

# DP-Forward: Fine-tuning and Inference on Language Models with Differential Privacy in Forward Pass

Minxin Du

The Chinese University of Hong Kong  
dm018@ie.cuhk.edu.hk

Xiang Yue\*

The Ohio State University  
yue.149@osu.edu

Sherman S. M. Chow<sup>†</sup>

The Chinese University of Hong Kong  
smchow@ie.cuhk.edu.hk

Tianhao Wang

University of Virginia  
tianhao@virginia.edu

Chenyu Huang

Independent  
hcyray@gmail.com

Huan Sun

The Ohio State University  
sun.397@osu.edu

## ABSTRACT

Differentially private stochastic gradient descent (DP-SGD) adds noise to gradients in back-propagation, safeguarding training data from privacy leakage, particularly membership inference. It fails to cover (inference-time) threats like embedding inversion and sensitive attribute inference. It is also costly in storage and computation when used to fine-tune large pre-trained language models (LMs).

We propose DP-Forward, which directly perturbs embedding *matrices* in the forward pass of LMs. It satisfies stringent *local* DP requirements for training and inference data. To instantiate it using the smallest matrix-valued noise, we devise an analytic matrix Gaussian mechanism (aMGM) by drawing possibly non-i.i.d. noise from a *matrix* Gaussian distribution. We then investigate perturbing outputs from different hidden (sub-)layers of LMs with aMGM noises. Its utility on three typical tasks almost hits the non-private baseline and outperforms DP-SGD by up to 7.7pp at a moderate privacy level. It saves 3× time and memory costs compared to DP-SGD with the latest high-speed library. It also reduces the average success rates of embedding inversion and sensitive attribute inference by up to 88pp and 41pp, respectively, whereas DP-SGD fails.

## CCS CONCEPTS

• **Security and privacy** → **Data anonymization and sanitization; Privacy-preserving protocols; Privacy protections.**

## KEYWORDS

Local Differential Privacy, Natural Language Processing, Pre-trained Language Models, Privacy-preserving Fine-tuning and Inference of LMs, Embedding Matrices, Analytic Matrix Gaussian Mechanism

## ACM Reference Format:

Minxin Du, Xiang Yue, Sherman S. M. Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 2023. DP-Forward: Fine-tuning and Inference on Language

\*The first two authors contributed equally to this work.

<sup>†</sup>Corresponding author is from Dept. of Information Engineering, CUHK, Hong Kong.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions.acm.org](https://permissions.acm.org).

CCS '23, November 26–30, 2023, Copenhagen, Denmark.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 979-8-4007-0050-7/23/11...\$15.00  
<https://doi.org/10.1145/3576915.3616592>

Models with Differential Privacy in Forward Pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, November 26–30, 2023, Copenhagen, Denmark. ACM, New York, NY, USA, 18 pages. <https://doi.org/10.1145/3576915.3616592>

## 1 INTRODUCTION

The deep learning architecture of transformer [68] is now gaining popularity in computer vision and has been widely utilized in natural language processing (NLP). Transformer-based language models (LMs), such as BERT [20] and GPT [59, 60], have remarkably achieved state-of-the-art performance in almost every NLP task. They are first pre-trained on massive (public) self-labeled corpora and then fine-tuned for various tasks using much smaller, potentially private corpora. It avoids training from scratch and the possible shortage of task-specific corpora while earning versatility.

Training data contributing to the improved utility of fine-tuned LMs can be sensitive. LMs can (unintentionally) memorize them [12] and become vulnerable to membership inference attacks (MIAs) [63] that identify whether an example is in the training set. Worse still, verbatim training text (e.g., SSNs) can be extracted via only black-box access to GPT-2 [13]. It is also possible to recover personal health information (e.g., patient-condition pairs) from BERT trained over a clinical corpus [42] based on the extraction attack [13].

Differential privacy (DP) [22] has emerged as the *de facto* privacy standard for protecting individual privacy. To thwart MIAs on individuals' *training* data, DP stochastic gradient descent (DP-SGD) [1] can be used. It clips the gradients of each example in a batch and adds random Gaussian noise to the aggregated gradient. It is more general than earlier attempts [17, 18] that focus on convex problems and has been implemented in modern ML frameworks, such as PyTorch and TensorFlow. One can apply it to fine-tune LM-based NLP pipelines while ensuring *example-level* privacy, assuming each individual contributes an example, typically a sequence-label pair.

Unfortunately, DP-SGD often uses a *trusted* party to curate users' sensitive training data. Although it can be done distributively [9, 50] via secure aggregation [15] with extra costs and trust assumptions, it offers *central* DP (CDP) at its core.<sup>1</sup> Instantiating *per-example* gradients as large as *entire* pipelines (e.g., >110M parameters for BERT-Base) is obviously costly. Moreover, maintaining the utility of pipelines trained by the noisy aggregated one is tricky due to

<sup>1</sup>Distributed DP-SGD adds local noise too small to achieve LDP. But it is protected by secret sharing. When *all* shares are aggregated, they cancel out each other, assuming an honest majority. It thus faces a "synchronization" issue begging for identification and recovery mechanisms with computation and communication overheads [9].

the dimensional “curse.” A recent study [79, Table 4] shows that the average accuracy in fine-tuning LMs for four NLP tasks at moderate privacy is 65.7% (vs. 91.8% without DP). Finally, the inference-time embeddings are not perturbed by the noise added during training, leaving *inference queries* vulnerable to various recovery attacks [56, 64], ranging from sensitive attributes (e.g., authorship) to raw text.

## 1.1 Natural Privacy via Perturbing Embeddings

We propose DP-Forward, a radically different approach that perturbs *forward-pass* signals: Users can locally inject noise into the *embeddings* of (labeled) sequences before sharing them for training, in contrast to perturbing gradients in back-propagation (possibly by an untrusted party). It is meant for provable *local* DP (LDP) guarantees, thus protecting against stronger adversaries than DP-SGD.

Our approach also naturally fits the federated learning (FL) setting that does not gather users’ data but with substantial differences – FL typically shares *noiseless* local model updates. Note that any subsequent computation (e.g., gradient computation) on noisy embeddings incurs no extra privacy loss due to the free post-processing of LDP. One might force DP-SGD to offer LDP by adding “enough” noise to the *orders-of-magnitude larger* per-example gradient from a user, but it may yield unusable models at a similar privacy level.

DP-Forward also extends its applicability to *inference* via adding noise to users’ test-sequence embeddings, ensuring LDP as in training. As a “side” benefit, it can effectively mitigate emerging embedding-based privacy risks [56, 64] beyond MIAs.

It is evident that the design goals of DP-Forward naturally align in tandem with our overarching objectives: LDP (vs. CDP), more direct protection of raw data (vs. gradients) against new threats [56, 64], and can be as efficient as regular non-private training (allowing batch processing of noisy embeddings). The foundation supporting these desiderata, unfortunately, was unavailable. A dedicated mechanism to perturb the forward-pass signals is indispensable.

Specifically, we need to derive noises for embeddings of training/inference text sequences obtained through the forward pass of LM-based pipelines as a real- and matrix-valued function. One might adopt the classical Gaussian mechanism (GM) [23] to add i.i.d. noise drawn from a univariate Gaussian distribution. Yet, GM calibrates its noise variance based solely on a *sufficient condition* for DP, and its variance formula is not applicable to a low privacy regime [7]. Another candidate is the matrix-variate Gaussian (MVG) mechanism [14], tailored for matrix-valued data: It exploits possibly *non-i.i.d.* noise from a *matrix* Gaussian distribution to perturb more important rows/columns less. Although it may show better utility over GM [14], it is still sub-optimal due to the sufficient condition.

To optimize MVG, we propose an analytic matrix Gaussian mechanism (aMGM) by integrating a *necessary* and *sufficient* condition from the analytic GM (aGM) [7] for non-i.i.d. noise calibration. Our challenge lies in manipulating the two covariance matrices instead of a single variance. We deduce a constraint only on the *two smallest singular values* (Section 4.2), indicating that i.i.d. noise (as in aGM) may already be optimal for general applications like DP-Forward.<sup>2</sup>

A transformer-based pipeline contains an input embedding layer, encoders, and task layers. All these layers prominently manipulate

embeddings of text inputs in training and subsequent inference. We investigate adding aMGM noise to embeddings output by any hidden (sub-)layer before task layers (Figure 1). To ensure *sequence-level* LDP, we need to estimate the  $L_2$ -sensitivity [23] of “pre-noise” functions for *any* two sequences. It is non-trivial since the functions can include different (sub-)layers that may not even be Lipschitz [39]. Our strategy is to normalize the function outputs to have a fixed Frobenius (or  $L_2$ ) norm, similar to gradient clipping [1]. It works especially well for deeper sub-layers, achieving comparable task accuracy to the non-private baseline (Section 5). For the first few (sub-)layers, we also make two specializations in relaxing LDP to the token level, elaborated in Appendix A.2, to improve accuracy.

## 1.2 Our Contributions

Motivated by prevailing privacy concerns in LM fine-tuning and inference and inherent shortcomings of DP-SGD, we initiate a formal study of an intuitive but rarely studied approach and explore its integration with a transformer-based NLP pipeline. Specifically:

- 1) We propose DP-Forward fine-tuning, which perturbs the forward-pass embeddings of *every* user’s (labeled) sequence. It offers more direct protection than DP-SGD perturbing aggregated gradients. Its provable guarantee (Theorem 1) is a new sequence-level LDP notion (SeqLDP, Definition 4), with the more stringent  $(\epsilon, \delta)$ -LDP guarantee to hold w.r.t. only sequences. Moreover, DP-Forward can naturally extend to inference, ensuring the standard LDP (Theorem 3) for test sequences without labels, whereas DP-SGD cannot.
- 2) To instantiate an optimal output perturbation mechanism for DP-Forward, we propose aMGM, owning independent interests for any matrix-valued function. By exploiting a necessary and sufficient DP condition from aGM [7], it can draw possibly non-i.i.d. noise from a *matrix* Gaussian distribution like MVG [14] while producing orders-of-magnitude smaller noise for high-dimensional data (Section 5.3).
- 3) We conduct experiments<sup>3</sup> on three typical NLP tasks in Section 5, showing how crucial hyperparameters (e.g., the sequence length) impact task accuracy. To fairly compare with DP-SGD on privacy-vs.-utility: i) We perturb labels by the randomized response [70] such that DP-Forward fine-tuning offers the standard LDP for sequence-label pairs (Theorem 2). ii) We “translate” DP-Forward with standard LDP to (example-level) CDP (as offered by DP-SGD) via shuffling [25]. Our accuracy gain (for deep-layer DP-Forward instantiations) is up to 7.7 percentage points (pp), compared to DP-SGD or its recent improvements [78, 79] (reviewed in Section 7.3), at a similar privacy level. Efficiency-wise, DP-SGD incurs  $>3\times$  time and GPU-memory costs even with the latest Opacus library [77].
- 4) We evaluate three classes of privacy threats. Like DP-SGD, DP-Forward (including the two token-level designs in Appendix A.3) can effectively defend against sequence-level MIAs, but only DP-Forward can thwart the two threats on (inference-time) embeddings. Specifically, Section 6 shows that DP-SGD *totally* fails in two embedding inversion attacks, while DP-Forward remarkably reduces their success rates by up to 88pp. For a neural-network-based attribute inference attack, DP-SGD reduces its success rates by only 15pp on average, while DP-Forward achieves  $\sim 41$ pp reduction, making the attack predict like assigning all labels to the majority class.

<sup>2</sup>With extra assumptions, dedicated allocation of other singular values by optimizing/maximizing utility functions specific to applications could help.

<sup>3</sup>Our code is available at <https://github.com/xiangyue9607/DP-Forward>.

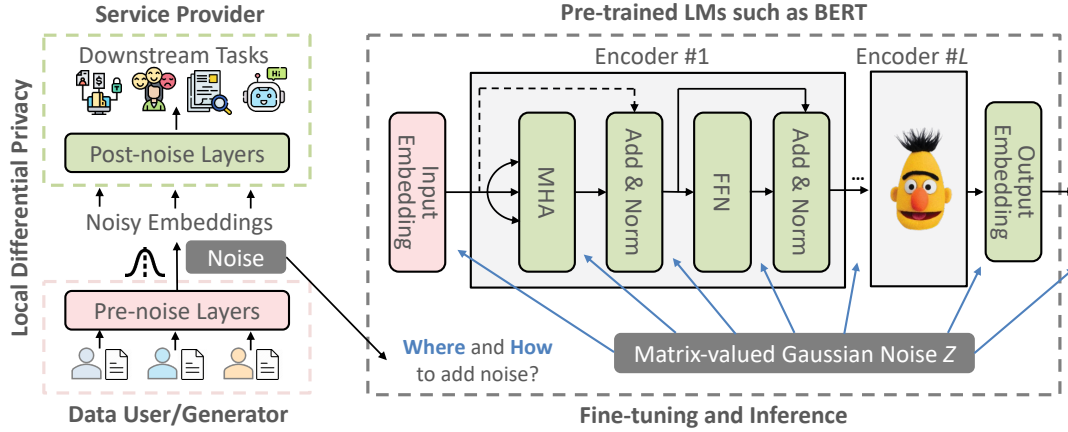


Figure 1: A typical NLP pipeline built atop a pre-trained LM such as BERT with our matrix-valued Gaussian noise layer

In short, DP-Forward is a better alternative to DP-SGD in training (and testing) deep-learning models, e.g., gigantic LM-based ones.

## 2 PRELIMINARIES AND NOTATIONS

### 2.1 Transformer Encoders in BERT

Modern transformer-based LMs, including BERT [20] and GPT [59], are first pre-trained on enormous unannotated (public) corpora to learn contextualized text representations. Later, they can be fine-tuned for various downstream NLP tasks (e.g., sentiment analysis, question answering) using much smaller, task-specific datasets.

We consider BERT (Figure 1), which comprises a stack of  $L$  identical layers (i.e., bidirectional transformer encoders [68]). Each layer has two sub-layers: the dot-product *multi-head attention* (MHA) [68] with  $h$  heads and a *feed-forward network* (FFN). Each sub-layer has an extra residual connection, followed by layer normalization [6].

Let  $X = \langle x_i \rangle_{i=1}^n$  be an input sequence of  $n$  tokens (e.g., characters, words, sub-words, q-grams), where  $x_i$  is from a vocabulary  $\mathcal{V}$ . The input embedding layer first maps each  $x_i$  to its representation in  $\mathbb{R}^d$ , which is the sum of the token, segment, and position embeddings. We re-use  $X$  to represent the hidden *embedding matrix* in  $\mathbb{R}^{n \times d}$ . For each of  $h$  attentions  $\text{Att}_i$  in the MHA layer, we derive the query, key, and value matrices  $Q, K, V \in \mathbb{R}^{n \times d/h}$  ( $h$  divides  $d$ ) by multiplying  $X$  with head-specific weights  $W^Q, W^K, W^V \in \mathbb{R}^{d \times d/h}$ . Its output is

$$\text{Att}_i(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d/h}}\right)V, \forall i \in [1, h].$$

The input to  $\text{softmax}(\cdot)$  is an  $n \times n$  matrix of pairwise dot products. Finally, MHA concatenates (denoted by  $\parallel$ ) all the head outputs into a matrix in  $\mathbb{R}^{n \times d}$ , right multiplied by a projection matrix  $W^O \in \mathbb{R}^{d \times d}$ :

$$\text{MHA}(X) = [\text{Att}_1 \parallel \dots \parallel \text{Att}_h] W^O.$$

FFN is composed of two linear mappings with a ReLU activation in between. It separately and identically operates on each  $x_i \in [1, n]$ ,

$$\text{FFN}(x_i) = \text{ReLU}(0, x_i W_1 + b_1) W_2 + b_2,$$

where  $W_1, W_2, b_1$ , and  $b_2$  are trainable matrix/vector-valued parameters. Its output on  $X$  is  $\text{FFN}(X) = [\text{FFN}(x_1)^\top \parallel \dots \parallel \text{FFN}(x_n)^\top]$ . The residual connection for sub-layers is  $X + \text{MHA}(X)/\text{FFN}(X)$ .

The layer normalization  $\text{LN}(x_i)$  normalizes all  $x_i$  entries to have zero mean and unit variance using an extra scale-then-shift step.

At the output of the final encoder, the hidden embedding matrix is reduced to a sequence feature in  $\mathbb{R}^{1 \times d}$ . Standard reduction methods include mean pooling [61] (computing  $\sum_{i=1}^n x_i/n$ ) or taking the last embedding of a special token [CLS] for classification [20].

The pre-training of BERT is based on two self-supervised tasks: *masked language model* (MLM) and next sentence prediction [20]. We adopt MLM: It randomly masks out some tokens, indexed by  $\mathcal{I}$ , in an input sequence  $X$ . The objective is to predict those masked tokens using their context by minimizing the cross-entropy loss

$$L_{\text{MLM}} = - \sum_{i \in \mathcal{I}} \log \Pr[x_i | \hat{X}; \theta], \text{ with } \hat{X} = X \setminus \{x_i | i \in \mathcal{I}\}, \quad (1)$$

where  $\theta$  denotes all the parameters of BERT transformer encoders.

### 2.2 (Local) Differential Privacy

DP [22] is a rigorous, quantifiable privacy notion. It has two popular models, *central* and *local*. In central DP, a *trusted* data curator accesses the set  $X$  of all individuals' raw data and processes  $X$  by a randomized mechanism  $\mathcal{M}$  with some random noise. Formally:

**Definition 1 (Central DP).** For privacy parameters  $\epsilon \geq 0$  and  $0 \leq \delta \leq 1$ ,  $\mathcal{M}$  fulfills  $(\epsilon, \delta)$ -DP if, for all neighboring datasets  $X$  and  $X'$  (denoted by  $X \approx X'$ ) and any subset  $O$  of the outputs of  $\mathcal{M}$ ,

$$\Pr[\mathcal{M}(X) \in O] \leq e^\epsilon \Pr[\mathcal{M}(X') \in O] + \delta.$$

We call it  $\epsilon$ -DP or pure DP when  $\delta = 0$ .

The neighboring notion is application-dependent (to be discussed in Section 3.1). Typically, it involves the “replace-one” relation:  $X'$  can be obtained from  $X$  by replacing a single individual's data point (e.g., a sequence-label pair). CDP offers plausible deniability to any individual in a dataset. In contrast, local DP (LDP) [38] removes the trusted curator, allowing individuals to locally perturb their data using  $\mathcal{M}$  before being sent to an untrusted aggregator for analytics.

**Definition 2 (Local DP).** For  $\epsilon \geq 0, 0 \leq \delta \leq 1$ ,  $\mathcal{M}$  is  $(\epsilon, \delta)$ -LDP if, for any two inputs  $X, X'$  and any possible output subset  $O$  of  $\mathcal{M}$ ,

$$\Pr[\mathcal{M}(X) \in O] \leq e^\epsilon \Pr[\mathcal{M}(X') \in O] + \delta.$$

Similarly, we call it  $\epsilon$ -LDP when  $\delta = 0$ .

**Table 1: Acronyms (newly proposed ones are marked with \*)**

NLP	Natural Language Processing
LM	Language Model
BERT	Bidirectional Encoder Representations from Transformers
MLM	Masked Language Modeling
MHA	Multi-Head Attention
FFN	Feed-Forward Network
MIA	Membership Inference Attack
DP-SGD	Differentially Private Stochastic Gradient Descent
PLRV	Privacy Loss Random Variable
(C)DP	(Central) Differential Privacy
LDP	Local Differential Privacy
Seq(L)DP*	Sequence (Local) Differential Privacy
GM	Gaussian Mechanism
RR	Randomized Response
MVG	Matrix-Variate Gaussian (Mechanism)
aGM	Analytic Gaussian Mechanism
aMGM*	Analytic Matrix Gaussian Mechanism

*Privacy Loss Random Variable (PLRV)*. For a specific pair of inputs  $X \simeq X'$ , the *privacy loss* (or the “actual  $\epsilon$  value”) [7] incurred by observing an output  $O$  is the log-ratio of two probabilities:

$$\mathcal{L}_{\mathcal{M}, X, X'}(O) = \ln \frac{\Pr[\mathcal{M}(X) = O]}{\Pr[\mathcal{M}(X') = O]}.$$

When  $O$  varies according to  $\mathcal{M}(X)$ , we get the PLRV  $\mathcal{L}_{\mathcal{M}, X, X'}$ . A helpful way to work with DP is to analyze tail bounds on PLRVs [23], which we utilize to build our proposed mechanism in Section 4.2.

DP has two desirable properties: *free post-processing* and *composability*. The former means that further computations on the outputs of an  $(\epsilon, \delta)$ -DP mechanism incur no extra privacy loss. The latter allows us to build more complicated mechanisms atop simpler ones: sequentially (and adaptively) running an  $(\epsilon, \delta)$ -DP mechanism for  $k$  times on the same input is at least  $(k\epsilon, k\delta)$ -DP. The two properties also hold for LDP when considering a dataset has only one row.

An output perturbation mechanism  $\mathcal{M}$  for a matrix-valued function  $f : X \rightarrow \mathbb{R}^{n \times d}$  is given by computing  $f$  on the inputs and then adding random noise drawn from a random variable to its outputs.

**Gaussian Mechanism (GM)**. For  $(\epsilon, \delta)$ -DP, a typical instance of  $\mathcal{M}$  is the classical GM [23], which adds noise  $Z \in \mathbb{R}^{n \times d}$  with each entry i.i.d. drawn from a univariate Gaussian distribution  $\mathcal{N}(0, \sigma^2)$ . The variance  $\sigma^2 = 2 \ln(1.25/\delta) S_2^2(f)/\epsilon^2$  with the  $L_2$ -sensitivity:

$$S_2(f) = \sup_{X \simeq X'} \|f(X) - f(X')\|_F,$$

where  $\|\cdot\|_F$  denotes the matrix Frobenius norm [34].

Table 1 summarizes the acronyms throughout this work.

### 3 DP-FORWARD

We study BERT-based pipelines as an example due to their superior performance in classification tasks. DP-Forward can be readily applied to other (transformer-based) NLP or computer vision models that involve matrix-valued computation during the forward pass.

Suppose each user holds a sequence-label pair  $(X, y)$  or only  $X$  for fine-tuning or testing a pipeline at an *untrusted* service provider. Sharing redacted  $X$  (with common PII removed) or its feature, a non-human-readable real-valued embedding matrix, is leaky [56, 64, 67].

For DP-Forward training, users perturb their embedding matrices locally to ensure (new notions of) LDP before being shared, and they should also perturb the corresponding labels if deemed sensitive (Section 3.4). We explore different options for splitting pipelines into *pre-noise* functions  $f(\cdot)$  and *post-noise* processing  $p(\cdot)$  in Section 3.2: Users can access  $f(\cdot)$  to derive embedding matrices, perturbed by an output perturbation mechanism  $\mathcal{M}$  (e.g., GM); the service provider runs  $p(\cdot)$  on noisy (labeled) embeddings for fine-tuning (Section 3.3) or pre-training (Section 3.6). The challenge lies in analyzing  $S_2(f)$  for different pipeline parts, which we address by normalizing  $f(\cdot)$ .

DP-Forward can be naturally used to protect *inference* sequences (Section 3.5), unlike DP-SGD. It exploits the free post-processing (i.e., inference works on noisy embeddings), incurring minimal changes to pipelines with the extra “plug-and-play” noise layer.

#### 3.1 Notions of Sequence (Local) DP

Embeddings  $f(X)$  encode semantic information of input sequences  $X$ , each of which has  $n$  tokens (Section 2.1). Fine-tuning (or subsequent inference of) NLP pipelines essentially processes  $f(X)$ . DP-Forward fine-tuning protects *every*  $X$  by an output perturbation mechanism  $\mathcal{M}$  over  $f(X)$ , in contrast to DP-SGD, which perturbs *aggregates* of gradients  $f'(X, y)$  over  $X$  and label  $y$ . Simply put, our  $(\epsilon, \delta)$ -LDP holds for  $X$  while DP-SGD provides CDP for  $(X, y)$ .

Sequence-only protection is meaningful since sequences often convey (implicit) sensitive information (e.g., authorship), whereas labels (e.g., a single bit denoting positive/negative) can be public. We defer to Section 3.4 for achieving “full” LDP over  $(X, y)$ . To bridge the gap between theoretical guarantees of DP-SGD and DP-Forward, we first define sequence DP<sup>4</sup> (SeqDP) in the *central* setting.

**Definition 3 (SeqDP)**. For  $\epsilon \geq 0, 0 \leq \delta \leq 1$ ,  $\mathcal{M}$  is  $(\epsilon, \delta)$ -SeqDP, if  $\forall X \simeq X'$  that only differ in a sequence at some index  $i$ :  $(X_i, y_i) \in X$  and  $(X'_i, y_i) \in X'$ ,  $\forall X_i, X'_i$ , and any possible output subset  $O$ ,

$$\Pr[\mathcal{M}(X) \in O] \leq e^\epsilon \Pr[\mathcal{M}(X') \in O] + \delta.$$

**3.1.1 Label DP.** The recently proposed notion of label DP [26, 31] is originally studied in PAC learning [16]. It only protects *labels* (not the corresponding inputs/images):  $(\epsilon, \delta)$ -DP is only w.r.t. labels.

Our SeqDP is “more secure” than or at least “complements” label DP, which has an inherent flaw [11]: As labels typically rely on their sequences (but not vice versa), it is very likely to recover the true labels from the raw sequences, even if the labels are protected (by any label-DP mechanism). The follow-up [72] shows the impossibility of label protection under label DP even with arbitrarily small  $(\epsilon, \delta)$  when models generalize. Moreover, labels can be absent (e.g., inference or self-supervised learning), for which SeqDP upgrades to the standard  $(\epsilon, \delta)$ -DP, whereas label DP is simply inapplicable.

**3.1.2 Sequence Local DP (SeqLDP).** We further define SeqLDP, the *local* counterpart of sequence DP. Note that the above discussion of label DP in relation to SeqDP also carries over to SeqLDP.

**Definition 4 (SeqLDP)**. For  $\epsilon \geq 0, 0 \leq \delta \leq 1$ ,  $\mathcal{M}$  satisfies  $(\epsilon, \delta)$ -SeqLDP, if  $\forall X, X'$  with the same  $y$ , and any possible output subset  $O$ ,

$$\Pr[\mathcal{M}(X, y) \in O] \leq e^\epsilon \Pr[\mathcal{M}(X', y) \in O] + \delta.$$

<sup>4</sup>One could generalize it to “feature” (or “input”) DP, as DP-Forward also allows other types of features beyond embeddings (and its essence is input-only privacy). To keep our focus on NLP, we use “sequence” here. (PixelDP [41] treats pixels as image features.)

In theory, SeqLDP remains a strong notion (like the standard LDP). It is meant to be information-theoretic protection on sequence and bounds the indistinguishability of *any*  $X, X'$  (differing by up to  $n$  tokens), and hence governing the “usefulness” of noisy embeddings.

**3.1.3 Sequence-Level SeqLDP vs. Token-Level SeqLDP.** In practice, as a strong notion balancing seemingly conflicting requirements (ideal theoretical guarantees and empirical utility), attaining a meaningful range of  $\epsilon$  for SeqLDP is a struggle. Adding Gaussian noise to the outputs of  $f(\cdot)$  for  $(\epsilon, \delta)$ -SeqLDP requires bounding the  $L_2$ -sensitivity  $S_2(f)$ ,  $\forall X, X'$ . Our approach is to *normalize* the outputs (with extra benefits elaborated in Section 3.2), similar to clipping gradients in DP-SGD. It generally works better when  $f(\cdot)$  has more layers (at the same meaningful range of  $\epsilon$ ) since fewer (trainable) parameters/layers of  $p(\cdot)$  are “affected” by the noisy outputs.

Unfortunately, when  $f(\cdot)$  includes the first few layer(s), e.g., only the input embedding layer is available to the users (say, for saving user-side storage and computation overheads), it leads to poor utility. As a comprehensive study, we resort to *row-wise* normalization with the (composition of) Lipschitz constants [39] to maintain utility for those cases.<sup>5</sup> In contrast to the general normalization, it aims for weaker SeqLDP at the *token* level (cf. event-level vs. user-level LDP [81]), a finer granularity in the “protection hierarchy,” protecting any *neighboring* sequences (vs. datasets) differing in any single token (vs. sequence). Details are deferred to Appendix A.

## 3.2 Our Approach for Sequence LDP

DP-Forward in our paper (except Appendix A) applies the general normalization approach to any  $f(\cdot)$  for *sequence-level* (Seq)LDP.

Let  $f(\cdot)$  be an arbitrarily deep forward pass, ranging from the first (input embedding) layer itself to all but the last (task) layer in a BERT-based pipeline (Figure 1). Correspondingly, let  $p(\cdot)$  be the remaining layers, ranging from the last task layers themselves to all but the first (input embedding) layer. Every sequence  $X$  becomes an embedding matrix  $f(X) \in \mathbb{R}^{n \times d}$  at the output of layers in encoders or  $\mathbb{R}^{1 \times d}$  before task layers (Section 2.1). To offer  $(\epsilon, \delta)$ -SeqLDP, we adopt a suitable output perturbation mechanism  $\mathcal{M}$ , such as GM, considering that a dataset has only one labeled sequence.

Since  $\mathcal{M}$  can work on the output of *any* hidden layer, estimating  $S_2(f)$  is non-trivial. Specifically, MHA itself, let alone more layers included, is not Lipschitz continuous, meaning its outputs can change arbitrarily for even slight input variation [39]. To address this, our approach is to normalize or clip the function outputs:

$$\|f(\cdot)\|_F = C \text{ or } f(\cdot) / \max(1, \frac{\|f(\cdot)\|_F}{C})$$

as in DP-SGD [1], where  $C$  is a tunable parameter. We then have  $S_2(f) = 2C$ . Such normalization makes task utility less “sensitive” to the choice of  $C$  since signal and noise increase proportionally with  $C$ , whereas the signal may be unchanged when  $f(\cdot)$  is not clipped. It also has many other benefits, such as stabilizing training, avoiding overfitting, and accelerating convergence [2]. Hence, we resort to normalization in our experiments. One can then calibrate Gaussian noise  $Z$  and derive  $f(X) + Z$  for the post-noise layers  $p(\cdot)$ .

<sup>5</sup>One might also resort to the weaker random DP [33] –  $(\epsilon, \delta)$ -DP holds on all but a small  $\gamma$ -proportion of “unlikely”  $X \approx X'$  for an extra parameter  $\gamma \in (0, 1)$ . It is useful when the global sensitivity is hard to compute. Exploring it is left as future work.

Note that we remove the residual connection when adding noise to the output of the *first* MHA layer to avoid  $p(\cdot)$  reaccessing  $X$  (dashed arrow, Figure 1) to maintain free post-processing. This may lead to instability (e.g., gradient vanishing) [79], but it can be mitigated by pre-training new BERT without such a residual connection to keep consistent with later fine-tuning/inference.

DP-Forward using GM suffers from the “curse of dimensionality” when  $d$  is large (e.g., 768 for BERT-Base). To alleviate the curse, we can append two linear maps,  $M_1, M_2 \in \mathbb{R}^{d \times d'}$  with  $d' \ll d$ , such that  $f(\cdot)$  and  $p(\cdot)$  respectively have  $M_1$  and  $M_2$ . Both maps are randomly initialized and updated like other weights using gradients. The raw embedding matrix is first right multiplied by  $M_1$ , leading to  $\mathbb{R}^{n \times d'}$  or  $\mathbb{R}^{1 \times d'}$ , before being normalized. Our privacy guarantee will not be affected since  $S_2(f)$  remains the same. We then use  $M_2$  to restore the dimensionality to be compatible with the raw pipeline;  $M_2$  incurs no extra privacy loss due to the free post-processing. Nevertheless, it needs dedicated efforts to modify the pipeline; dimension-reduced embedding matrices may also lose useful information, degrading task utility. We thus make  $M_1$  and  $M_2$  optional (see Section 5.2).

## 3.3 DP-Forward Fine-tuning

Suppose we use a raw, public BERT checkpoint<sup>6</sup> for fine-tuning. In the forward pass of the  $i$ -th ( $i \geq 1$ ) step, it offers the latest  $f^{(i-1)}(\cdot)$  to a *batch* of users, mimicking the regular mini-batch SGD.  $f^{(0)}$  is from the raw checkpoint. Users are randomly chosen (without replacement), and their number is a fixed parameter. Users in the batch individually compute their noisy embeddings  $f^{(i-1)}(X) + Z$  to ensure SeqLDP (Theorem 1). They then send them with unperturbed labels  $y$  to the service provider, who runs  $p^{(i-1)}(\cdot)$  over  $(f^{(i-1)}(X) + Z, y)$  to compute the batch loss; any post-processing of embeddings under SeqLDP incurs no extra privacy degradation on  $X$ .  $p^{(0)}$  here includes the rest raw BERT part and randomly initialized task layers.

During the back-propagation, the service provider can update  $p^{(i-1)}(\cdot)$  to  $p^{(i)}(\cdot)$  via the gradient (derived from the loss and noisy embeddings) of the post-noise layers. To avoid accessing users’ raw  $X$ , it needs to freeze the pre-noise layers  $f^{(i-1)}(\cdot)$  as  $f^{(0)}$ . Parameter freezing is compatible with the more recent zero-shot or in-context learning paradigm [52]. It is useful when models are gigantic and full fine-tuning is expensive. However, the more layers are frozen, the worse the utility might be (even in non-private settings).

There are two general ways to update  $f^{(i-1)}(\cdot)$  securely: i) We can assume an extra trusted party (as in DP-SGD), but it becomes central DP. ii) Users can first derive the gradients for the layers inside  $f^{(i-1)}(\cdot)$  locally on their  $X$  and then resort to secure aggregation [9] for global updates at the service provider. However, it is costly. For better utility, we update  $f^{(i-1)}(\cdot)$  in experiments, requiring us to consider privacy degradation across different *epochs* due to the composability (as detailed below). Dedicated approaches (that balance efficiency, privacy, and utility) are left as future work.

<sup>6</sup>Using noisy BERT for fine-tuning (and subsequent inference) is deferred to Section 3.6.

**Theorem 1.** *Let  $f(\cdot)$  be the pre-noise function (of BERT-based pipelines) and  $\mathcal{M}$  be GM with  $\epsilon \geq 0, 0 \leq \delta \leq 1$ . DP-Forward fine-tuning running  $\mathcal{M}$  on normalized/clipped  $f(\cdot)$  ensures  $(\epsilon, \delta)$ -SeqLDP.*

The proof follows that of GM [23]. The crux is that  $S_2(f), \forall X, X'$  is given by the output normalization, independent of the inputs.

**Privacy Accounting.** An epoch refers to an entire transit of the private training corpus. Every  $X$  is used once per epoch. The number of epochs  $k$  is a hyperparameter, which is typically small. Repeated applications of GM over the same  $X$  ask for estimating the overall privacy loss due to the composability (unless freezing  $f$  for re-using  $f(X) + Z$ ). The well-known moments accountant [1] (or its generalization to Rényi DP [53]) only provides a loose *upper* bound, which is even inapplicable if unbounded moments exist. Gaussian DP [10] proposes an accountant based on the central limit theorem. Yet, it leads to significant underestimation by a *lower* bound. Instead, we resort to a recent numerical accountant [32], which outperforms RDP or GDP by approximating the true overall  $\epsilon$  to arbitrary accuracy. It composes the privacy curve of a mechanism by truncating and discretizing PLRVs with their PDFs convoluted by FFT [32].

### 3.4 DP-Forward with Shuffling versus DP-SGD

DP-Forward ensures SeqLDP for fine-tuning, while DP-SGD offers *central* DP (for sequence-label pairs). To facilitate a fair comparison (on privacy-utility tradeoffs), we make two changes. First, we also perturb the labels with a suitable mechanism for the standard LDP, *i.e.*, extending the protection from sequence to sequence-label pairs. Second, we use shuffling [25] to “translate” our (label-protected) DP-Forward with LDP to claim (example-level) CDP as DP-SGD.

**Discrete Labels Perturbation.** For most NLP tasks, *e.g.*, bi-/multi-classification in the GLUE benchmark [69], the size  $|y|$  of label space is often small. A simple yet effective solution for discrete data is randomized response (RR) [70] proposed decades ago! Specifically, RR perturbs a true label  $y$  to itself  $\hat{y} = y$  with the probability

$$\Pr[y = \hat{y}] = e^\epsilon / (e^\epsilon + |y| - 1),$$

or to  $\forall \hat{y} \in y \setminus y$  uniformly, where  $y$  denotes the label space.

When  $|y|$  is large, we can use prior to “prune”  $y$  to smaller  $y'$  [31]. The prior can be publicly available (*e.g.*, auxiliary corpora similar to the users’ data) or progressively refined from a uniform distribution via the multi-stage training [31]. One can then estimate an optimal  $|y'|$  by maximizing the probability that the output is correct, *i.e.*,  $\Pr[y = \hat{y}]$ . With (prior-aided) RR [31], we can achieve full LDP.

**Theorem 2.** *Let  $f(\cdot)$  be the pre-noise function (of BERT-based pipelines),  $\mathcal{M}$  be GM with  $\epsilon_1 \geq 0, 0 \leq \delta \leq 1$ , and  $\mathcal{M}_{RR}$  be (prior-aided) RR with  $\epsilon_2 \geq 0$ . DP-Forward fine-tuning perturbing  $f(X)$  and  $y$  separately by  $\mathcal{M}$  and  $\mathcal{M}_{RR}$  ensures  $(\epsilon_1 + \epsilon_2, \delta)$ -LDP.*

The proof follows from the basic composition theorem [23].

**Privacy Amplification by Shuffling.** If noisy embedding-label pairs are also shuffled properly, DP-Forward can claim example-level CDP (as in DP-SGD), which “amplifies” LDP guarantees by  $\Theta(\sqrt{N})$  for a total number of  $N$  users (without extra noise addition) [25]. We then show that DP-Forward qualitatively outperforms DP-SGD from the SNR perspective under a similar privacy regime.

Suppose we train for an epoch, and the normalization factor is  $C$ . For DP-SGD, the batch size is  $b$ ; the subsampling probability and the

number of training steps are respectively  $b/N$  and  $N/b$ . If each step is  $(\epsilon, \delta)$ -DP, the overall privacy loss is  $(O(\epsilon\sqrt{b/N}), \delta)$ -DP using the strong composition and privacy amplification by subsampling [1].

DP-Forward with shuffling can also be seen as composing  $N$  subsamplings, each a fraction of size 1 [66]. It is  $(O(\epsilon\sqrt{1/N}), \delta)$ -DP, which is “amplified” from  $(\epsilon, \delta)$ -LDP. For an easier analysis of SNR, we omit  $\epsilon_2$  of RR since the overall  $\epsilon$  is dominated by composing subsampled Gaussian. So, our Gaussian noise variance is  $b \times$  smaller than DP-SGD’s in each step; the SNR of each entry in embeddings *vs.* the aggregation of  $b$  gradients can be estimated as  $O(C/\sqrt{nd})$  for DP-Forward *vs.*  $O(C/\sqrt{d'})$  for DP-SGD, where  $d'$  is the gradient dimension and is much larger than  $nd$ , the embedding-matrix size.

### 3.5 DP-Forward Inference

Given only fine-tuned pipeline parts  $f(\cdot)$ , users can derive the noisy embedding matrices of their *test* sequences for inferences at the service provider while ensuring  $(\epsilon, \delta)$ -LDP. Inference using noise aligned to the noisy fine-tuning is also beneficial for task accuracy.

Local inference (as in DP-SGD) without noise forces the service provider to reveal its *entire* pipeline, losing its intellectual property and incurring more time and storage costs for both  $f(\cdot)$  and  $p(\cdot)$ .

**Theorem 3.** *Let  $f(\cdot)$  be the fine-tuned pre-noise layers (of BERT-based pipelines) and  $\mathcal{M}$  be GM with  $\epsilon \geq 0, 0 \leq \delta \leq 1$ . DP-Forward inference running  $\mathcal{M}$  on normalized/clipped  $f(\cdot)$  ensures  $(\epsilon, \delta)$ -LDP.*

The proof is inherited from GM [23]. Different from DP-Forward fine-tuning, LDP holds for test sequences since the labels are absent.

### 3.6 DP-Forward Pre-training

Directly using the raw BERT might not “match” DP-Forward fine-tuning/inference, degrading task utility. Pre-training BERT with DP-Forward on publicly available text (*e.g.*, Wikipedia), besides the private user-shared data, can make future operations “adaptive” to noise. It requires us to modify the raw MLM objective in Eq. (1):

$$L_{\text{MLM}}^* = - \sum_{i \in I} \log \Pr[x_i | \mathcal{M}(f(\hat{X})); \theta^*],$$

where  $\theta^*$  denotes the parameters of “noisy” BERT. This endows the noisy BERT with some “de-noising” ability since the objective is to predict the raw masked tokens from noisy embeddings  $\mathcal{M}(f(\hat{X}))$ . It does not really breach privacy due to the free post-processing: LDP is ensured for each sequence, as the pre-training is self-supervised (without labels). Such noisy pre-training can also be outsourced to dedicated GPU clusters, enabling “de-noising BERT as a service.”

De-noising as post-processing is not new, but most prior arts need prior knowledge, *e.g.*, Bayesian prior. aGM formulates it as an unusual estimation problem since a single noisy output is observed for each input, which can then be solved by appropriate estimators, *e.g.*, the Bayesian one [7]. Another attempt [41] trains a separate noisy auto-encoder, which learns the identity function  $f(X) = X$  stacked before an image classification network, to de-noise the noisy input. It has limited applications for only noisy input embeddings and incurs extra changes when migrating it to an NLP pipeline.

## 4 OPTIMIZING MATRIX GAUSSIAN NOISE

To instantiate  $\mathcal{M}$  for  $f(\cdot) \in \mathbb{R}^{n \times d}$  of DP-Forward, a natural question is whether the classical GM is optimal. The answer is no. Its privacy analysis applies a *sufficient but not necessary* condition for  $(\epsilon, \delta)$ -DP while using Gaussian tail approximations, and its variance formula cannot extend to  $\epsilon > 1$  for a single run (e.g., inference) [23].

Another candidate is the matrix-variate Gaussian (MVG) mechanism [14], tailored for matrix-valued functions. It exploits possibly *non-i.i.d.* noise from a *matrix* Gaussian distribution and outperforms GM in several usage cases [14]. Yet, it is not optimal either, with the root cause still being based on a sufficient DP condition (Section 4.1). To improve it, we resort to a necessary and sufficient condition from aGM [7] for calibrating the matrix Gaussian noise (Section 4.2).

### 4.1 Matrix-Variate Gaussian (MVG) Mechanism

In contrast to the classical GM, MVG adopts possibly *non-i.i.d.* noise  $Z \in \mathbb{R}^{n \times d}$  drawn from the zero-mean matrix Gaussian distribution  $\mathcal{MN}_{n,d}(0, \Sigma, \Psi)$ , where  $\Sigma \in \mathbb{R}^{n \times n}$  and  $\Psi \in \mathbb{R}^{d \times d}$  are the row- and column-wise covariance matrices. Intuitively, it adds less noise to more “important” rows or columns for possible better utility.

**Definition 5 (Matrix Gaussian Distribution).** *The PDF for an  $n \times d$  random variable  $Z$  following  $\mathcal{MN}_{n,d}(0, \Sigma, \Psi)$  has the form:*

$$\Pr(Z|0, \Sigma, \Psi) = \frac{\exp\left(-\frac{1}{2}\|U^{-1}ZV^{-\top}\|_F^2\right)}{(2\pi)^{nd/2}|\Psi|^{d/2}|\Sigma|^{n/2}}, \quad (2)$$

where  $U \in \mathbb{R}^{n \times n}$  and  $V \in \mathbb{R}^{d \times d}$  are invertible with  $\Sigma = UU^\top$  and  $\Psi = VV^\top$ , and  $|\cdot|$  denotes the matrix determinant [34].

The definition is equivalent to the conventional form given by the matrix trace. It generalizes the univariate Gaussian used in GM;  $Z$  becomes i.i.d. when  $\Sigma, \Psi$  are diagonal and equal-valued. Below recites the main theorem of the MVG mechanism for  $(\epsilon, \delta)$ -DP.

**Theorem 4 (The MVG Mechanism with  $(\epsilon, \delta)$ -DP [14]).** *Let*

$$\sigma(\Sigma^{-1}) = [\sigma_1(\Sigma^{-1}), \dots, \sigma_n(\Sigma^{-1})]^\top,$$

$$\sigma(\Psi^{-1}) = [\sigma_1(\Psi^{-1}), \dots, \sigma_d(\Psi^{-1})]^\top$$

*be the vectors of (non-increasingly ordered) singular values of  $\Sigma^{-1}$  and  $\Psi^{-1}$ , respectively. The MVG mechanism using noise from the matrix Gaussian distribution  $\mathcal{MN}_{n,d}(0, \Sigma, \Psi)$  satisfies  $(\epsilon, \delta)$ -DP if*

$$\|\sigma(\Sigma^{-1})\|_2 \cdot \|\sigma(\Psi^{-1})\|_2 \leq \frac{(-\beta + \sqrt{\beta^2 + 8\alpha\epsilon})^2}{4\alpha^2},$$

where  $\alpha = [H_r + H_{r,1/2}]\gamma^2 + 2H_r\gamma S_2(f)$ ,  $\beta = 2(nd)^{1/4}H_r S_2(f)\zeta(\delta)$ , with  $H_r$  (or  $H_{r,1/2}$ ) being the generalized harmonic number of order  $r$  (of  $1/2$ ),  $\gamma$  being  $\sup_X \|f(X)\|_F$ , and  $\zeta(\delta) = 2\sqrt{-nd \ln \delta} - 2 \ln \delta + nd$ .

To illustrate how the MVG mechanism works, we quote an example [14]: performing regression using an identity query on a liver disorders dataset [49] with 6 features and 248 samples (i.e.,  $f \in \mathbb{R}^{248 \times 6}$ ). MVG treats ‘ALT’ and a teacher label as the two most indicative features based on some prior, thus added with less noise [14]. To report the best *empirical* results, it tries different precision budget (or noise variance) allocation strategies so that the total budget (Theorem 4) is not overspent. For example, it evenly allocates  $\tau > 50\%$  (a tunable parameter) of the budget to the two

---

### Algorithm 1: $A(\epsilon, \delta)$ : Derive the Upper Bound $\mathcal{B}$

---

**Input:** Privacy parameters  $\epsilon, \delta$

**Output:**  $\mathcal{B}$

```

1 Let  $\delta_0 = \Phi(0) - e^\epsilon \Phi(-\sqrt{2\epsilon})$ ;
2 if  $\delta \geq \delta_0$  then
3   Re-write  $g_\epsilon^+(v) = \Phi(\sqrt{\epsilon}v) - e^\epsilon \Phi(-\sqrt{\epsilon(v+2)})$ ;
4   Compute  $v^* = \sup\{v \in \mathbb{R}_{\geq 0} : g_\epsilon^+(v) = \delta\}$ ;
5   Let  $\alpha = \sqrt{1 + v^*/2} - \sqrt{v^*/2}$ ;
6 else
7    $g_\epsilon^-(u) = \Phi(-\sqrt{\epsilon}u) - e^\epsilon \Phi(-\sqrt{\epsilon(u+2)})$ ;
8   Compute  $u^* = \inf\{u \in \mathbb{R}_{\geq 0} : g_\epsilon^-(u) = \delta\}$ ;
9   Let  $\alpha = \sqrt{1 + u^*/2} + \sqrt{u^*/2}$ ;
10 end
11 Return  $\mathcal{B} = \sqrt{2\epsilon}/\alpha$ 

```

---

important features and the rest to the other four. Compared to GM using i.i.d. Gaussian noise, MVG can improve root mean square error (RMSE) by up to 0.003 at the same privacy level [14].

**Sub-optimality of MVG.** Theorem 4 presents an upper bound on the product of  $L_2$ -norms of two singular-value vectors  $\sigma(\Sigma^{-1})$  and  $\sigma(\Psi^{-1})$ , assuming  $\|f(X)\|_F$  is bounded for any  $X$  by a constant  $\gamma$ . The upper bound monotonically decreases with  $\beta$  that depends on  $nd$  and approaches 0 as  $nd \rightarrow \infty$ , making the sums of noise variances large. A similar situation exists in high privacy regimes  $\epsilon \rightarrow 0$ .

At least two slacks caused the sub-optimality. The first and foremost is due to a *sufficient condition* for  $(\epsilon, \delta)$ -DP [23]:  $\Pr[\mathcal{L}_{\mathcal{M}, X, X'} \geq \epsilon] \leq \delta$ , which is also used in the classical GM. With the Laurent-Massart Theorem [40], MVG further transforms it to  $\Pr[\mathcal{L}_{\mathcal{M}, X, X'} \leq \epsilon] = 1$  for a subset of all the possible outputs. The second lies in a loose matrix-trace-based privacy analysis; a follow-up [75] derives a tighter bound from Definition 5 and a matrix-norm inequality.

### 4.2 Analytic Matrix Gaussian Mechanism

To enhance MVG while still adding possibly *non-i.i.d.* noise  $Z \sim \mathcal{MN}_{n,d}(0, \Sigma, \Psi)$ , we put forth the analytic matrix Gaussian mechanism (aGM) by exploiting a *necessary and sufficient* condition for  $(\epsilon, \delta)$ -DP, which is formulated using two PLRVs by the analytic GM (aGM) [7]. It is non-trivial<sup>7</sup> since we now need to work with two covariance matrices  $\Sigma$  and  $\Psi$  instead of a single variance  $\sigma^2$  in aGM.

**Theorem 5 ([7]).** *A mechanism  $\mathcal{M}$  is  $(\epsilon, \delta)$ -DP iff,  $\forall X \approx X'$ ,*

$$\Pr[\mathcal{L}_{\mathcal{M}, X, X'} \geq \epsilon] - e^\epsilon \Pr[\mathcal{L}_{\mathcal{M}, X', X} \leq -\epsilon] \leq \delta. \quad (3)$$

It directly implies the sufficient condition due to  $\Pr[\mathcal{L}_{\mathcal{M}, X', X} \leq -\epsilon] \geq 0$ . We next show that  $\mathcal{L}_{\mathcal{M}, X, X'}$  or  $\mathcal{L}_{\mathcal{M}, X', X}$  of aGM is also Gaussian, a similar result has been proven in aGM [7, Lemma 3].

**Lemma 1.** *The PLRVs of our aGM follow a distribution  $\mathcal{N}(\eta, 2\eta)$  with  $\eta = \frac{\|U^{-1}\Delta V^{-\top}\|_F^2}{2}$ , where  $\Delta = f(X) - f(X')$ .*

<sup>7</sup>A recent pre-print [74] also studied using matrix Gaussian distribution. The proof of [74, Lemma 4], pivotal for our Theorem 6, is problematic. We prove it in Appendix C.



With Lemma 1, we can then specialize the left-hand side of Eq. (3). Particularly, we use the Gaussian cumulative density function (CDF)

$$\Phi(t) = \Pr[\mathcal{N}(0, 1) \leq t] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^t e^{-y^2/2} dy$$

to explicitly express the two probabilities (see Lemma 2) instead of approximating them by the tail bounds of a Gaussian distribution.

**Lemma 2.** *For any  $\mathcal{X} \simeq \mathcal{X}'$ , let  $\Delta' = U^{-1}\Delta V^{-\top}$  with  $\Delta = f(\mathcal{X}) - f(\mathcal{X}')$ . The following holds for any  $\epsilon \geq 0$ :*

$$\begin{aligned} \Pr[\mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'} \geq \epsilon] &= \Phi\left(\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right), \\ \Pr[\mathcal{L}_{\mathcal{M}, \mathcal{X}', \mathcal{X}} \leq -\epsilon] &= \Phi\left(-\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right). \end{aligned}$$

We can further re-write the left-hand side of Eq. (3) as  $g(\|\Delta'\|_F)$ :

$$\Phi\left(\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right) - e^{\epsilon} \Phi\left(-\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right), \quad (4)$$

a function of  $\Delta$  and  $(\Sigma, \Psi)$ ; it is defined w.r.t.  $\Delta$  and  $\sigma^2$  for aGM [7]. To satisfy Theorem 5, we require  $g(\|\Delta'\|_F) \leq \delta, \forall \mathcal{X} \simeq \mathcal{X}'$ . Since  $g(\cdot)$  is monotonically increasing [7, Lemma 7], we first find the upper bound  $\mathcal{B}$  of  $\|\Delta'\|_F$  as the “solution” to  $g(\|\Delta'\|_F) = \delta$  and then determine  $U, V$  (hence  $\Sigma, \Psi$ ) based on  $\mathcal{B}$  and  $\Delta$  with  $S_2(f) = \sup_{\mathcal{X} \simeq \mathcal{X}'} \|\Delta\|_F$ .

**4.2.1 Computing the upper bound  $\mathcal{B}$ .** One could derive an analytic expression for  $\mathcal{B}$  using the tail bounds of  $\Phi(t)$ , which is sub-optimal due to the slack in the tail bounds. Instead, we adapt a “numerical solver,” as detailed in Alg. 1, for  $\mathcal{B}$  since  $\Phi(t)$  can also be represented by  $(1 + \text{erf}(t/\sqrt{2}))/2$ , where  $\text{erf}$  is the standard error function.<sup>8</sup>

For the first term of Eq. (4), its input  $\|\Delta'\|_F/2 - \epsilon/\|\Delta'\|_F$  changes sign at  $\|\Delta'\|_F = \sqrt{2\epsilon}$ , while the other term’s input  $-\|\Delta'\|_F/2 - \epsilon/\|\Delta'\|_F$  is always negative. Therefore, we only consider  $\|\Delta'\|_F = \sqrt{2\epsilon}/\alpha$  under two cases  $0 < \alpha \leq 1$  and  $\alpha > 1$  for a variable  $\alpha$ .

When  $\alpha = 1$ ,  $\delta_0 = g(\sqrt{2\epsilon})$  in line 1. If  $\delta \geq \delta_0$  (or  $0 < \alpha \leq 1$ ), we can use  $v = (1/\alpha - \alpha)^2/2$  to re-write  $g(\cdot)$  as  $g_v^+(v)$  (line 3). For  $\alpha > 1$ , we can use  $u = (\alpha - 1/\alpha)^2/2$  to re-write  $g(\cdot)$  as  $g_v^-(u)$  (line 7). In either case, given the “oracle” computing  $\Phi(t)$  via  $\text{erf}$ , we derive  $u^*$  or  $v^*$  using Newton’s method, recover  $\alpha$ , and return  $\mathcal{B} = \sqrt{2\epsilon}/\alpha$ .

**4.2.2 Determining the covariance matrices  $\Sigma = UU^\top$  and  $\Psi = VV^\top$ .** With Lemma 6, and let  $\sigma_i(\cdot)$  be the  $i^{\text{th}}$  singular value; we have

$$\|U^{-1}\Delta V^{-\top}\|_F^2 \leq \sum_{i=1}^{\min\{n,d\}} \sigma_i^2(U^{-1})\sigma_i^2(\Delta)\sigma_i^2(V^{-\top}). \quad (5)$$

Since  $\sigma_i(U^{-1}) = 1/\sigma_{n-i+1}(U)$  and  $\sigma_i(V^{-\top}) = 1/\sigma_{d-i+1}(V)$  with  $i \in [1, r]$ , we transform the right-hand side of Eq. (5) to

$$\sum_{i=1}^r \frac{\sigma_i^2(\Delta)}{\sigma_{n-i+1}^2(U)\sigma_{d-i+1}^2(V)} \leq \frac{\sum_{i=1}^r \sigma_i^2(\Delta)}{\sigma_n^2(U)\sigma_d^2(V)} = \frac{\|\Delta\|_F^2}{\sigma_n^2(U)\sigma_d^2(V)},$$

where the inequality follows from Theorem 8, i.e.,  $\sigma_1(\cdot) \geq \dots \geq \sigma_r(\cdot)$ , and the last equality is directly from Lemma 5.

Given  $\mathcal{B} \geq \|U^{-1}\Delta V^{-\top}\|_F$ , it suffices to let  $\|\Delta\|_F/\sigma_n(U)\sigma_d(V) \leq \mathcal{B}$  with  $\Delta = f(\mathcal{X}) - f(\mathcal{X}'), \forall \mathcal{X} \simeq \mathcal{X}'$ . Recall that  $S_2(f)$  is the upper bound on  $\|\Delta\|_F, \forall \mathcal{X} \simeq \mathcal{X}'$ , we now reach the main theorem.

<sup>8</sup>Its efficient implementation to extremely high accuracy is supported in most statistical and numerical software packages, e.g., Python math library.

**Theorem 6.** *Our aMGM satisfies  $(\epsilon, \delta)$ -DP, iff*

$$\frac{S_2(f)}{\mathcal{B}} \leq \sigma_n(U)\sigma_d(V),$$

where  $\mathcal{B} = A(\epsilon, \delta)$  as in Alg. 1,  $S_2(f)$  is the  $L_2$ -sensitivity,  $\sigma_n(U)$  and  $\sigma_d(V)$  are respectively the smallest singular values of  $U$  and  $V$ .

Theorem 6 only constrains the lower bound on the product of  $\sigma_n(U)$  and  $\sigma_d(V)$ , the two *smallest* singular values; it offers infinite choices for all the others with the design space for  $(\Sigma, \Psi)$  even larger than that of MVG (Theorem 4). More importantly, the lower bound is independent of  $nd$ , which can lead to orders-of-magnitude variance reduction than MVG, confirmed by our experiments in Section 5. For  $\epsilon \rightarrow 0$ , we can still derive a valid  $\mathcal{B}$  from  $2\Phi^{-1}((1 + \delta)/2)$ .

To determine  $\Sigma, \Psi$ , another implicit constraint is to keep smaller noise for better utility. Let us first consider  $\Sigma = UU^\top$ . Since it is positive definite, we can also decompose it into  $W_\Sigma \Lambda_\Sigma W_\Sigma^\top$ ; we then have  $U = W_\Sigma \Lambda_\Sigma^{1/2}$ , where  $\Lambda_\Sigma^{1/2} = \{\sigma_i(U)\}_{i=1}^n$  specifies the row-wise noise magnitudes. Assuming that the smallest overall noise will yield the best utility, we let all the singular values be the smallest:  $\sigma_1(U) = \dots = \sigma_n(U)$ . As  $W_\Sigma$  can be any unitary matrix, we simply use the standard basis, resulting in  $U = \sigma_n(U) \cdot I_n$  for an  $n \times n$  identity matrix  $I_n$  and hence the final  $\Sigma$ . Similarly, we can pick  $\Psi = VV^\top$  with  $V = \sigma_d(V) \cdot I_d$ , where  $I_d$  is a  $d \times d$  identity matrix.

**4.2.3 Drawing the noise  $Z$ .** With  $\Sigma$  and  $\Psi$ , the last step is to draw  $Z$ . Pragmatically, we adopt the affine transformation below.

**Lemma 3 ([14]).** *Let  $Z' \in \mathbb{R}^{n \times d}$  be a random variable with each entry i.i.d. drawn from the standard normal distribution. The transformed  $Z = UZ'V^\top$  follows  $\mathcal{MN}_{n,d}(0, UU^\top, VV^\top)$ .*

Hence, we can first sample  $nd$  i.i.d. values from  $\mathcal{N}(0, 1)$  to form  $Z'$ , then employ the transformation  $UZ'V^\top$  such that

$$Z \sim \frac{S_2(f)}{\mathcal{B}} \mathcal{MN}_{n,d}(0, I_n, I_d). \quad (6)$$

**Discussion.** When instantiating DP-Forward using aMGM, we set  $\sigma_1(U) = \dots = \sigma_n(U)$  and  $\sigma_1(V) = \dots = \sigma_d(V)$  such that the row- and column-wise noises are the smallest, and our pilot experiments show this yields optimal task utility; aMGM actually “degenerates” to aGM with i.i.d. noise. Nevertheless, aMGM also allows non-i.i.d. noise like MVG: By tuning the corresponding singular values larger, we can add more noise to the rows/columns that negatively impact the utility. It might be helpful when  $f(\cdot)$  (e.g., linear regression on a small liver dataset [14]) is simple or  $p(\cdot)$  does not “mix up” noisy rows/columns. In contrast to our empirical approach (like MVG), one could theoretically formulate the allocation of singular values as optimization problems that maximize different utility functions tailored to applications. It might outperform our uniform treatment but takes more dedicated efforts, which we leave as future work.

## 5 EXPERIMENTS

### 5.1 Experimental Setup

We use three typical datasets/tasks that are widely used in NLP/DP literature [45, 78–80] and GLUE benchmark [69]: i) Stanford sentiment treebank (SST-2), ii) Internet movie database (IMDb) [47] for binary sentiment classification of single- and multi-sentence movie



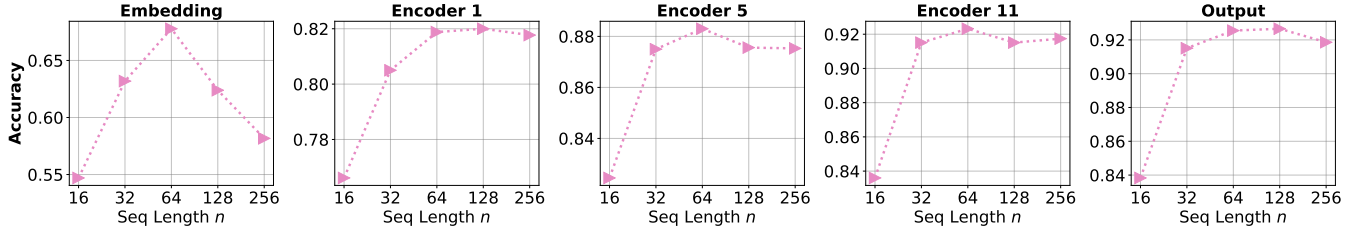
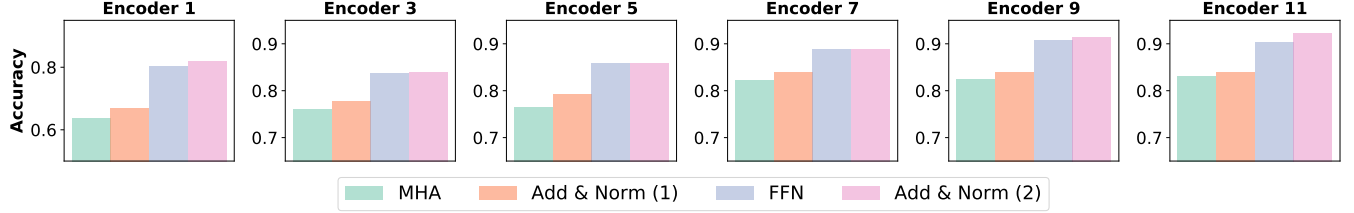
Figure 2: SST-2 accuracy when tuning  $n$  with noise added at different positions ( $\epsilon = 16$  for SeqLDP)Figure 3: SST-2 accuracy with noise added to the outputs of different *sub-layers* in six encoders ( $\epsilon = 16$  for SeqLDP)

Table 2: Statistics of the downstream task datasets

	SST-2 [69]	IMDb [47]	QQP [69]
#train samples	67,349	25,000	363,846
#test samples	872	25,000	40,320

reviews, and iii) Quora question pairs (QQP) for semantic equivalence test over question pairs on Quora.com. Their test sets do not have any labels; we use the original dev sets as the test sets. Table 2 summarizes their characteristics. They all carry privacy risks; e.g., stylistic features of posts may leak the author’s identity. We use task *accuracy* (w.r.t. the ground truth labels) as the utility metric.

**Baselines.** We instantiate  $\mathcal{M}$  in DP-Forward by the classical GM, MVG [14], and aMGM. If not specified, all the results are based on aMGM. For MVG, we adopt its *unimodal* type, applicable to asymmetric functions like pre-noise layers  $f(\cdot)$ . Specifically, we make the row-wise noise directional and assign the same precision budget to each row, assuming that tokens share the same importance.

By default, we report the accuracy of DP-Forward inferences on tasks fine-tuned using DP-Forward (with  $\sim 2$ pp gains compared to the case of “DP-Forward fine-tuning + non-private inference”). We also realize DP-SGD fine-tuning with the latest Opacus [77] but do not add any noise to its inference. Another baseline is non-private (in both fine-tuning and inference).

**Implementation.** We run experiments on a cluster with Tesla P100 GPUs. We implement all the mechanisms and baselines in Python. We use a raw BERT checkpoint bert-base-uncased [27], available in the Huggingface transformers library, for fine-tuning (Section 3.3) or further pre-train it over WikiCorpus (Section 3.6).

For the hyperparameters used throughout our experiments, we set the number of training epochs  $k = 3$ , learning rate  $\eta = 2e-5$ , batch size  $b = 32$ , and normalization/clipping factor  $C = 1$ . We keep others (e.g., no weight decay, no learning rate decay) default as literature [69]. The privacy parameter  $\delta$  is fixed as  $1e-5$  [78].

Table 3: SST-2 accuracy with different  $d$  and  $\epsilon$  for SeqLDP

$d \backslash \epsilon$	2	4	8	12	16
16	0.6801	0.7851	0.8833	<b>0.9232</b>	0.9266
64	0.6766	0.7752	0.8727	0.9037	0.9209
128	0.6732	<b>0.7856</b>	0.8807	0.9128	0.9232
256	<b>0.6835</b>	0.7695	<b>0.8965</b>	0.9186	<b>0.9249</b>
512	0.6411	0.7626	0.8831	0.9128	0.9243
768	0.6686	0.7741	0.8739	0.9128	0.9186

## 5.2 Configuring Matrix Dimensions

The sequence length  $n$  is variable. While the hidden dimensionality  $d$  is tied as 768 for BERT-Base, we can resort to two linear maps for “mediating” it (see Section 3.2). Since we normalize embedding matrices of size  $n \times d$  to have a fixed norm  $C$ , each entry’s signal magnitude relies on  $(n, d)$ . In contrast, the noise variance is the same given  $C$  and fixed privacy parameters. The signal-to-noise ratios (SNRs) affecting accuracy can be configured based on  $(n, d)$ .

Figure 2 shows the evaluation accuracy of SST-2 fine-tuned using DP-Forward with  $n$  tuning from 16 to 256. We study adding aMGM noise at five hidden layers’ outputs. The results indicate that the best accuracy is often achieved at  $n = 64$  or  $128$ , so we opt for  $n = 128$  (which is sufficient for most sequences) in subsequent experiments.

We fine-tuned SST-2 on noisy output embeddings under different choices of  $\epsilon$  and reduced  $d$ . Table 3 summarizes the results. Reducing  $d$  leads to larger SNRs (under fixed  $C$  and  $n$ ) but may also lose useful information, degrading accuracy. For the same  $\epsilon$ , most accuracy variations are within 2pp under different choices of  $d$ . Balancing everything, we use the raw  $d = 768$  in later experiments such that no extra changes (including two linear maps) are made to pipelines.

## 5.3 Fine-tuning with Sequence LDP

Our approach also supports perturbing *sub-layer* outputs during fine-tuning. We study six encoders as an example, with the results

**Table 4: Accuracy on output embeddings under SeqLDP**

Task	Mech.	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$
SST-2	GM	0.5424	0.5757	0.6537	0.7466	0.8624
	aMGM	<b>0.5516</b>	<b>0.6021</b>	<b>0.6686</b>	<b>0.7741</b>	<b>0.8739</b>
	MVG	~0.5				
IMDb	GM	0.5244	0.5498	0.6016	0.6902	0.8002
	aMGM	<b>0.5353</b>	<b>0.5676</b>	<b>0.6224</b>	<b>0.7093</b>	<b>0.8109</b>
	MVG	~0.5				
QQP	GM	0.6304	0.6321	0.6571	0.7458	0.8638
	aMGM	<b>0.6312</b>	<b>0.6348</b>	<b>0.6747</b>	<b>0.7685</b>	<b>0.8653</b>
	MVG	~0.5				

shown in Figure 3. Overall, DP-Forward performs better with deeper encoders since fewer parameters are directly affected by noise during fine-tuning. Another observation is that perturbing different sub-layer outputs, even inside the same encoder, may result in huge accuracy variation; *e.g.*, using noisy outputs of the last sub-layer in Encoder 1 can bring ~20pp gains over those of the first sub-layer.

We next evaluate the privacy-accuracy tradeoffs under different  $\epsilon$  and compare the instantiations using the classical GM, MVG [14], and aMGM. Note that we still compute the GM variance as  $\sigma^2 = 2 \ln(1.25/\delta) S_2^2(f)/\epsilon^2$  for empirical evaluation, albeit it cannot extend to  $\epsilon > 1$  for a single run to ensure theoretical DP guarantees.

For the GM- and aMGM-based instantiations, Table 4 shows all three tasks' accuracy increases with  $\epsilon$ . Ours has better accuracy than the GM-based one due to the smaller noise produced by aMGM in all choices of  $\epsilon$ . Although the noise variance gap (between GM and aMGM) widens as  $\epsilon$  decreases, one cannot fine-tune effective models in a high privacy regime  $\epsilon < 1$ . The MVG-based one behaves like random guessing for all three tasks since its noise variance is proportional to  $n \cdot d$ , which is even much larger than the classical GM for high-dimensional settings (see Section 4.1). For instance, under the same parameter setting (*e.g.*,  $n = 128$ ,  $d = 768$ , and  $\epsilon = 8$ ), MVG produces noise with the variance orders-of-magnitude larger than aMGM (*e.g.*,  $> 10^8$  vs.  $\sim 0.6$ ), even assuming  $\sup \|f(\cdot)\|_F = 1$ .

We remark that the used local  $\epsilon$  value is not large. Most classical LDP works that deem such  $\epsilon$  lies in a low privacy regime are for statistical analytics. In great contrast, we aim at fine-tuning large LM-based pipelines with high-dimensional signals and limited training data, which is much more complicated. Many prior works [28, 29, 58, 80] use a larger  $\epsilon$  to ensure even a weaker token-level LDP variant, while others [51] categorizes  $\epsilon < 10$  and  $10 \leq \epsilon < 20$  as strong and moderate privacy respectively<sup>9</sup> for sequence-level LDP like ours. More importantly, they provide effective protection against various privacy threats, as detailed in Section 6.

#### 5.4 DP-Forward versus DP-SGD

*Fairness of comparisons on privacy-accuracy tradeoffs.* As elaborated in Section 3.4, we can adopt RR [70] to perturb the labels and then report central  $\epsilon$  values for DP-Forward, amplified by shuffling using the following parameters, ensuring that comparisons are fair under

<sup>9</sup>Such choices can be “reduced” to smaller ones under the shuffling model (Section 5.4), *cf.* U.S. census discloses demographic data at central  $\epsilon = 11.14$  [3].

(example-level) CDP. For DP-SGD, the subsampling probability is  $b/N$ , with  $b = 32$  and the dataset size  $N$ ; the number of fine-tuning steps is  $T = k \cdot N/b$  with  $k = 3$ . For DP-Forward, the subsampling and non-flipping probabilities are respectively  $1/N$  (with  $T = k \cdot N$ ) and 0.9; we still process  $b$  noisy embeddings as a batch. For both, we use aGM [7], the degenerated version of aMGM (Section 4.2), and the same accountant [32] to report *approximated* overall  $\epsilon$  values<sup>10</sup>.

We study eight instances of DP-Forward, including perturbing the outputs of the input embedding layer, six different encoders, and BERT. Their accuracies on all three tasks under three privacy levels, plus those of DP-SGD and the non-private baseline, are shown in Table 5. About half or more of our instances have better accuracy than DP-SGD for each task; the largest accuracy gain is ~7.7pp for QQP. The noisy output embeddings often lead to the best accuracy for all tasks, even comparable to the non-private baseline, due to the dimension reduction at the last encoder output (Section 2.1).

Recent DP-SGD variants [78, 79] improve DP-SGD [1] by perturbing *partial gradient* entries using additional tricks (*e.g.*, low-rank adaption). They report the best accuracy of 92.5% and 85.7% on SST-2 and QQP, respectively, with 2.3pp and 6.2pp drops from the non-private baselines at central  $\epsilon = 6.7$  [78, Table 4]. DP-Forward with label privacy, incurring  $< 1.7$ pp accuracy drops on the two tasks at  $\epsilon \approx 3$ , can still beat them, albeit their fine-tuning is based on RoBERTa-base, a robustly optimized BERT approach, which by itself outperforms BERT due to larger training set, longer training time, and better techniques (*e.g.*, dynamic masking in MLM).

Figure 4 shows the efficiency comparisons on fine-tuning SST-2. The time and storage overheads of our approach (for all possible instances) are almost the same as the non-private baseline and ~3× smaller than DP-SGD. It is because we allow batch processing as in the normal fine-tuning – no need to handle per-example gradients. Meanwhile, our normalization and noise sampling/addition are also faster since the size of embeddings is smaller than that of gradients.

#### 5.5 Noisy Pre-training

Pre-training BERT using DP-Forward, aligned with the noisy fine-tuning, does help accuracy. We use SST-2 as an example and perturb the input embedding matrices. We continue pre-training BERT over English WikiCorpus, the 2006 dump with about 600M words, for an epoch. Table 6 shows that we can obtain 1–2pp accuracy gains for most choices of  $\epsilon$ , compared to fine-tuning on the original BERT.

Efficiency-wise, DP-Forward pre-training also consumes much fewer resources; *e.g.*, an existing work [5] pre-trains BERT-Large (with 340 million parameters) using DP-SGD on Google TPUs, which requires sufficient memory for handling batch sizes of millions.

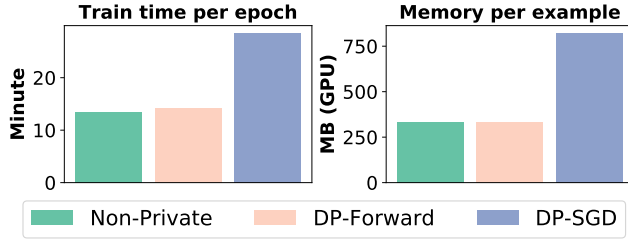
### 6 DEFENSE AGAINST PRIVACY THREATS

Following the recent taxonomy [64], we study MIAs and two new threats of sequence leakage from their *embeddings*: embedding inversion and attribute inference. We moderately adapt them to suit our context, *e.g.*, upgrading MIAs [64] to sequence-level.

<sup>10</sup>They are dominated by composing subsampled Gaussian, *e.g.*, composing subsampled RR only consumes 0.03 for SST-2, which is even overestimated by AutoDP.

**Table 5: Accuracy of task models fine-tuned using DP-Forward and DP-SGD under (example-level) CDP**

Method	Noise position	SST2			IMDb			QQP		
		$\epsilon \approx 1$	$\epsilon \approx 3$	$\epsilon \approx 8$	$\epsilon \approx 1$	$\epsilon \approx 3$	$\epsilon \approx 8$	$\epsilon \approx 1$	$\epsilon \approx 3$	$\epsilon \approx 8$
DP-Forward	Embedding	0.6055	0.6146	0.6278	0.5	0.5	0.5	0.6534	0.6589	0.6594
	Encoder 1	0.7971	0.8096	0.8073	0.5000	0.5016	0.5022	0.7857	0.7885	0.7906
	Encoder 3	0.8096	0.8394	0.8463	0.7525	0.7545	0.7549	<b>0.8513</b>	<b>0.8585</b>	<b>0.8607</b>
	Encoder 5	0.8544	0.8658	0.8716	0.7642	0.7709	0.7719	<b>0.8698</b>	<b>0.8765</b>	<b>0.8806</b>
	Encoder 7	0.8624	<b>0.8819</b>	<b>0.8872</b>	0.7765	<b>0.7883</b>	<b>0.7924</b>	<b>0.8840</b>	<b>0.8887</b>	<b>0.8926</b>
	Encoder 9	<b>0.8945</b>	<b>0.8979</b>	<b>0.9002</b>	<b>0.7995</b>	<b>0.8105</b>	<b>0.8181</b>	<b>0.8895</b>	<b>0.8941</b>	<b>0.8955</b>
	Encoder 11	<b>0.8819</b>	<b>0.8968</b>	<b>0.8985</b>	<b>0.8042</b>	<b>0.8187</b>	<b>0.8265</b>	<b>0.8952</b>	<b>0.8997</b>	<b>0.9007</b>
	Output	<b>0.8865</b>	<b>0.9009</b>	<b>0.9055</b>	<b>0.8096</b>	<b>0.8160</b>	<b>0.8270</b>	<b>0.8987</b>	<b>0.8994</b>	<b>0.9038</b>
DP-SGD	Gradient	0.8650	0.8713	0.8759	0.7779	0.7826	0.7903	0.8219	0.8345	0.8433
Non-private baseline		0.9178			0.8378			0.9019		

**Figure 4: Efficiency comparison for the case of SST-2****Table 6: SST-2 accuracy gain with pre-training under SeqLDP**

$\epsilon$	Raw BERT	Noisy BERT	$\Delta_{\text{acc}}$
2	0.5501	0.5665	0.0164
4	0.5950	0.5999	0.0049
8	0.6550	0.6708	0.0158
16	0.7345	0.7450	0.0105

## 6.1 Threat Models

For MIAs, we follow prior arts [63, 64, 76] to consider an adversary with only *black-box* access to an *entire* (DP-SGD/DP-Forward-trained) pipeline: It can query the prediction results (e.g., each-class probability) of target sequences but cannot access the pipeline weights and architecture; the hidden embeddings are not revealed.

Despite “different” objectives of inverting or inferring (Section 6.3 or 6.4) from embeddings, we consider both threats involving a general adversary with *black-box* access to the trained pipeline part  $f(\cdot)$ . It can get the inference-time (clear/noisy) *embeddings* of target sequences [64]. Besides public prior knowledge, it can collect a *limited auxiliary* dataset  $\mathcal{X}_{\text{aux}}$ , sharing similar attributes to the targets [64].

DP-SGD only offers CDP for training data and does not protect inference-time input.<sup>11</sup> What follows intends to empirically confirm a major merit of DP-Forward in protecting against stronger adversaries and threats to both training- and inference-time inputs.

<sup>11</sup>One might add the same noise to it as DP-Forward inference, which indeed mitigates the new threats. However, perturbing *gradients* in training, inherently “mismatches” from perturbing *embeddings* in inference, deteriorating task performance significantly, e.g., SST-2 accuracy will be reduced to 0.7786 (with a  $\sim 10\text{pp}$  drop) at central  $\epsilon \approx 8$ .

## 6.2 Membership Inference Attacks

**Attack Objective.** MIAs predict whether a data point is in the training set [63]. They often exploit the disparity in model behavior between training data and unseen data, *i.e.*, poor model generalization due to overfitting [76]. Inferring membership at the token/word level, e.g., a sliding window of tokens [64], is not interesting. We consider more realistic MIAs on *entire* sequences, which can be extended for more devastating attacks, such as extracting verbatim pre-training sequences via black-box access to GPT-2 [13].

Prior arts [65, 76] suggest that threshold-based MIAs using only prediction confidence [76] or entropy [65] with proper assumptions are comparable to the more sophisticated one [63] based on shadow training. Adapting the confidence-based MIA to our context exploits that a pipeline is fine-tuned by minimizing its prediction loss: The confidence/probability of predicting a training sequence as its true label should be close to 1. The adversary can then infer a candidate sequence  $X^*$  as a member when the confidence for the predicted label  $l$  output by pipeline  $\mathcal{F}$  is larger than a pre-set threshold  $\tau$ :

$$\mathbb{1}\{\Pr[\mathcal{F}(X^*) = l] \geq \tau\},$$

where  $\mathbb{1}\{\cdot\}$  is the indicator function. We simply use a fixed  $\tau$  for all possible labels in our evaluation, albeit it can be label-dependent.

The second MIA we use is based on the prediction output (*i.e.*, a vector of probabilities) of a training sequence tends to be a one-hot vector, *i.e.*, its entropy should be close to 0. Similarly, the adversary can infer  $X^*$  as a member when its prediction entropy falls below a preset threshold  $\tau$ ; otherwise, it is not deemed a member:

$$\mathbb{1}\{-\sum_i \Pr[\mathcal{F}(X^*) = l_i] \log(\Pr[\mathcal{F}(X^*) = l_i]) \leq \tau\},$$

for all possible labels  $\{l_i\}$ . Note that a totally wrong prediction with probability  $\sim 1$  also leads to entropy approaching 0. We can address it by encoding the information of the ground-truth label of  $X^*$  [65].

**Numerical Results.** As in [79], all the test examples and a random subset of the training examples (as many as the test ones) are evenly split into two subsets (each has half of the training/test examples), one for finding the optimal  $\tau$ , and the other for reporting the attack success rates. Given that the training and test examples likely share the same distribution, we randomly drop/replace tokens in the test examples to enlarge the prediction difference to make MIAs easier.

**Table 7: Success rates of two MIAs with (translated) central  $\epsilon$** 

$\epsilon$	Method	Attack Success Rate	
		Entropy	Confidence
$\infty$	Non-private baseline	0.659	0.645
1	DP-SGD	0.567	0.561
	DP-Forward (Embedding)	0.586	0.576
	DP-Forward (Encoder 1)	0.535	0.537
	DP-Forward (Encoder 7)	<b>0.494</b>	<b>0.506</b>
	DP-Forward (Encoder 11)	<b>0.506</b>	<b>0.494</b>
	DP-Forward (Output)	<b>0.508</b>	<b>0.502</b>
3	DP-SGD	0.584	0.576
	DP-Forward (Embedding)	0.584	0.576
	DP-Forward (Encoder 1)	0.543	0.530
	DP-Forward (Encoder 7)	<b>0.510</b>	<b>0.507</b>
	DP-Forward (Encoder 11)	0.512	<b>0.500</b>
	DP-Forward (Output)	<b>0.503</b>	<b>0.499</b>
8	DP-SGD	0.580	0.580
	DP-Forward (Embedding)	0.597	0.576
	DP-Forward (Encoder 1)	<b>0.510</b>	0.536
	DP-Forward (Encoder 7)	<b>0.510</b>	<b>0.504</b>
	DP-Forward (Encoder 11)	0.520	<b>0.513</b>
	DP-Forward (Output)	<b>0.506</b>	<b>0.490</b>

We evaluated the adapted confidence- and entropy-based MIAs on SST-2 fine-tuned by the non-private baseline, DP-Forward, and DP-SGD. For DP-Forward, we investigate five instances, perturbing input embeddings, three encoders' outputs, and output embeddings. Table 7 presents the results, where success rates within 0.49–0.51 are shown in bold. Both DP-Forward and DP-SGD can mitigate MIAs effectively. For all choices of  $\epsilon$ , the two MIAs' success rates on DP-Forward are reduced to  $\sim 0.5$  (like random guessing) for deeper layers, outperforming DP-SGD by  $>6$ pp at the same privacy level.

### 6.3 Embedding Inversion Attacks

**Attack Objective.** These attacks aim at recovering the raw text as (unordered) tokens  $\{x_i\}_{i \in [n]} \subseteq X$  from embeddings, highlighting the risk of directly sharing (without noise) even only text embeddings (for training/inference). They have been employed to reconstruct specific patterns, *e.g.*, identity codes and gene segments [56].

We first propose a simple *token-wise* inversion attack to invert (noisy) token embeddings output by the input embedding layer  $\phi(\cdot)$  that maps every token in  $\mathcal{V}$  to  $\mathbb{R}^d$  [73]. It can be formulated as:

$$\forall i \in [n] : \min_{x_i^* \in \mathcal{V}} \|\phi(x_i^*) - (\phi(x_i) + z_i)\|_2,$$

where  $z_i$  is the  $i^{\text{th}}$  row of noise  $Z$  from  $\mathcal{M}$  (omitted for DP-SGD or the non-private baseline). It returns  $x_i^*$  with its embedding closest to the observed one of  $x_i$  via a nearest-neighbor search over  $\mathcal{V}$ .

A token's hidden embedding from deeper layers encodes more "abstract" contextual information of the entire sequence it belongs to; the token-wise inversion may be less accurate. We thus require a more general attack [64]. It first maps the observed (noisy) embedding back to a lower-layer one using a linear least square model  $M$  and then selects  $n$  tokens as  $X^*$  to minimize the  $L_2$ -distance between

**Table 8: Success rates of two inversion attacks on (the lowest-layer) input embeddings with (translated) central  $\epsilon \approx 8$** 

	Nearest Neighbor			Gradient-based		
	SST-2	IMDb	QQP	SST-2	IMDb	QQP
Non-private	1	1	1	1	1	1
DP-SGD	1	1	1	.9991	.9982	1
DP-Forward	<b>.1811</b>	<b>.1420</b>	<b>.2457</b>	<b>.1622</b>	<b>.1241</b>	<b>.2226</b>

**Table 9: Success rates of a (neural-network-based) sensitive attribute inference attack with (translated) central  $\epsilon \approx 8$** 

	action	comedy	drama	horror	Overall
Non-private	0.727	0.858	0.516	0.439	0.687
DP-SGD	0.664	0.733	0.253	0.324	0.536
DP-Forward (Embedding)	0.998	0	0	<b>0.009</b>	<b>0.276</b>
DP-Forward (Encoder 1)	1.0	0	0	0	<b>0.276</b>
DP-Forward (Encoder 7)	1.0	0	0	0	<b>0.276</b>
DP-Forward (Encoder 11)	1.0	0	0	0	<b>0.276</b>
DP-Forward (Output)	0.998	0	0	<b>0.009</b>	<b>0.276</b>

the lower-layer representation of  $X^*$  and the one from  $M$ :

$$\min_{X^* \in \mathcal{V}^n} \|\zeta(X^*) - M(f(X) + Z)\|_2,$$

where  $\zeta(\cdot)$  is a lower-layer representation function than  $f(\cdot)$ .

The above minimization is over  $|\mathcal{V}|^n$ , larger than the token-wise candidate space. To determine  $X^*$ , we first relax the token selection at each position  $i \in [n]$  using a continuous vector in  $\mathbb{R}^{|\mathcal{V}|}$ , which is then input (with another temperature parameter) to a softmax function to model the probabilities of selecting each token in  $\mathcal{V}$ . We further derive the token embedding  $x_i^*$  by multiplying the relaxed vector (with each entry as a weight) and the original embedding matrix. Finally, we solve it by a gradient-based method [64].

**Numerical Results.** The gradient-based attack reports the highest recall (or precision) on inverting the lowest-layer (clear) embeddings [64, Figure 2]. To show that DP-Forward can mitigate such "strongest" inversion, we implement both (nearest-neighbor and gradient-based) attacks to invert input embeddings, with the public BERT embedding lookup table as prior. We also report their success rates as *recall* – the ratios of correct recoveries over the raw targets.

Table 8 shows that DP-Forward can reduce their success rates to a relatively low level, most are within 0.2. However, DP-SGD fails in defense. The results corroborate our claim: DP-Forward directly adds noise to embeddings, thus mitigating embedding inversion, whereas DP-SGD only perturbs gradients, offering no protection for the (clear) inference-time embeddings of test sequences.

### 6.4 Sensitive Attribute Inference Attacks

**Attack Objective.** Instead of recovering exact tokens, one can try to infer sensitive attributes about target sequences from their embeddings. The attributes are often statistically unrelated to the training/inference objective but inherent in sequences, *e.g.*, stylometry, implying the text's authorship for sentiment analysis [62]. We are not interested in any global property of an entire corpus [30].

We assume  $\mathcal{X}_{\text{aux}}$  has sequences labeled with sensitive attributes. The adversary can query  $f(\cdot)$  for the noisy (or clear) embeddings:

$$\mathcal{X}_{\text{aux}} = \{(f(X_i) + Z_i, s_i)\}, \forall s_i \in \mathcal{S},$$

where  $\mathcal{S}$  is the set of all possible sensitive attributes of interest, say, authorship. It does not care about non-sensitive attributes.

To infer sensitive attributes, the adversary first trains a classifier on  $\mathcal{X}_{\text{aux}}$  via supervised learning and then uses it for an observed noisy (or clear) embedding  $f(X^*) + Z$  to predict  $s^* \in \mathcal{S}$  of  $X^*$ .

**Numerical Results.** We train a three-layer neural network with a linear head as the classifier to infer the film genre (e.g., ‘horror’) as a sensitive attribute from a movie review using its output embedding. We employ IMDb with 20k examples (90% for training and 10% for validation) as  $\mathcal{X}_{\text{aux}}$ , and SST-2 contributes 3.3k examples for testing the classifier. The attack success rates are measured using *recall*.

We investigate five DP-Forward instances. Table 9 shows that they “reduce” the classifier to majority-class prediction, which returns the majority class (‘action’) on all inputs. In contrast, DP-SGD only reduces success rates moderately compared to the non-private baseline. It is because the embeddings from DP-SGD-trained/noisy models still “lose” some useful information (cf., accuracy drops of DP-SGD inference on embeddings without noise). The results confirm DP-Forward is more effective in thwarting attribute inference.

## 7 RELATED WORK

### 7.1 Privacy Threats on LMs and Embeddings

An active line of research [8, 13, 56, 64] discloses severe privacy risks in modern LMs (even used as black-box query “oracles”) concerning their (hidden/output) text embeddings. Song and Raghunathan [64] build a taxonomy of attacks that covers a broader scope than a parallel work [56]. These attacks include embedding inversion (which can partially recover raw texts), membership inference (establishing the *is-in* relation between a target and private training data), and inferring sensitive attributes like text authorship from embeddings. A common defense for them is adversarial training, e.g., [24].

Others [8, 13] study the “memorization” of training data in LMs (a.k.a. membership inference attack). In particular, Carlini *et al.* [13] define  $k$ -eidetic memorization, where a string is extractable or memorized if it appears in at most  $k$  examples. Their black-box attacks on GPT-2 [59] can extract verbatim training texts even when  $k = 1$  (e.g., a name that only appears once is still extractable). A smaller  $k$  means a higher privacy risk. Beguelin *et al.* [8] define differential score and rank as two new metrics for analyzing the update leakage, enabling the recovery of new text used to update LMs. Incorporating DP to address memorization is a promising solution.

### 7.2 Input (Text/Feature) Perturbation for LDP

SynTF [71] synthesizes term-frequency (feature) vectors under LDP, which have limited applications compared to sentence embeddings or text itself. Feyisetan *et al.* [28, 29] resort to metric-LDP [4], a relaxed variant of LDP with a distance metric (e.g., Euclidean or Hyperbolic), which allows the indistinguishability of outputs to grow proportionally to the inputs’ distance. They first add noise to the outputs of a non-contextualized token embedding model (e.g., GLoVe [57]), which are then projected back to “sanitized” text using the nearest neighbor search as post-processing. In contrast,

Yue *et al.* [80] sanitize text by directly sampling token-wise replacements, avoiding adding noise to high-dimensional embeddings. All these works only achieve (variants of) token-level metric-LDP.

To offer *sequence-level* protection, recent studies [46, 51] apply Laplace or exponential mechanism to perturb (the average of) sentence embeddings extracted by an LM (e.g., BERT [20]). Both ensure pure LDP (homogeneously protecting any entire sequence), which may be too stringent and impact utility. In contrast, heterogeneous protection [28, 80] can strategically manage the privacy demands across inputs. Du *et al.* [21] achieve metric-LDP (by Purkayastha and planar Laplace mechanisms) at the sequence level (unlike token-level in prior arts [28, 80]). To further boost the utility, they mitigate the dimensional curse via a random-projection-like approach. They also perturb sensitive sequence labels for enhanced privacy. Nevertheless, perturbing different hidden (rather than token or sentence) embeddings inside LM-based NLP pipelines remains unexplored.

### 7.3 DP-SGD (Variants) in Training LMs

An early attempt [50] uses DP-SGD to train long short-term memory LMs in the federated learning setting. By configuring hyperparameters properly (e.g., setting the batch size to millions), one can even pre-train BERT-Large, an LM with ~340M parameters, using DP-SGD/Adam while achieving acceptable (MLM) accuracy [5].

Using the vanilla DP-SGD in pre-training/fine-tuning large LMs leads to significant efficiency and accuracy drops due to the “curse of dimensionality.” Yu *et al.* [79] propose reparameterized gradient perturbation: It first reparameterizes/decomposes *each* high-rank weight matrix into two low-rank (gradient-carrier) ones with a residual matrix and then only perturbs the two low-rank gradients to alleviate the dimensional curse. The noisy low-rank gradients are finally projected back to update the raw high-rank weights.

Applying reparameterization to every weight in each update is still costly and may introduce instability (e.g., noises are “zoomed up” during the projection). Instead, the follow-up [78] builds atop the recent success of parameter-efficient fine-tuning (e.g., LoRA [36], Adapter [35], and Compacter [48]): It perturbs the gradients of a much smaller number of additional “plug-in” parameters. However, Li *et al.* [45] empirically show that parameter-efficient fine-tuning is not necessarily better than the full one; they propose ghost clipping, a memory-saving technique (“orthogonal” to dimension reduction), to use DP-SGD in full fine-tuning without instantiating per-example gradients. Despite efficiency/accuracy gains, all these works still only protect training data by perturbing (smaller) gradients.

### 7.4 DP Mechanisms for Matrix Functions

Gaussian and Laplace mechanisms are typically for scalar-/vector-valued functions [23]. Vectorizing the outputs and adding i.i.d. noise could generalize them for matrix-valued functions, but the structural information of matrix functions is not exploited. The MVG mechanism [14] is thus devised, which draws directional or non-i.i.d. noise from a *matrix* Gaussian distribution. It injects less noise into more “informative” output directions for better utility, with only a constraint on the sum of the singular values (determining the noise magnitude) of two covariance matrices. Such a constraint is only a sufficient condition for  $(\epsilon, \delta)$ -DP, which is improved by the follow-up [75] with a tighter bound on the singular values.

There also exist mechanisms dedicated to *restricted* matrix-valued functions. The matrix mechanism [43] considers a collection of linear counting queries represented by  $Wx$  for query matrix  $W$  and input vector  $x$ . It still resorts to additive Laplace/Gaussian noise but with an extra transformation solving the min-variance estimation to the noisy  $Wx$ . Another very recent study [37] focuses on matrix-valued queries with only binary (matrix) outputs. It then devises an exclusive-or (xor) mechanism xor-ing the outputs with noise attributed to a matrix-valued Bernoulli distribution.

## 8 CONCLUSION

Pre-trained LMs became pivotal in NLP. Alarming, fine-tuning corpora or inference-time inputs face various privacy attacks. The popular DP-SGD only provides limited protection for training data by adding noise to gradients. Raw tokens or sensitive attributes of training/inference data can be inverted or inferred from embeddings in forward-pass computation. Vanilla DP-SGD also imposes high GPU memory and computational burdens but cannot be batched.

We propose DP-Forward, which directly adds noise to embedding matrices derived from the raw training/inference data in the forward pass. Its core is the analytic matrix Gaussian mechanism, a general-purpose tool that owns independent interests. It draws optimal matrix-valued noise from a matrix Gaussian distribution in a dedicated way using a necessary and sufficient condition for DP.

Perturbing embeddings at various positions across multiple layers yields at least two benefits. DP-Forward users are only required to download pipeline parts for deriving noisy embeddings, which is more storage- and time-efficient than deriving noisy gradients. Together with our prior attempts [21, 80] at sanitizing input text tokens and output sentence embeddings, we provide a full suite of forward-pass signal sanitization options for users only to share their sanitized data for LM-as-a-Service APIs while protecting privacy.

Beyond the theoretical contribution of two local DP notions and the experimental comparisons with baselines (e.g., GM, MVG, and DP-SGD) across three typical NLP tasks, we investigate the hyperparameter configuration for reproducible validations of DP-Forward's potential in terms of efficiency, accuracy, and its ability to withstand diverse against diverse attacks.

Altogether, our new perspective leads to a better approach to privacy-aware deep neural network training, challenging the traditional wisdom focusing on gradients. As a new paradigm for local DP in fine-tuning and inference, our work paves the way for a myriad of possibilities for new machine-learning privacy research [54], e.g., generalization to transformer-based computer vision tasks.

## ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their comments and Ashwin Machanavajhala for his comments on a related Ph.D. thesis. Sherman Chow is supported in part by the General Research Funds (CUHK 14209918, 14210319, 14210621), Research Grants Council, University Grants Committee, Hong Kong. Authors at OSU are sponsored in part by NSF IIS #1815674, NSF CAREER #1942980, and Ohio Supercomputer Center [55]. Tianhao Wang is supported in part by National Science Foundation (NSF) with grants CNS-2220433 and OAC-2319988.

## REFERENCES

- [1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.
- [2] Prince Osei Aboagye, Yan Zheng, Chin-Chia Michael Yeh, Junpeng Wang, Wei Zhang, Liang Wang, Hao Yang, and Jeff M. Phillips. 2022. Normalization of Language Embeddings for Cross-Lingual Alignment. In *ICLR (Poster)*. 32 pages.
- [3] John Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson Garfinkel, Micah Heineck, Christine Heiss, Robert Johns, Daniel Kifer, Philip Leclerc, Ashwin Machanavajhala, Brett Moran, William Sexton, Matthew Spence, and Pavel Zhuravlev. 2022. The 2020 Census Disclosure Avoidance System TopDown Algorithm. *Harvard Data Science Review* Special Issue 2: Differential Privacy for the 2020 U.S. Census (Jun 24 2022), 77 pages. <https://hdsr.mitpress.mit.edu/pub/7evz361i>.
- [4] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Anna Pazzi. 2018. Invited Paper: Local Differential Privacy on Metric Spaces: Optimizing the Trade-Off with Utility. In *CSF*. 262–267.
- [5] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2022. Large-Scale Differentially Private BERT. In *Findings of EMNLP*. 6481–6491.
- [6] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. arXiv:1607.06450.
- [7] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *ICML*. 403–412.
- [8] Santiago Zanella Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing Information Leakage of Updates to Natural Language Models. In *CCS*. 363–375.
- [9] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. 1175–1191.
- [10] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J. Su. 2019. Deep Learning with Gaussian Differential Privacy. arXiv:1911.11607.
- [11] Robert Istvan Busa-Fekete, Andres Munoz Medina, Umar Syed, and Sergei Vassilvitskii. 2021. On the pitfalls of label differential privacy. In *NeurIPS Workshop*. 6 pages.
- [12] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security*. 267–284.
- [13] 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. 2021. Extracting Training Data from Large Language Models. In *USENIX Security*. 2633–2650.
- [14] Thee Chanyswad, Alex Dytso, H. Vincent Poor, and Prateek Mittal. 2018. MVG Mechanism: Differential Privacy under Matrix-Valued Query. In *CCS*. 230–246.
- [15] Melissa Chase and Sherman S. M. Chow. 2009. Improving privacy and security in multi-authority attribute-based encryption. In *CCS*. 121–130.
- [16] Kamalika Chaudhuri and Daniel J. Hsu. 2011. Sample Complexity Bounds for Differentially Private Learning. In *COLT*. 155–186.
- [17] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In *NIPS*. 289–296.
- [18] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially Private Empirical Risk Minimization. *J. Mach. Learn. Res.* 12 (2011), 1069–1109.
- [19] George Dasoulas, Kevin Scaman, and Aladin Virmaux. 2021. Lipschitz normalization for self-attention layers with application to graph neural networks. In *ICML*, Vol. 139. 2456–2466.
- [20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.
- [21] Minxin Du, Xiang Yue, Sherman S. M. Chow, and Huan Sun. 2023. Sanitizing Sentence Embeddings (and Labels) for Local Differential Privacy. In *WWW*. 2349–2359.
- [22] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*. 265–284.
- [23] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.
- [24] Yanai Elazar and Yoav Goldberg. 2018. Adversarial Removal of Demographic Attributes from Text Data. In *EMNLP*. 11–21.
- [25] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*. 2468–2479.
- [26] Mani Malek Esmaeili, Ilya Mironov, Karthik Prasad, Igor Shilov, and Florian Tramèr. 2021. Antipodes of Label Differential Privacy: PATE and ALIBI. In *NeurIPS*. 6934–6945.
- [27] Hugging Face. 2023. BERT base model (uncased). <https://huggingface.co/bert-base-uncased>, last accessed: September 12, 2023.



- [28] Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy and Utility-Preserving Textual Analysis via Calibrated Multivariate Perturbations. In *WSDM*. 178–186.
- [29] Oluwaseyi Feyisetan, Tom Diethe, and Thomas Drake. 2019. Leveraging Hierarchical Representations for Preserving Privacy and Utility in Text. In *ICDM*. 210–219.
- [30] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. 2018. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *CCS*. 619–633.
- [31] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep Learning with Label Differential Privacy. In *NeurIPS*. 27131–27145.
- [32] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. 2021. Numerical Composition of Differential Privacy. In *NeurIPS*. 11631–11642.
- [33] Robert J. Hall, Larry A. Wasserman, and Alessandro Rinaldo. 2013. Random Differential Privacy. *J. Priv. Confidentiality* 4, 2 (2013), 43–59.
- [34] Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis, 2nd Ed.* Cambridge University Press, N/A.
- [35] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *ICML*, Vol. 97. 2790–2799.
- [36] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR (Poster)*. 13 pages.
- [37] Tianxi Ji, Pan Li, Emre Yilmaz, Erman Ayday, Yanfang Ye, and Jinyuan Sun. 2021. Differentially Private Binary- and Matrix-Valued Data Query: An XOR Mechanism. *Proc. VLDB Endow.* 14, 5 (2021), 849–862.
- [38] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2008. What Can We Learn Privately?. In *FOCS*. 531–540.
- [39] Hyunjik Kim, George Papamakarios, and Andriy Mnih. 2021. The Lipschitz Constant of Self-Attention. In *ICML*, Vol. 139. 5562–5571.
- [40] Beatrice Laurent and Pascal Massart. 2000. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics* 28, 5 (2000), 1302–1338.
- [41] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified Robustness to Adversarial Examples with Differential Privacy. In *S&P*. 656–672.
- [42] Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron C. Wallace. 2021. Does BERT Pretrained on Clinical Notes Reveal Sensitive Data?. In *NAACL-HLT*. 946–959.
- [43] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.* 24, 6 (2015), 757–781.
- [44] Jing Li, Aixin Sun, Janglei Han, and Chenliang Li. 2022. A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.* 34, 1 (2022), 50–70.
- [45] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2022. Large Language Models Can Be Strong Differentially Private Learners. In *ICLR*. 30 pages.
- [46] Lingjuan Lyu, Xuanli He, and Yitong Li. 2020. Differentially Private Representation for NLP: Formal Guarantee and An Empirical Study on Privacy and Fairness. In *Findings of EMNLP*. 2355–2365.
- [47] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *ACL*. 142–150.
- [48] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *NeurIPS*. 1022–1035.
- [49] James McDermott and Richard S. Forsyth. 2016. Diagnosing a disorder in a classification benchmark. *Pattern Recognit. Lett.* 73 (2016), 41–43.
- [50] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR (Poster)*. 14 pages.
- [51] Casey Meehan, Khalil Mrini, and Kamalika Chaudhuri. 2022. Sentence-level Privacy for Document Embeddings. In *ACL*. 3367–3380.
- [52] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?. In *EMNLP*. 11048–11064.
- [53] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.
- [54] Lucien K. L. Ng and Sherman S. M. Chow. 2023. SoK: Cryptographic Neural Network Computation. In *S&P*. 497–514.
- [55] OSC. 1987. Ohio Supercomputer Center. <http://osc.edu/ark:/19495/F5s1ph73>
- [56] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy Risks of General-Purpose Language Models. In *S&P*. 1314–1331.
- [57] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.
- [58] Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Natural Language Understanding with Privacy-Preserving BERT. In *CIKM*. 1488–1497.
- [59] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI Report.
- [60] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. OpenAI Report.
- [61] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*. 3980–3990.
- [62] Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4NT: Author Attribute Anonymity by Adversarial Training of Neural Machine Translation. In *USENIX Security*. 1633–1650.
- [63] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *S&P*. 3–18.
- [64] Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models. In *CCS*. 377–390.
- [65] Liwei Song and Prateek Mittal. 2021. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security*. 2615–2632.
- [66] Thomas Steinke. 2022. Composition of Differential Privacy & Privacy Amplification by Subsampling. [arXiv:2210.00597](https://arxiv.org/abs/2210.00597).
- [67] Latanya Sweeney. 2015. Only You, Your Doctor, and Many Others May Know. *Technology Science* 2015092903, 9 (2015), 29.
- [68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NeurIPS*. 5998–6008.
- [69] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR (Poster)*. 20 pages.
- [70] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *JASA* 60, 309 (1965), 63–69.
- [71] Benjamin Weggenmann and Florian Kerschbaum. 2018. SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. In *SIGIR*. 305–314.
- [72] Ruihan Wu, Jin Peng Zhou, Kilian Q. Weinberger, and Chuan Guo. 2023. Does Label Differential Privacy Prevent Label Inference Attacks?. In *AISTAT*. 4336–4347.
- [73] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. [arXiv:1609.08144](https://arxiv.org/abs/1609.08144).
- [74] Jungang Yang, Liyao Xiang, Weiting Li, Wei Liu, and Xinbing Wang. 2021. Improved Matrix Gaussian Mechanism for Differential Privacy. [arXiv:2104.14808](https://arxiv.org/abs/2104.14808).
- [75] Jungang Yang, Liyao Xiang, Jiahao Yu, Xinbing Wang, Bin Guo, Zhetao Li, and Baohun Li. 2023. Matrix Gaussian Mechanisms for Differentially-Private Learning. *IEEE Trans. Mob. Comput.* 22, 2 (2023), 1036–1048.
- [76] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *CSF*. 268–282.
- [77] 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. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. [arXiv:2109.12298](https://arxiv.org/abs/2109.12298).
- [78] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2022. Differentially Private Fine-tuning of Language Models. In *ICLR (Poster)*. 19 pages.
- [79] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021. Large Scale Private Learning via Low-rank Reparametrization. In *ICML*. 12208–12218.
- [80] Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman S. M. Chow. 2021. Differential Privacy for Text Analytics via Natural Text Sanitization. In *Findings of ACL/IJCNLP*. 3853–3866.
- [81] Mingxun Zhou, Tianhao Wang, T.-H. Hubert Chan, Giulia Fanti, and Elaine Shi. 2022. Locally Differentially Private Sparse Vector Aggregation. In *S&P*. 422–439.

## A TOKEN-LEVEL DP-FORWARD

### A.1 Definition and Related Notions

**Definition 6 (Token-level SeqLDP).** For  $\epsilon \geq 0, 0 \leq \delta \leq 1$ ,  $M$  fulfills token-level  $(\epsilon, \delta)$ -SeqLDP, if  $\forall X \approx X'$  that differ in any single token but with the same  $y$ , and any possible output subset  $O$ ,

$$\Pr[M(X, y) \in O] \leq e^\epsilon \Pr[M(X', y) \in O] + \delta.$$



Despite a token-level notion, our experiments (Appendix A.3) show that when  $f(\cdot)$  is only the input embedding layer, our token-level SeqLDP designs can also effectively mitigate MIAs on *entire* sequences, with up to 20pp accuracy gains at the same choices of  $\epsilon$ . It is not necessarily weaker than sequence-level CDP (as offered by DP-SGD). One might doubt its usefulness since two neighboring sequences may be too similar. Nevertheless, there are cases where a sentence, e.g., “How’s it going” may not matter in a bigger unit (paragraph/essay) of the training data either. Moreover, a token (e.g., yes/no) can play a crucial role, e.g., in named entity recognition [44]. Our LDP guarantee is for *any* such two sequences, covering the wide spectrum between “too similar” and radically different cases.

Note that weakening privacy notions by itself is not our goal<sup>12</sup>. Protection at the token level has been studied under metric-DP [28, 58], a relaxation of LDP. They require even much larger  $\epsilon$ , say, 175. Our goal of studying token-level SeqLDP is to narrow the gap between theory and practice, i.e., provable privacy notions tailored to the protection targets (the first few layers vs. the whole pipeline).

## A.2 Two Token-level SeqLDP Designs

For token-level SeqLDP, we need to bound a “new”  $S_2(f)$ ,  $\forall X \simeq X'$ , which should be tight and smaller than the one over  $\forall X, X'$ , hence producing smaller noise for better utility at meaningful *token-level*  $\epsilon$ . It is still non-trivial since  $f(\cdot)$ , except for being the input embedding layer, may differ in every entry for even  $X \simeq X'$ . One could also normalize the entire  $f(\cdot)$  for  $S_2(f)$ ,  $\forall X \simeq X'$ , which “degenerates” to the token-level SeqLDP. Instead, we tailor two designs to estimate a tighter  $S_2(f)$  than the “general” one for only the input embedding layer and the first two layers, respectively. Specifically, we employ *row-wise* normalization and the Lipschitz continuity [39].

**After the Input Embedding Layer.** When  $f(\cdot)$  is only the input embedding layer, we work on each row  $x_i$  independently:  $\|x_i\|_2 = C, \forall i \in [n]$ , where  $\|\cdot\|_2$  is vector 2-norm. As a token only affects one row, we have  $S_2(f) = C$ , independent of whether the embedding layer will be updated. Again with  $\mathcal{B}$ , we can draw noise  $Z \in \mathbb{R}^{n \times d}$ .

**In the First MHA Sub-layer.** The second option could be adding  $Z$  right after the first MHA sub-layer:  $\text{MHA}(X) + Z$ , where  $\text{MHA}(\cdot)$  is the concatenation of  $\text{Att}_i(\cdot)$ ,  $i \in [h]$ . Yet, it is non-trivial to estimate  $S_2(f)$  of  $\text{MHA}(\cdot)$  as  $\text{Att}_i(\cdot)$ , let alone  $\text{MHA}(\cdot)$ , is not Lipschitz [39].

**Definition 7 (Lipschitz Continuity).** Given two metric spaces  $(X, d_X)$  and  $(\mathcal{Y}, d_Y)$ , a function  $f : X \rightarrow \mathcal{Y}$  is Lipschitz continuous ( $K$ -Lipschitz) if there exists a constant  $K \geq 0$ ,

$$d_Y(f(X), f(X')) \leq K d_X(X, X'), \forall X, X' \in X.$$

The smallest  $K$  is the Lipschitz constant, denoted by  $\text{Lip}(f)$ .

We consider that  $X$  is the space of *row-wise* normalized matrices in  $\mathbb{R}^{n \times d}$ ,  $\mathcal{Y}$  is the output space  $\mathbb{R}^{n \times d/h}$  of  $\text{Att}_{i \in [h]}(\cdot)$  or  $\text{MHA}(\cdot)$ , and  $d_X = d_Y = \|\cdot\|_F$  (or  $p$ -norm  $\|\cdot\|_p$ ).  $\text{Lip}(f)$  generalizes  $S_2(f)$  since it focuses on *any* two inputs rather than just neighboring ones, allowing us to estimate an upper bound for  $S_2(f)$  given  $\text{Lip}(f)$ .

The non-Lipschitz continuity stems from the non-linear Softmax activation, which takes pairwise dot products as input [39]. To make MHA Lipschitz, one might apply pairwise  $L_2$ -distances (hence called

$L_2$ -MHA) [39] or add a normalization step called LipschitzNorm [19] in  $\text{softmax}(\cdot)$ . Unfortunately, estimating  $\text{Lip}(f)$  of  $L_2$ -MHA needs to solve an intractable optimization problem, and LipschitzNorm is ill-suited for the high-dimensional BERT attention.

Instead of adding  $Z$  to the outputs of  $\text{MHA}(\cdot)$  or  $\text{Att}_i(\cdot)$ , we can shift  $f(\cdot)$  inside  $\text{softmax}(\cdot)$ , where estimating  $\text{Lip}(f)$  or  $S_2(f)$  is feasible, e.g., the linear maps used to derive  $Q, K, V$  matrices. Considering a linear map  $f(x) = xW$  with  $W \in \mathbb{R}^{d \times d/h}$  and  $x \in \mathbb{R}^d$ , the 2-norm  $\text{Lip}_2(f)$  is the largest singular value  $\sigma_{\max}(W)$  [39]. When generalizing  $f(\cdot)$  for any two matrices  $X \simeq X'$ , we can estimate

$$S_2(f) = \sup \|f(X) - f(X')\|_F = \|f(x)\|_2 \leq C \cdot \sigma_{\max}(W).$$

We can now respectively derive the noisy  $Q, K, V$  matrices for  $p(\cdot)$ . The first step is to draw noise  $Z^{Q^*, K^*, V^*} \in \mathbb{R}^{n \times d}$  given  $W^{Q^*, K^*, V^*} \in \mathbb{R}^{d \times d}$ . It requires us to either estimate  $\sigma_{\max}(W^{Q^*, K^*, V^*})$  on the fly via power iteration or fix the linear maps in each forward pass of fine-tuning or inference. We then compute  $XW^{Q^*, K^*, V^*} + Z^{Q^*, K^*, V^*}$ , which are reshaped into  $3h$  matrices of size  $n \times d/h$  for  $\text{Att}_{i \in [h]}(\cdot)$ .

**Theorem 7.** The two instances (with row-wise normalization) for fine-tuning or inference fulfill token-level  $(\epsilon, \delta)$ -(Seq)LDP.

The proof is equivalent to our approach for (Seq)LDP. One just needs to compute  $S_2(f)$ ,  $\forall X \simeq X'$  properly, and we did.

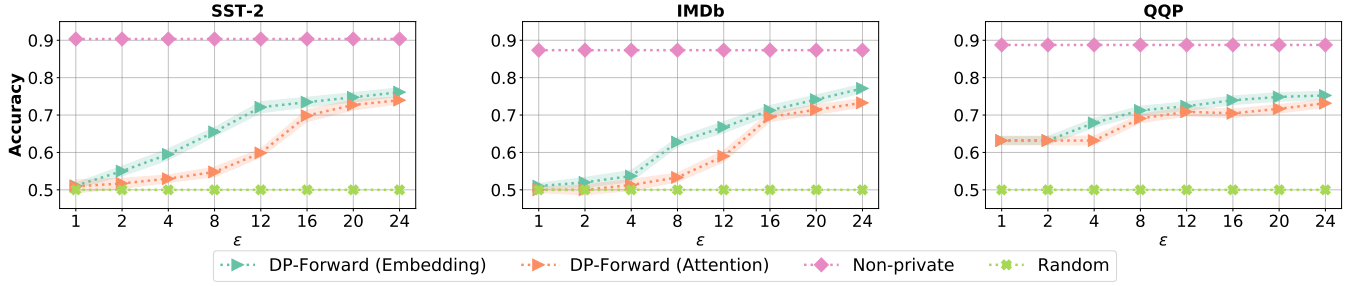
**Discussion.** For minimal changes to the pipeline, we adopt the raw WordPiece [73], which splits text into sub-words; using word-level tokenization yields word-level (Seq)LDP. Our notion can also extend to *phrase-level* (Seq)LDP by directly using the group privacy [23] or dedicatedly computing the  $L_2$ -sensitivity smaller than  $c \cdot S_2(f)$  for two sequences differing in (consecutive)  $c$  tokens. Typically,  $c$  is small since a few tokens are enough for most sensitive information. One could also add noise deeper in a pipeline using  $S_2(f_1 \circ f_2) \leq S_2(f_1) \cdot S_2(f_2)$ , where  $f_1 \circ f_2$  is function composition  $f_1(f_2(\cdot))$ . We then need to estimate  $S_2(f)$  of each (component of) sub-layer. For example, FFN( $\cdot$ ) has two linear maps  $W_1$  and  $W_2$  with  $\text{ReLU}(\cdot)$  in between, where  $S_2(f)$  of  $\text{ReLU}(\cdot)$  is 1. For  $W_{1,2}$ , its  $S_2(f)$  is bounded by  $\sqrt{d}C\sigma_{\max}(W_{1,2})$  since  $\|\cdot\|_F \leq \sqrt{d}\|\cdot\|_2$  with  $d$  as the rank. We can also estimate  $S_2(f)$  of  $\text{LN}(\cdot)$  from its Lipschitz constant [39]. When  $f(\cdot)$  is composed of more layers, we can only get a looser estimation on the final  $S_2(f)$ . Hence, our general recommendation is to add noise early when estimating a tight  $S_2(f)$  is feasible.

## A.3 More Experiment Results

We also study the privacy-accuracy tradeoff on all three tasks for our two token-level SeqLDP designs when tuning local  $\epsilon$ . The results are compared with the non-private baseline and fine-tuning using MVG noise. Figure 5 shows task accuracy increases with  $\epsilon$ . Perturbing input embeddings for token-level (vs. sequence-level) SeqLDP can achieve remarkable accuracy gain, e.g.,  $\sim 0.7$  vs. 0.5 for IMDB.

We evaluate the two MIAs on SST-2 fine-tuned by our two token-level SeqLDP instances. Table 10 shows the results, with success rates within 0.48–0.52 (like random guessing) bolded. Even if the provable guarantee is at the token level, our instances can notably reduce the success rates of the confidence-based attack by  $\sim 14$ pp and the entropy-based one by  $\sim 11$ pp, compared to the non-private baseline.

<sup>12</sup>As a related example, in image classification, PixelDP [41] has been proposed for a DP notion defined upon pixels. Its motivation is robustness to adversarial examples.

Figure 5: Privacy-accuracy tradeoff of token-level SeqLDP instances when tuning local  $\epsilon$ 

Local $\epsilon$	Method	Attack Success Rate	
		Entropy	Confidence
$\infty$	Non-private baseline	0.659	0.645
8	DP-Forward (Embedding)	0.536	<b>0.503</b>
	DP-Forward (Attention)	0.545	<b>0.516</b>
16	DP-Forward (Embedding)	0.542	<b>0.509</b>
	DP-Forward (Attention)	0.552	<b>0.519</b>
24	DP-Forward (Embedding)	0.552	<b>0.516</b>
	DP-Forward (Attention)	0.559	0.523

Table 10: Success Rates of the two (sequence-level) MIAs on our token-level SeqLDP instances

## B RELEVANT MATRIX ALGEBRA

**Proposition 1.** The PDF defined in Eq. (5) and the matrix-trace-based one used in MVG [14] are equivalent.

PROOF. For the numerator part in Eq. (5), we have

$$\begin{aligned} & \|U^{-1}(Z - M)V^{-\top}\|_F^2 \\ &= \text{Tr}[V^{-1}(Z - M)^\top U^{-\top}U^{-1}(Z - M)V^{-\top}] \\ &= \text{Tr}[V^{-1}(Z - M)^\top \Sigma^{-1}(Z - M)V^{-\top}], \end{aligned}$$

where  $\text{Tr}(\cdot)$  denotes the matrix trace. Denote

$$A = V^{-1}(Z - M)^\top \Sigma^{-1}(Z - M)V^{-\top}.$$

We compute

$$B = V^{-\top}AV^\top = \Psi^{-1}(Z - M)^\top \Sigma^{-1}(Z - M),$$

which is a similar matrix of  $A$ , and hence  $\text{Tr}(A) = \text{Tr}(B)$ . So, the two PDFs are equivalent since

$$\|U^{-1}(Z - M)V^{-\top}\|_F^2 = \text{Tr}(B). \quad \square$$

**Theorem 8 (Singular Value Decomposition or SVD [34]).** A matrix  $A \in \mathbb{R}^{n \times d}$  can be decomposed as  $W_1 \Lambda W_2^\top$ , where  $W_1 \in \mathbb{R}^{n \times n}$  and  $W_2 \in \mathbb{R}^{d \times d}$  are unitary, and  $\Lambda$  is an  $n \times d$  diagonal matrix whose diagonal entries are ordered singular values of  $A$ , denoted by  $\sigma_1(A) \geq \dots \geq \sigma_r(A) \geq 0$  (or simply  $\sigma(A)$ ) with  $r = \min\{n, d\}$ .

**Lemma 4.** Given a matrix  $A \in \mathbb{R}^{n \times d}$  and two orthogonal matrices  $W_1 \in \mathbb{R}^{n \times n}$ ,  $W_2 \in \mathbb{R}^{d \times d}$ , we have  $\|A\|_F = \|W_1 A\|_F = \|A W_2\|_F$ ;  $\|\cdot\|_F$  is immune to the pre- and post-orthogonal transformation.

PROOF. We first prove that  $\|A\|_F = \|W_1 A\|_F$  by

$$\|W_1 A\|_F^2 = \text{Tr}(A^\top W_1^\top W_1 A) = \text{Tr}(A^\top A) = \|A\|_F^2,$$

and similarly we can prove that  $\|A\|_F = \|A W_2\|_F$ .  $\square$

**Lemma 5.** For  $A \in \mathbb{R}^{n \times d}$ ,  $\|A\|_F^2 = \sum_{i=1}^r \sigma_i^2(A)$ , where  $\sigma_i(A)$  is the  $i^{\text{th}}$  singular value of  $A$  and  $r = \min\{n, d\}$ .

PROOF. The SVD of  $A$  is  $W_1 \Lambda W_2^\top$ . By Lemma 4, we have

$$\|A\|_F^2 = \|W_1 \Lambda W_2^\top\|_F^2 = \|\Lambda\|_F^2 = \sum_{i=1}^r \sigma_i^2(A). \quad \square$$

**Lemma 6 (Lemma 4 [74]).** Given matrices  $A \in \mathbb{R}^{n \times n}$ ,  $B \in \mathbb{R}^{n \times d}$ ,  $C \in \mathbb{R}^{d \times d}$ , we have  $\|ABC\|_F^2 \leq \sum_{i=1}^r \sigma_i^2(A) \sigma_i^2(B) \sigma_i^2(C)$  where  $\sigma_i(\cdot)$  is the  $i^{\text{th}}$  singular value, and  $r = \min\{n, d\}$ .

PROOF. With SVD,  $A, B, C$  are decomposed as

$$A = W_{A_1} \Lambda_A W_{A_2}^\top, \quad B = W_{B_1} \Lambda_B W_{B_2}^\top, \quad C = W_{C_1} \Lambda_C W_{C_2}^\top.$$

Based on Lemma 5, we have

$$\begin{aligned} \|ABC\|_F^2 &= \|W_{A_1} \Lambda_A W_{A_2}^\top W_{B_1} \Lambda_B W_{B_2}^\top W_{C_1} \Lambda_C W_{C_2}^\top\|_F^2 \\ &= \|\Lambda_A W \Lambda_B W' \Lambda_C\|_F^2, \end{aligned}$$

where  $W = W_{A_2}^\top W_{B_1} = (w_{ij})_{n \times n}$  and  $W' = W_{B_2}^\top W_{C_1} = (w'_{ij})_{d \times d}$  are still two unitary matrices. We further have

$$\begin{aligned} \|ABC\|_F^2 &= \sum_{i=1}^n \sum_{j=1}^d \sigma_i^2(A) \sigma_j^2(C) \left[ \sum_{k=1}^r \sigma_k(B) w_{ik} w'_{kj} \right]^2 \\ &= \sum_{i=1}^n \sum_{j=1}^d \sigma_i^2(A) \sigma_j^2(C) \beta_{ij}^2, \end{aligned}$$

where  $\beta_{ij} = \sum_{k=1}^r \sigma_k(B) w_{ik} w'_{kj}$ . Hence, we need to show

$$\sum_{i=1}^n \sum_{j=1}^d \sigma_i^2(A) \sigma_j^2(C) \beta_{ij}^2 \leq \sum_{i=1}^r \sigma_i^2(A) \sigma_i^2(B) \sigma_i^2(C). \quad (7)$$

Following the strategy in [74] (cf. Eq. (29), (30)), we rewrite  $\sigma_i^2(A)$  and  $\sigma_j^2(C)$  using non-negative values  $\xi_t$  and  $\eta_s$  s.t.

$$\sigma_i^2(A) = \sum_{t=1}^n \xi_t, \quad t \in [1, n]; \quad \sigma_j^2(C) = \sum_{s=1}^d \eta_s, \quad s \in [1, d].$$

For  $i \in [1, n], j \in [1, d]$ , we denote  $\gamma_{ij} = \sigma_i(B)$ , if  $i = j$ ;  $\gamma_{ij} = 0$ , otherwise. Then, we transform the Eq. (7) as

$$\begin{aligned} & \sum_{i=1}^r \sigma_i^2(A) \sigma_i^2(B) \sigma_i^2(C) - \sum_{i=1}^n \sum_{j=1}^d \sigma_i^2(A) \sigma_j^2(C) \beta_{ij}^2 \\ &= \sum_{i=1}^n \sum_{j=1}^d (\gamma_{ij}^2 - \beta_{ij}^2) \sigma_i^2(A) \sigma_j^2(C) \\ &= \sum_{i=1}^n \sum_{j=1}^d (\gamma_{ij}^2 - \beta_{ij}^2) \sum_{t=1}^n \xi_t \sum_{s=j}^d \eta_s \\ &= \sum_{t=1}^n \sum_{s=1}^d \xi_t \eta_s \sum_{i=1}^t \sum_{j=1}^s (\gamma_{ij}^2 - \beta_{ij}^2). \end{aligned}$$

Since  $\xi_t, \eta_s$  are non-negative, we only need to show

$$\sum_{i=1}^t \sum_{j=1}^s (\gamma_{ij}^2 - \beta_{ij}^2) \geq 0. \quad (8)$$

However, the original proof [74] has two issues: i)  $t > s$  is not considered, and ii) the commutative law of matrix multiplication in Eq. (35) does not hold as  $E(t)$  in Eq. (34) is not a standard diagonal matrix. To address them, we have

$$\sum_{i=1}^t \sum_{j=1}^s \gamma_{ij}^2 = \sum_{k=1}^{\min\{t,s\}} \sigma_k^2(B).$$

We then denote a sub-matrix  $B^* = (\beta_{ij})$  for  $i \in [1, t], j \in [1, s]$  of  $W \Lambda_B W'$ . With SVD of  $B^*$ , we have

$$\sum_{i=1}^t \sum_{j=1}^s \beta_{ij}^2 = \|B^*\|_F^2 = \sum_{k=1}^{\min\{t,s\}} \sigma_k^2(B^*) \leq \sum_{k=1}^{\min\{t,s\}} \sigma_k^2(B).$$

The last inequality is due to  $\sigma_k(B^*) \leq \sigma_k(B)$  for  $\forall k \in [1, r]$  [34]. So, Inequality (8) holds.  $\square$

## C PROOFS FOR OUR ANALYTIC MATRIX GAUSSIAN MECHANISM

This section proof Lemma 1, Lemma 2, and Theorem 6 in Section 4.2.

**PROOF OF LEMMA 1.** Recall that  $\mathcal{M}(f(\mathcal{X})) = f(\mathcal{X}) + Z$  with  $Z \sim \mathcal{MN}_{n,d}(0, \Sigma, \Psi)$ , the probability of  $\mathcal{M}(f(\mathcal{X})) = O$  is

$$\Pr[\mathcal{M}(f(\mathcal{X})) = O] = \frac{\exp(-\frac{1}{2} \|U^{-1}(O - f(\mathcal{X}))V^{-\top}\|_F^2)}{(2\pi)^{nd/2} |\Psi|^{d/2} |\Sigma|^{n/2}}.$$

Similarly, we can compute  $\Pr[\mathcal{M}(f(\mathcal{X}')) = O]$ . By plugging them into  $\mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'}(O)$ , and let  $\Delta = f(\mathcal{X}) - f(\mathcal{X}')$ ,

$$\begin{aligned} \mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'}(O) &= \ln \frac{\exp(-\frac{1}{2} \|U^{-1}(O - f(\mathcal{X}))V^{-\top}\|_F^2)}{\exp(-\frac{1}{2} \|U^{-1}(O - f(\mathcal{X}'))V^{-\top}\|_F^2)} \\ &= \frac{1}{2} \|U^{-1}(Z + \Delta)V^{-\top}\|_F^2 - \frac{1}{2} \|U^{-1}ZV^{-\top}\|_F^2 \\ &= \frac{1}{2} \|U^{-1}\Delta V^{-\top}\|_F^2 + \langle \text{vec}(U^{-1}\Delta V^{-\top}), \text{vec}(U^{-1}ZV^{-\top}) \rangle, \end{aligned}$$

where  $\text{vec}(\cdot)$  is the vectorization of a matrix and  $\langle \cdot, \cdot \rangle$  denotes the inner product. For easy presentation, we denote  $Z' = U^{-1}ZV^{-\top}$  and  $\Delta' = U^{-1}\Delta V^{-\top}$ , and then we re-write

$$\mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'}(O) = \frac{1}{2} \|\Delta'\|_F^2 + \langle \text{vec}(\Delta'), \text{vec}(Z') \rangle.$$

Given Lemma 3,  $Z' \sim \mathcal{MN}_{n,d}(0, I_n, I_d)$  with each entry i.i.d. drawn from  $\mathcal{N}(0, 1)$ .  $\langle \text{vec}(\Delta'), \text{vec}(Z') \rangle$  is thus the  $\Delta'$ -weighted sum of  $nd$  i.i.d. Gaussian random variables, which is a Gaussian variable<sup>13</sup>  $\mathcal{N}(0, \|\Delta'\|_F^2)$  too. So  $\mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'} \sim \mathcal{N}(\eta, 2\eta)$ ,  $\eta = \frac{1}{2} \|\Delta'\|_F^2$ .  $\square$

**PROOF OF LEMMA 2.** With Lemma 1 and the CDF, we have

$$\begin{aligned} \Pr[\mathcal{L}_{\mathcal{M}, \mathcal{X}, \mathcal{X}'} \geq \epsilon] &= \Pr[\mathcal{N}(\eta, 2\eta) \geq \epsilon] \\ &= \Pr[\mathcal{N}(0, 1) \geq \frac{\epsilon - \eta}{\sqrt{2\eta}}] = \Pr[\mathcal{N}(0, 1) \leq \frac{\eta - \epsilon}{\sqrt{2\eta}}] \\ &= \Phi\left(\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right), \end{aligned}$$

where we used  $\mathcal{N}(\eta, 2\eta) = \eta + \mathcal{N}(0, 1)/\sqrt{2\eta}$  and the symmetry of the standard normal distribution  $\Pr[\mathcal{N}(0, 1) \geq t] = \Pr[\mathcal{N}(0, 1) \leq -t]$ .

A similar argument applied to  $\mathcal{L}_{\mathcal{M}, \mathcal{X}', \mathcal{X}}$  yields

$$\Pr[\mathcal{L}_{\mathcal{M}, \mathcal{X}', \mathcal{X}} \leq -\epsilon] = \Phi\left(-\frac{\|\Delta'\|_F}{2} - \frac{\epsilon}{\|\Delta'\|_F}\right). \quad \square$$

**PROOF OF THEOREM 6.** The proof boils down to two directions. From  $(\epsilon, \delta)$ -DP (Theorem 5) to Theorem 6, the proof directly follows from all the derivations in Section 4.2. For the inverse direction, it is sufficient to show that  $\|\Delta'\|_F \leq \mathcal{B}$  holds for  $\forall \mathcal{X} \approx \mathcal{X}'$  given Theorem 6. In particular, for  $\|\Delta'\|_F = \|U^{-1}\Delta V^{-\top}\|_F$ , we have

$$\begin{aligned} \|U^{-1}\Delta V^{-\top}\|_F^2 &\leq \sum_{i=1}^r \frac{\sigma_i^2(\Delta)}{\sigma_{n-i+1}^2(U) \sigma_{d-i+1}^2(V)} \\ &\leq \frac{\sum_{i=1}^r \sigma_i^2(\Delta)}{\sigma_n^2(U) \sigma_d^2(V)} \leq \frac{\|\Delta\|_F^2}{S_2^2(f)/\mathcal{B}^2} \leq \mathcal{B}^2, \end{aligned}$$

where the first inequality is due to Lemma 6, the second one holds since  $\sigma_n(U)$  and  $\sigma_d(V)$  are the smallest singular values among the others, and the third one is from Theorem 6.  $\square$

<sup>13</sup>[en.wikipedia.org/wiki/Sum\\_of\\_normally\\_distributed\\_random\\_variables](https://en.wikipedia.org/wiki/Sum_of_normally_distributed_random_variables)