures whether the relevant (i.e., ground truth) item is present within the k items in prediction list. It is a binary metric indicating the presence (hit) or absence of relevant items.

$$\text{HIT@k} = \begin{cases} 1; & \text{if the next item is in top-}k\text{ prediction} \\ 0; & \text{otherwise} \end{cases} \tag{35}$$

NDCG@k measures the performance of a recommendation system based on the graded relevance of the recommended items. It is normalized based on the ideal order of items.

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}} \tag{36}$$

where

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} \quad 1}{\log_2(i+1)} \tag{37}$$

Here, rel_i is the relevance score of the item at position i in the recommendation list. Namely, if the i-th item (sorted by prediction score of the model) is equal to the ground truth, $\text{rel}_i = 1$ otherwise 0. IDCG@k (Ideal Discounted Cumulative Gain at k) is the maximum possible DCG@k obtained by placing the most relevant items in the top positions (i.e., the item with highest prediction score is equal to the ground truth). It is calculated similarly to DCG@k but for the ideal order.

We adhere to the evaluation method advocated in (Krichene & Rendle, 2020), i.e., ranking all the items rather than adopting the sampled metrics where only a smaller set of random items and the relevant items are ranked. Note that under this evaluation method the accuracy value is greatly lower than that obtained by sampled metrics, but is more consistent and meaningful.

D.5. Empirical Evaluation of Phantom Clipping

To show the importance of parameter sharing when training Transformer models with DP-SGD, we conduct experiments under the following three settings: (1) parameter sharing of the embedding layer, which aligns with the standard treatment in Transformer; (2) no parameter sharing; and (3) no parameter sharing coupled with a reduced embedding dimension by half. Note that the third setting is included to account for the potential impact of model dimension on accuracy in private training, given the difference in the number of parameters between models with and without parameter sharing. Model performance across different hyperparameters is shown in Figure 7. The consistency and signi cance of the performance improvement brought by parameter sharing during private training are not hard to perceive. The essence of embedding sharing lies in the assumption that, by tying the embedding of the input and output layers, the representation of each token remains consistent throughout its retrieval. This inductive bias enhances the statistical ef ciency of the model, enabling improved generalization. When training with DP-SGD on limited training data, the model must independently uncover this relationship from the noisy gradients with a low signal-to-noise ratio, heightening the convergence challenge.

(a) parameter sharing (b) w/o parameter sharing (c) halved dimension in (b)

Figure 7. Numbers are NDCG(%)@10 (higher is better) of the privately trained models with ε set to 5) on MovieLens (Figure 6). Parameter sharing for the embedding layer yields consistent and signi cant performance gains over the non-sharing setting in private training. The optimal hyperparameter con guration is always using a large batch size (with a large learning rate).

(a) Memory ef ciency (b) Training speed

Figure 8. Empirical speedup on small models, compared to the Ghost Clipping. Left: Phantom Clipping is 10-400 more memory ef cient than Ghost Clipping and is almost as ef cient as non-private training. Right: Phantom Clipping is 4-100 faster than Ghost Clipping, having comparable training speed with non-private training.

Empirical Speedup. As we mentioned, in addition to support standard Transformer models with embedding sharing, our Phantom Clipping can achieve great speedup than Ghost Clipping (Li et al., 2022b) when the model is relatively small. This is because the Ghost Clipping is a general method for all layers, not specialized for the embedding layer. However, when the model size is small, the dominating overhead will be at the embedding layer.

We implement our Phantom Clipping based on AWS's fastDP library[11], which has implemented Ghost Clipping. We then empirically compare our Phantom Clipping with Ghost Clipping in terms of both memory footprint and training speed on real-world datasets[12] (see Figure 6 for details of the datasets). Figure 8a shows the maximum batch size that can t into a Tesla V100 GPU (16 GB of VRAM). It can be seen that our technique is much more memory friendly. It allows up to 450 larger batch size compared with Ghost Clipping on Amazon, almost as large as those in non-private training. Figure 8b shows the training speed on a single Tesla V100 GPU. It allows up to 101 training speedup in practice compared to Ghost Clipping, achieving 0.68 training speed of the non-private version.

D.6. Empirical Evaluation of the Re-Attention Mechanism

Baselines and Implementation Details. We compare our method with vanilla Transformer (Vaswani et al., 2017) (i.e., the one without Re-Attention Mechanism), vanilla Transformer without parameter sharing, GRU (Cho et al., 2014), and LSTM (Hochreiter & Schmidhuber, 1997). For a fair comparison, embedding sharing is applied for all evaluated methods if not explicitly stated. The number of epochs is set to 100, where the rst 20% of epochs are used for learning rate warm-up. After that, we linearly decay the learning rate through the remaining epochs. Following (Bu et al., 2023a; Yang et al., 2022), we normalize the gradients and set the clipping norm to 1, which eliminates the hyperparameter tuning for clipping norm C. For privacy accounting, we x the total training epochs (iterations) and derive the noise required for each iteration from the preset privacy budget. The parameter in DP guarantee is set to $1=$size of dataset.

Table 2. Best results (%) on MovieLens at different privacy levels.

DP Guarantee	" = 5				" = 8				" = 10			
Metric	NDCG@10		HIT@10		NDCG@10		HIT@10		NDCG@10		HIT@10	
GRU	2.26	0.04	4.58	0.09	2.40	0.03	4.75	0.20	2.81	0.03	5.53	0.05
LSTM	2.65	0.07	5.08	0.08	2.76	0.03	5.41	0.06	2.95	0.03	5.55	0.06
TRANSFORMER W/O PS	2.33	0.05	4.47	0.07	2.56	0.03	5.11	0.05	2.74	0.04	5.39	0.08
TRANSFORMER(VANILLA)	4.57	0.26	8.69	0.53	7.05	0.23	13.17	0.37	7.99	0.21	14.82	0.38
OURS	5.88	0.24	11.13	0.43	7.70	0.26	14.31	0.37	8.42	0.22	15.40	0.32
Relative Improvement	29%"		28%"		9.2%"		8.7%"		5.4%"		3.9%"	

Table 2 and Table 3 show the best[13] NDCG@10 and HIT@10 for all the methods on MovieLens and Amazon. The vanilla

[11] https://github.com/awslabs/fast-differential-privacy

[12] Since Ghost Clipping does not support parameter sharing, its results are obtained from training models without embedding sharing. This leads to more model parameters. For a fair comparison, we halve its embedding dimension $d_e = 2$, ending up with a similar number of parameters as in the model with embedding sharing.

[13] Strictly speaking, the process of hyperparameter tuning would cost privacy budget (Papernot & Steinke, 2022), but is mainly of theoretical interest. We perform grid search on learning rate $2 f 10^{-3}; 3 \quad 10^{-3}; 5 \quad 10^{-3}; 7 \quad 10^{-3}; 9 \quad 10^{-3} g$ and batch size $2 f 256; 512; 1024; 2048; 4096 g$ for each method, ensuring fair comparison.

Table 3. Best results (%) on Amazon at different privacy levels.

DP Guarantee	$\varepsilon = 5$				$\varepsilon = 8$				$\varepsilon = 10$			
Metric	NDCG@10		HIT@10		NDCG@10		HIT@10		NDCG@10		HIT@10	
GRU	1.13	0.02	2.46	0.03	1.33	0.02	2.22	0.02	1.47	0.03	2.48	0.02
LSTM	1.19	0.01	2.46	0.04	1.23	0.01	2.46	0.04	1.34	0.01	2.51	0.02
TRANSFORMER W/O PS	1.16	0.01	2.36	0.01	1.20	0.02	2.38	0.01	1.40	0.01	2.47	0.02
TRANSFORMER(VANILLA)	1.37	0.04	2.47	0.10	1.54	0.03	2.77	0.07	1.57	0.03	2.83	0.08
OURS	1.64	0.01	3.01	0.01	1.98	0.05	3.70	0.15	1.99	0.04	3.73	0.11
Relative Improvement	20%		22%		28%		34%		27%		31%	

Transformer outperforms all other baselines, reaf rming its dominance in sequential data modeling due to the Attention Mechanism. Our Re-Attention Mechanism further boosts the performance by around 20% on average. Notably, under a low privacy budget ($\varepsilon = 5$), our method achieves a relative improvement of around 25%, demonstrating its ef cacy in attenuating attention distraction during private training. On MovieLens, as expected, the performance gain increases with decreasing privacy budget (i.e., increasing noise strength during training). This is because larger noise corresponds to more severe attention distraction, which better highlights the Re-Attention Mechanism's advantage. However, on Amazon, our method achieves a smaller relative improvement at $\varepsilon = 5$ than at $\varepsilon = 10$. We suspect that this is due to the differences of the two datasets in terms of sparsity (i.e., density in Figure 6) as well as the inherent hardness of training Transformer (Zhang et al., 2019; Xu et al., 2021; Huang et al., 2020) and the overwhelming DP noise.

Figure 4 shows the model accuracy every ve epochs during training. Evidently, the training dynamics of the vanilla Transformer, impacted by attention distraction, can suffer from high variance and/or substantial uctuation, especially on Amazon. In contrast, our method enjoys faster and smoother convergence, highlighting its training stability under differential privacy.

(a) Ours (b) Vanilla (c) Ours (d) Vanilla

Figure 9. Results of grid search for hyperparameter tuning on Amazon with privacy budget $\varepsilon = 8$.

To study the robustness and sensitivity with respect to the hyperparameters of our method, Figure 9 shows the results of hyperparameter tuning via grid search.[14] For reasonable (along the main diagonal (Tramer & Boneh, 2021)) hyperparameter con gurations, our method signi cantly and consistently outperforms the vanilla Transformer.

We present the visualization of the attention score map in Figure 10.

We randomly draw ve sentences from the dataset MovieLens and visualize their attention matrices with $\varepsilon = 5, 8, 10$ (Figure 10). Recall that as the value of ε increases (indicating a lesser amount of noise), the accuracy of the attention score improves (and consequently, the model performance) will be. With a larger ε (i.e., 10), the attention score distribution more closely aligns with the ground truth. It is clear from Figure 10 that due to the attention distraction caused by DP noise, when ε is low, some tokens receive a larger amount of attention than they should have (compared with $\varepsilon = 10$), thereby resulting in suboptimal model performance.

D.7. Additional Experiments on NLP Task

The main focus of this paper is to, in the context of the introduced modular treatment, initiate a systematic study on the reduction from DP Transformer to (vanilla) DP neural nets (Figure 1). The experiments in the work intend to corroborate

[14] Rather than run an additional differentially private algorithm to report a noisy max (or argmax) (Papernot & Steinke, 2022), we opt for this practice of directly displaying all results due to its transparency and comprehensiveness.

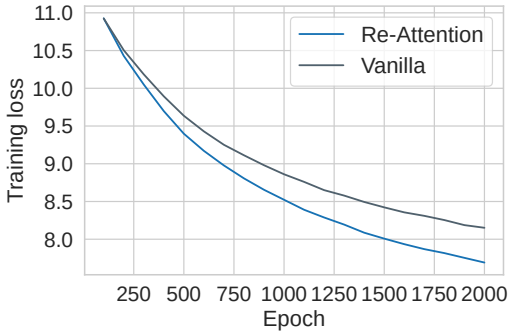
Figure 10. Visualization of attention score with varying σ on MovieLens for five training samples.

the theoretical analysis empirically.

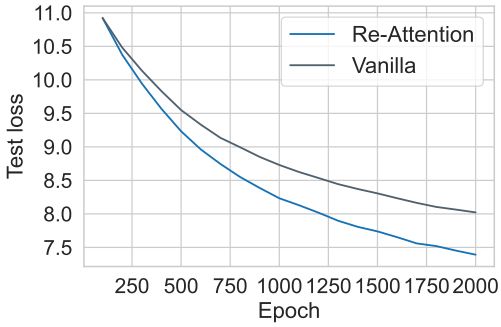
However, when it comes to the natural language processing (NLP) tasks, it inevitably involves the use of large models and potentially more reasonable ways to exploit public data while maintaining provable privacy, which itself is already non-trivial and largely open. Since the focus of this work is on initiating a systematic study on DP Transformers (as a basic primitive rather than a concrete application), we leave that more complicated scenario (and also on various domains) for future work. With that being said, we conducted an experiment of language modeling on TinyShakespeare dataset[15], where a Transformer model is trained from scratch.

We use the standard Transformer encoder described in (Vaswani et al., 2017). The model dimension is set to 384. The number of heads in the Attention Mechanism is set to 6. The number of Transformer blocks is set to 6. The number of epochs is set to 2000. The batch size is chosen from $\{256, 512, 1024, 2048, 4096\}$. The learning rate is chosen from $\{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$. The dropout rate is 0.2. We use the Adam optimizer.

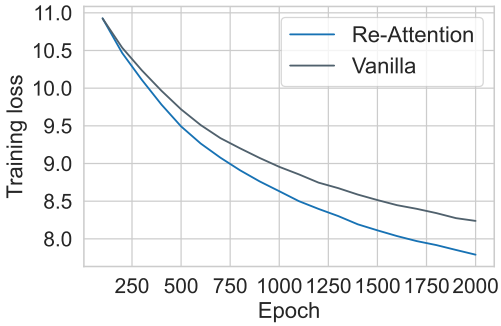
[15] https://paperswithcode.com/dataset/tinyshakespeare



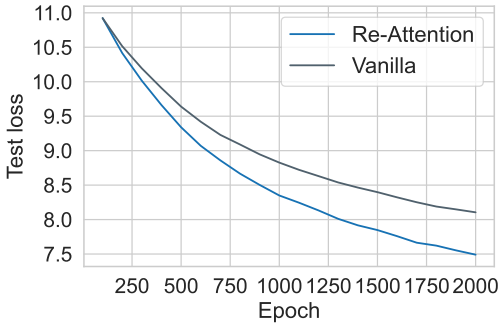
(a) Training loss ($\varepsilon = 10$)



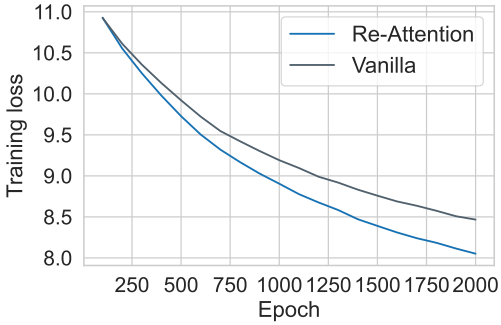
(b) Test loss ($\varepsilon = 10$)



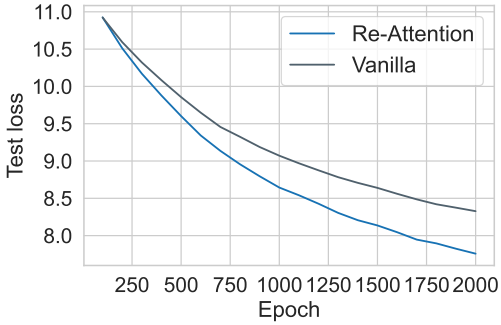
(c) Training loss ($\varepsilon = 8$)



(d) Test loss ($\varepsilon = 8$)



(e) Training loss ($\varepsilon = 5$)



(f) Test loss ($\varepsilon = 5$)