Per-Attribute Privacy in Large Language Models Using Matrix-Variate Gaussian Mechanism

Islam A. Monir, Gabriel Ghinita
College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar
Email: {ismo58166,gghinita}@hbku.edu.qa

***Abstract*—Large Language Models (LLMs) have become central to modern NLP applications, yet their reliance on sensitive training data introduces significant privacy risks. Traditional approaches to differential privacy, such as DP-SGD, apply uniform noise at the gradient level and treat all features equally, ignoring the inherent correlations in structured sequence data. In this work, we propose a novel forward-pass privacy mechanism that applies per-attribute differential privacy across correlated sequence inputs. Our framework leverages the Matrix-Variate Gaussian (MVG) mechanism to inject structured, directional noise during the forward computation, enabling fine-grained privacy control that aligns with attribute sensitivity. Privacy budgets are assigned on a per-token basis using an Inverse Gaussian Distribution, allowing position-aware adaptation across input sequences. To propagate these budgets through the model, we introduce a data-independent Layer-Wise Contribution Propagation (LCP) algorithm that maps input sensitivity to output features, even in transformer architectures. We validate our method on the SST-2 sentiment classification benchmark, demonstrating improved utility over existing approaches such as DP-SGD and DP-Forward, particularly under strict privacy regimes. Our results highlight the benefits of structured noise in preserving utility while ensuring strong, attribute-level privacy in models trained on correlated sequential data.**

I. Introduction

Large language models (LLMs) showed impressive performance in tasks like text generation, summarization, translation, and question answering. However, they also raise privacy risks, as models often memorize training data, making them susceptible to membership inference attacks (MIAs) [1]. Differential Privacy (DP) [2] is a widely adopted framework which ensures that an individual's participation in training has a mathematically bounded effect on the model's output. The most common approach, Differentially Private Stochastic Gradient Descent (DP-SGD) [2], [3], enforces DP by clipping per-sample gradients and injecting noise during training updates. However, DP-SGD applies uniform privacy budget (ε) per sample, assuming all attributes require equal protection, whereas in real-world scenarios different attributes have varying sensitivity levels.

We explore individualized DP at the attribute level (or token level in the context of LLM), where each attribute within a sample receives a different privacy budget based on its sensitivity. This approach enables finer-grained privacy control, allowing highly-sensitive attributes to receive stronger privacy protection, while less-sensitive attributes retain more detail for improved model utility. Furthermore, tokens in LLM are often highly-interdependent, e.g., word embeddings within a sentence are correlated. If noise is added independently to each token, it will destroy meaningful feature relationships. We leverage the Matrix-Variate Gaussian (MVG) mechanism [4], which models noise as a matrix distribution, ensuring that perturbations preserve feature relationships. A crucial property of MVG is its ability to apply *directional noise*, which is dynamically adjusted based on feature sensitivity. This way, we can apply more noise along sensitive attribute directions.

Another limitation of DP-SGD is that it perturbs gradient values, after feature representations have been processed, which provides little control over the privacy-accuracy trade-off. Instead, our approach adds MVG in the forward phase of training, early within the model's computational pipeline. This allows flexibility by applying per-attribute noise. It also provides more stable optimization, as noise does not interfere with gradient updates, leading to better learning efficiency.

Our key contributions include: (i) a flexible per-token DP sanitization process using MVG, where each feature receives custom privacy protection based on its sensitivity; (ii) incorporation of MVG's directional noise properties to ensure that perturbation is applied strategically along sensitive attribute dimensions while preserving less-sensitive features; and (iii) a new architecture for enforcing DP whereby noise is applied in the forward phase, ensuring early-stage privacy control, rather than in the back-propagation step.

The rest of the paper is organized as follows: Section II introduces fundamental concepts and definitions. Section III presents the details of our proposed approach. We provide an extensive experimental evaluation against benchmarks in Section IV. Section V reviews related work. We conclude with directions for future work in Section VI. An extended version of our paper, including appendices with additional proofs and experimental settings details is available online at https://.

II. Background

A. DP-SGD and DP-Forward

Differential Privacy [5], [6] is a formal privacy framework that ensures the output of a computation does not significantly change when a single individual's data is added or removed.

Definition 1. (ε, δ)-Differential Privacy. A randomized mechanism M satisfies (ε, δ)-differential privacy if, for any pair of neighboring datasets $D \sim D'$ (differing in one record) and any subset of possible outputs S:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta$$

The parameter ε (privacy budget) controls the level of privacy: smaller values provide stronger privacy guarantees but may reduce utility [5]. The parameter δ allows for a small probability of failure in the privacy guarantee[7], [8]. A central DP concept is **sensitivity**, which measures the change in the output of a function when a single record is changed.

Definition 2. ℓ_2-Sensitivity. For a function $q : D \to \mathbb{R}^d$, the ℓ_2-sensitivity is defined as:

$$\Delta_2(q) = \sup_{D \sim D'} \|q(D) - q(D')\|_2$$

One widely-used method to achieve DP is the **Gaussian Mechanism** [7], [6], which adds normally distributed noise scaled to the sensitivity of the function.

Definition 3. Gaussian Mechanism. Let $f : D \to \mathbb{R}^d$ be a function with ℓ_2-sensitivity Δ_2. The mechanism

$$M(x) = f(x) + w \quad \text{where } w \sim \mathcal{N}(0, \sigma^2 I)$$

satisfies (ε, δ)-DP if

$$\sigma = \frac{\Delta_2 \sqrt{2 \ln(1.25/\delta)}}{\varepsilon}$$

The noise added is directly proportional to sensitivity and inversely proportional to ε.

Differentially Private Stochastic Gradient Descent (DP-SGD) adds DP noise to gradients.To bound ℓ_2 sensitivity, DP-SGD applies per-sample clipping, where the gradient norm is truncated to a constant C called clipping norm. Gaussian noise is then added to the clipped gradient [9].

A key challenge in DP-SGD is controlling the privacy budget consumption across multiple iterations. Abadi et al. introduced the *Moments Accountant* (MA) [7], a tight privacy accounting method that provides an accurate upper bound on the total privacy loss by analyzing the composition of Gaussian mechanisms under Poisson sampling. MA significantly reduces the accumulated privacy cost and enables practical training of deep models with differential privacy guarantees.

DP-Forward [10] is a recent training paradigm that performs DP sanitization in the forward pass of a neural network. DP-Forward adds noise either at the level of input features or their first-layer representation. However, similar to DP-SGD, DP-Forward requires noise to be added in every training iteration, which results in high privacy budget consumption.

B. Inverse Gaussian Distribution

The *Inverse Gaussian Distribution* (IGD) [11], [12] is a continuous probability distribution commonly denoted as IG(μ, λ), where $\mu > 0$ is the mean and $\lambda > 0$ is the shape parameter. The PDF of the IG distribution is given by:

$$f(x; \mu, \lambda) = \left(\frac{\lambda}{2\pi x^3}\right)^{1/2} \exp\left(-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right), \quad x > 0.$$

To capture heterogeneity in sensitivity across individual input attributes, we model personalized privacy budgets ε_i by drawing them from an Inverse Gaussian distribution $IG(\mu, \lambda)$.

This approach provides flexible control over the distribution of privacy budgets: the mean μ determines the expected privacy budget, while the variance μ^3/λ controls the spread. This allows flexible assignment of privacy budgets, enabling per-attribute control of the privacy-utility trade-off.

C. Layer-Wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) [13] provides a mechanism to trace the contribution of individual input features to a particular output decision. It does so by propagating a quantity called relevance backward through the network layers, starting from the output neuron corresponding to the predicted class. At each layer, the relevance values are redistributed to the neurons in the previous layer using specific propagation rules until the input layer is reached, at which point the resulting relevance scores indicate the contribution of each input feature to the model's decision. LRP offers a principled framework for identifying which features were most influential in generating a specific prediction.

Let the output of a neuron at layer l be denoted as z_i^l. The relevance score R_i^l for each neuron at layer l is distributed based on its contribution to the activations of the next layer:

$$R_i^l = \sum_j \frac{z_i^l w_{ij}}{\sum_k z_k^l w_{kj}} R_j^{l+1} \tag{1}$$

where w_{ij} represents the weight connecting neuron i in layer l to neuron j in layer $l+1$. The sum of relevance values remains constant, ensuring interpretability.

Although effective, LRP is inherently data-dependent, as it assigns relevance dynamically based on the specific input data. Different input samples result in different relevance distributions, which can lead to potential privacy concerns in sensitive applications. To address this, we introduce a data-independent alternative called Layer-Wise Contribution Propagation (LCP) discussed in Section II-C.

D. Matrix Variate Gaussian mechanism

The *Matrix Variate Gaussian (MVG) Mechanism* [4] extends the classical Gaussian Mechanism by introducing non-i.i.d. matrix-valued noise, particularly suitable for matrix-valued queries [10]. While conventional GM perturbs each scalar entry with i.i.d. noise, MVG samples noise $Z \in \mathbb{R}^{n \times d}$ from the zero-mean matrix-variate Gaussian distribution:

$$Z \sim \text{MN}_{n,d}(0, \Sigma, \Psi)$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is the row-wise covariance matrix, and $\Psi \in \mathbb{R}^{d \times d}$ is the column-wise covariance matrix.

The mechanism injects less noise into less sensitive parts of the data, increasing utility. The PDF of the matrix-variate Gaussian distribution is defined as:

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

where $\Sigma = UU^\top$, $\Psi = VV^\top$, and $\|\cdot\|_F$ denotes the Frobenius norm [4], [14]. When Σ and Ψ are diagonal and share equal values along the diagonal, the distribution simplifies to the standard i.i.d. Gaussian case [10].

Let $f(X) \in \mathbb{R}^{m \times n}$ be a matrix-valued query function. The L_2 sensitivity of f is defined as:

$$s_2(f) = \sup_{d(X_1, X_2)=1} \|f(X_1) - f(X_2)\|_F$$

[4] where $\|\cdot\|_F$ is the Frobenius norm. MVG perturbs the output of f by adding matrix-valued Gaussian noise:

$$\text{MVG}(f(X)) = f(X) + Z$$

where $Z \sim \text{MVG}_{m,n}(0, \Sigma, \Psi)$. MVG ensures (ε, δ)-DP if:

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

with the constants defined as: $\alpha = [H_r + H_{r,1/2}]\gamma^2 + 2H_r\gamma s_2^2(f)$, $\beta = 2(nd)^{1/4} H_r s_2^2(f)\zeta(\delta)$, $\zeta(\delta) = 2\sqrt{-nd\ln\delta} - 2\ln\delta + nd$ and $\gamma = \sup_X \|f(X)\|_F$. H_r and $H_{r,1/2}$ represent generalized harmonic numbers of order r and $1/2$, respectively.

III. Proposed Approach

A. System Architecture

The proposed architecture (Figure 1) consists of two parallel processing pipelines for input attributes. **Pipeline #1** is a conventional feedforward path through the neural network up to the penultimate layer, designated as the **sanitization layer**, similar to the work in [2]. For text classification, we use BERT and follow the conventional fine-tuning setup where the output of the BERT encoder is used to predict the class label. Specifically, we utilize the final hidden state corresponding to the special `[CLS]` token, which is designed to aggregate information from the entire input sequence. Let $\mathbf{h}_{\text{[CLS]}} \in \mathbb{R}^{768}$ denote this representation.

In the standard BERT classification pipeline, $\mathbf{h}_{\text{[CLS]}}$ is passed through a fully connected linear layer classifier. The classification head is parameterized by a weight matrix $W_c \in \mathbb{R}^{2\times d}$ (for binary classification) and a bias vector $\mathbf{b}_c \in \mathbb{R}^2$, where d is the input dimensionality to the classification head, which corresponds to the dimensionality of the output from the sanitization layer.

To support privacy-aware representation learning, we introduce a *sanitization layer* between the BERT encoder and the classification head. This layer is designed to project the rich contextualized representation from BERT into a lower-dimensional subspace, which can serve as a privacy bottleneck by filtering potentially sensitive information. The sanitization layer is implemented as a single linear transformation without any non-linearity. It takes the `[CLS]` token representation as input and outputs a vector of dimension d. Formally, the transformation is defined as:

$$\mathbf{z} = W_s \mathbf{h}_{\text{[CLS]}} + \mathbf{b}_s$$

where $\mathbf{h}_{\text{[CLS]}} \in \mathbb{R}^{768}$ is the output of the BERT encoder corresponding to the `[CLS]` token, $W_s \in \mathbb{R}^{d\times 768}$ is the projection matrix, $\mathbf{b}_s \in \mathbb{R}^d$ is the bias term, and $\mathbf{z} \in \mathbb{R}^d$ is the resulting *sanitized representation*.

To satisfy the bounded sensitivity requirement of differential privacy, as discussed in Section II-A, and to simplify noise calibration, we apply a *bounded ReLU activation* to the output of the sanitization layer. This clamps each scalar value in the resulting matrix of shape (batch size $\times$ d) to the range $[0, C]$. As a result, the sensitivity of each *per-output feature column* (i.e., each output dimension across the batch) is bounded by $\Delta_f = C$. This simplification enables direct application of independent noise to each column. A formal sensitivity analysis is provided in Appendix B in our extended version available online.

Pipeline #2 independently assigns privacy budgets to attributes, ensuring that more sensitive attributes receive stronger privacy protections. We compute the initial noise multipliers σ_i using the Rényi Differential Privacy (RDP) moments accountant, following the method of Abadi et al. [7]. To align these noise multipliers with the specific requirements of the sanitization layer, we introduce a **noise multiplier mapping mechanism**, which determines the final noise levels σ_s for each per-feature output from the sanitization layer.

Each column in the output of the sanitization layer corresponds to a feature representation, while each row represents an individual sample from the input batch. To facilitate selective noise injection across different features, we leverage the **directional noise property** of the **Matrix-Variate Gaussian (MVG) mechanism** [10], [4]. This technique allows for controlled noise magnitudes across different output dimensions.

B. Privacy Budget Assignment

An essential step in our approach is how to determine the amount of individual privacy budget for each input attribute. In traditional machine learning settings where inputs directly correspond to features, the sensitivity of each feature can be easily assessed. However, in scenarios where inputs are text tokens (e.g., LLM) or pixel values (e.g., CNN), choosing privacy budgets is more challenging. One possibility is to use the position of the token in the input to quantify sensitivity. Existing approaches for text data [15] consider that the central region of the input carries more sensitive information than the edges, thus necessitating stronger privacy protection. We employ **Inverse Gaussian Distribution mechanism (IGD)** [11] to generate budget values for different input tokens (we use the terms tokens and attributes interchangeably).

Specifically, we use four parameters for the IGD mechanism: k, v, $\varepsilon_{\min}$, and $\varepsilon_{\max}$, which control the distribution of privacy budgets across the sequence of input attributes. We define the *normalized token positions* p_i along a symmetric interval as follows:

$$p_i = \frac{2i}{L-1} - 1 - k, \quad i = 0, 1, \ldots, L-1 \tag{2}$$

where m denotes the sequence length, and k is a shift parameter that modulates the position of the peak privacy sensitivity.

The k value determines the location of the *token position with the lowest privacy budget* ε_{min}, corresponding to the

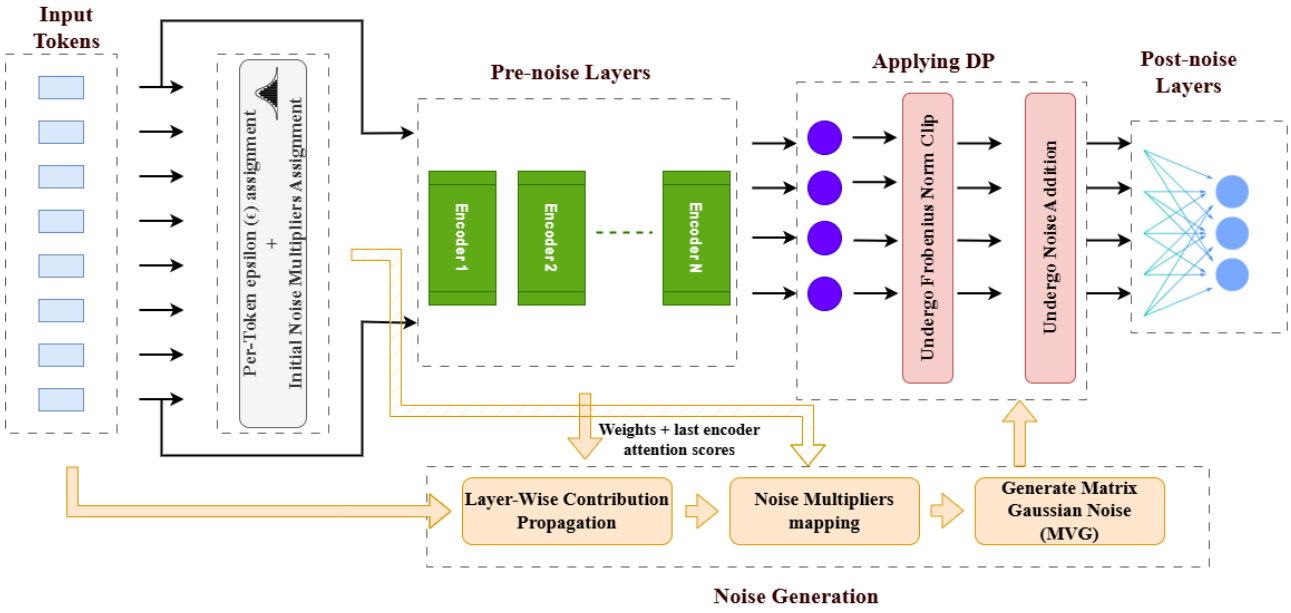


Fig. 1: BERT-based architecture of our approach

point in the sequence where privacy is most emphasized. A **negative** value of k shifts the inverse Gaussian peak to the left, thereby concentrating privacy protection more towards the **beginning of the sequence**. Conversely, a **positive** value of k shifts the peak to the right, focusing privacy towards the **later parts of the sequence**. Figure 2 shows how different choices of k affect the ε distribution.

To determine the budget amount for attributes, we first compute the Gaussian curve centered at zero:

$$G(p_i) = \exp\left(-\frac{p_i^2}{2v^2}\right) \tag{3}$$

where v controls the spread of the Gaussian distribution. Next, we **invert** the Gaussian function to align with privacy budget requirements, ensuring that lower ε values correspond to stronger privacy:

$$IG(p_i) = 1 - G(p_i) \tag{4}$$

Finally, we **normalize** the inverted Gaussian values to fit within the specified ε range $[\varepsilon_{\min}, \varepsilon_{\max}]$:

$$\varepsilon_i = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min}) \cdot \frac{IG(p_i) - \min(IG)}{\max(IG) - \min(IG)} \tag{5}$$

This ensures that ε_i's remain within the desired bounds. Once the privacy budgets are assigned, we employ the RDP moments accountant to compute the initial required noise multipliers σ_{i_s}.

C. Mapping noise multipliers to contribution scores

After computing σ_{i_s}, noise is applied to the outputs of the sanitization layer rather than directly to the original attributes. To achieve this, we introduce a methodology that maps σ_{i_s} to σ_{s_s} at the sanitization layer, based on the contribution scores C_{xy} of each input to each corresponding output.

Consider a system with n input attributes $X_1, X_2, \ldots, X_n$, each associated with a noise multiplier $\sigma_1, \sigma_2, \ldots, \sigma_n$. These inputs contribute to m output features $Y_1, Y_2, \ldots, Y_m$ through a linear transformation of the form:

$$Y_m = \sum_{i=1}^{n} C_{mi} X_i, \tag{6}$$

where C_{mi} is the contribution of attribute X_i to feature Y_m.

Since noise in the inputs propagates through the transformation, the resulting noise at the output level, σ_{out_m}, is:

$$\sigma_{out_m} = \sqrt{\sum_{i=1}^{n} C_{mi}^2 \sigma_i^2}. \tag{7}$$

This mapping ensures that the noise applied at the sanitization layer aligns with the noise distribution of the input attributes, thereby satisfying the privacy guarantees (proof in Appendix A in our extended version available online.). We focus next on how to quantify C_{xy}.

We introduce a data-independent algorithm called *Layer-Wise Contribution Propagation (LCP)* which quantifies the contribution of every input to each output of the network. It computes contribution scores for each neuron relying solely on model weights. Unlike LRP [13] (Section II-C), LCP does not use activations or neuron outputs, making it data-independent.

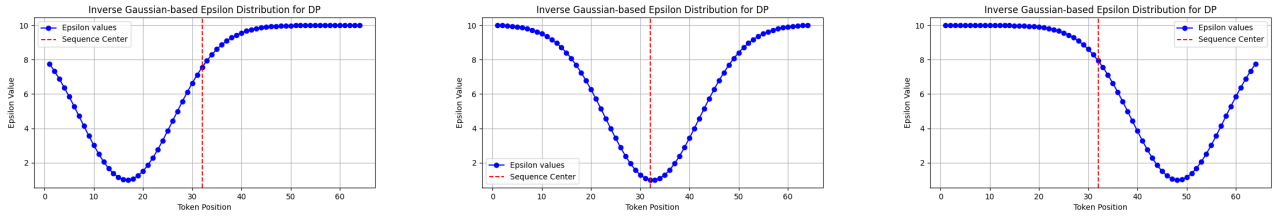
LCP consists of two passes:

Step 1: Forward Pass – Contribution Score Computation

The algorithm first computes layer-wise contribution scores r_{ij} to quantify how much each input contributes to an output neuron at each layer. These scores are computed as follows:

$$r_{ij} = \frac{|W_{ij}|}{\sum_{k=1}^{n} |W_{kj}|} \tag{8}$$

where W_{ij} is the weight connecting input i to output j, and the denominator performs normalization.



((a)) $k = -0.5$ ((b)) $k = 0$ ((c)) $k = 0.5$

Fig. 2: Different ε distributions with different values for k

Step 2: Backward Pass – Contribution Propagation

The precomputed contribution scores are propagated layer-by-layer from the output to the inputs using equation:

$$R_{xj}^{(l)} = \sum_i R_{xi}^{(l-1)} \cdot r_{ij}^{(l)} \tag{9}$$

where:

- $R_{xj}^{(l)}$ is the contribution score at neuron j in layer l.
- $R_{xi}^{(l-1)}$ represents the contribution score at neuron i in the previous layer $l-1$.
- $r_{ij}^{(l)}$ is the score computed in the forward pass.

Algorithm 1 Layer-Wise Contribution Propagation (LCP)

Require: Neural Network Model f, Model Weights W
  First Pass: Compute Contribution Scores
  for each layer $l = 1$ to L **do**
    for each input neuron i in layer l **do**
      for each output neuron j in layer l **do**
        $r_{ij} = \frac{|W_{ij}|}{\sum_{k=1}^{n} |W_{kj}|}$
      end for
    end for
  end for
  Second Pass: Contribution Propagation
  for each layer $l = L$ down to input layer **do**
    for each neuron x in layer l **do**
      for each output neuron j in layer l **do**
        $R_{xj}^{(l)} = \sum_i R_{xi}^{(l-1)} \cdot r_{ij}^{(l)}$
      end for
    end for
  end for
  Return: Contribution scores R

LCP is well-suited for architectures that rely on explicit weight matrices defining connections between neurons, such as fully connected layers, convolutional layers, and recurrent networks. In these cases, LCP can effectively quantify the contribution of each neuron to the output by leveraging the fixed weight matrices. However, when applied to transformer-based models such as BERT, LCP encounters challenges due to the dynamic nature of the self-attention mechanisms.

BERT employs a self-attention mechanism where each token attends to all other tokens in the sequence [16]. Unlike linear or convolutional layers, where contributions are determined by static weight matrices, the attention scores in BERT are dynamically computed based on the specific input. The contribution of an input token to an output token is determined by attention scores, which vary depending on the context of the input sentence. Another fundamental challenge arises from how attention scores are computed. Self-attention in transformers assigns importance to each token through attention weights, which are defined as softmax-normalized dot products between query and key vectors. The attention weight between tokens X_i and Y_j is computed as:

$$A_{ij} = \frac{\exp(Q_i K_j^T)}{\sum_k \exp(Q_i K_k^T)} \tag{10}$$

where Q and K are the query and key vectors, respectively [16]. Since attention scores are input-dependent, the relative influence of different tokens varies, preventing the use of a fixed contribution matrix as required by LCP.

As a solution, we average the attention scores output from the last encoder layer and propagate them through the remaining linear layers using LCP. This approach allows us to approximate the contribution of each token in a data-independent manner while leveraging the structured propagation capabilities of LCP for the feed-forward components of the transformer. However, using averaged attention scores from the last encoder layer and propagating them through the remaining linear layers introduces potential privacy concerns. Since attention scores are inherently data-dependent, averaging over multiple inputs retains statistical patterns from the training data, leading to privacy leakage. To mitigate this risk, we propose two alternative approaches:

1) **Applying Differential Privacy to Averaged Attention Scores:** An additional privacy budget ε_{att} is used to sanitize attention scores immediately before their use in LCP calculations. All subsequent computations preserve DP from the perspective of attention scores, effectively obfuscating individual token dependencies. This approach preserves privacy while still allowing structured contribution propagation. However, it introduces a trade-off: the mapping is performed on noisy contribution scores, which may reduce model precision.
2) **Using Fixed Positional Embeddings to Extract Attention Scores:** We can extract attention values from fixed positional embeddings. This ensures that attention weights are determined purely by token positions rather than input content, thereby eliminating privacy risks. The

advantage of this approach is that it provides highly accurate contribution scores based on token positions. The resulting attention scores can still encode how tokens at different positions interact in the model's architecture, which may lead to residual privacy risks if positional dependencies correlate with sensitive training patterns.

D. Modified MVG mechanism

As discussed in Section II-D, the Matrix-Variate Gaussian (MVG) Mechanism, introduced by Chanyaswad et al. [4], is designed to perturb matrix-valued outputs by adding structured noise following a matrix-variate Gaussian distribution, as opposed to traditional i.i.d. Gaussian noise [17], [10].

When the function $f(\mathbf{x})$ produces an output in $\mathbb{R}^{M\times N}$, instead of applying independent Gaussian noise to each element, the MVG mechanism perturbs the entire matrix using a noise matrix sampled from a matrix-variate Gaussian distribution. This approach accounts for the underlying structure of the data and introduces correlations between different noise elements.

Matrix-variate Gaussian noise, in its vectorized form, can be sampled from a multivariate Gaussian distribution with a structured full-rank covariance matrix:

$$\Xi = \Sigma \otimes \Psi \in \mathbb{R}^{MN\times MN} \tag{11}$$

where Σ and Ψ define the row-wise and column-wise covariance structures, respectively, and $\otimes$ represents the Kronecker product. Unlike the classical Gaussian Mechanism (GM), which assumes independent noise components, the MVG mechanism results in a covariance matrix Ξ with nonzero off-diagonal elements, meaning the additive noise components are not mutually independent.

The MVG formulation ensures that noise injection respects the structure of the matrix-valued output, leading to a more effective privacy-utility tradeoff than traditional i.i.d. noise mechanisms. Furthermore, the property of applying less noise to more important elements aligns well with our approach. This intuition allows us to selectively assign different noise magnitudes to different rows or columns of the 2D matrix.

We perturb the column covariance matrix Ψ **per feature (i.e., per column)**. Specifically, we construct Ψ as a diagonal matrix:

$$\Psi = \text{diag}(\sigma_{s_1}^2, \sigma_{s_2}^2, \ldots, \sigma_{s_d}^2) \tag{12}$$

where each element σ_i^2 represents the noise variance assigned to the corresponding feature. This structured perturbation allows us to effectively distribute noise based on the relative importance of different features while maintaining an identity structure for the row covariance matrix $\Sigma = I_n$. The identity row covariance ensures that noise remains uncorrelated across different samples while enforcing controlled dependencies across different features via Ψ. The final noise matrix Z is:

$$Z \sim MN_{n,d}(0, I_n, \Psi) \tag{13}$$

By leveraging this approach, we introduce a refined privacy-preserving mechanism that minimizes information loss for significant features while ensuring differential privacy guarantees. This structured noise application allows us to achieve a balanced tradeoff between privacy protection and model utility, outperforming conventional isotropic noise injection methods.

IV. Experimental Evaluation

Experimental Testbed. We briefly review our experimental testbed. For a detailed description please consult Appendix C in our extended version available online. We use the Stanford Sentiment Treebank (SST-2) [18], part of the GLUE benchmark, a widely used benchmark for binary sentiment classification. SST-2 consists of 67,349 training examples and 872 examples in the development set. We exclude neutral sentiments to simplify the task to a binary setting. Each instance contains a single sentence, and the classification is based on overall sentiment polarity. We use the development set as our test set, since the official test set is publicly available without labels. All inputs are preprocessed using the BERT tokenizer with a maximum sequence length of 128 tokens.

Model Characteristics. An exhaustive description of model characteristics is included in Appendix D in our extended version available online.. We adopt BERT-base-uncased [16] as our backbone encoder. It provides a strong performance baseline while maintaining a reasonable computational footprint, making it a practical choice for privacy-preserving training. By building upon BERT-base, we ensure comparability with prior work in both standard NLP and differential privacy for text, where BERT-base remains the most commonly used architecture due to its balance of accuracy and efficiency. We initialize BERT-base using pretrained weights from Hugging Face's Transformers library [19] and fine-tune it end-to-end on the SST-2 dataset. The model processes input sentences using WordPiece tokenization, prepending a [CLS] token and appending a [SEP] token as required by the original BERT design. The final hidden representation of the [CLS] token is used for downstream classification [16].

Model Parameters. We use a fixed training batch size B over E fine-tuning epochs. Learning rate is initialized to η_i. As training progresses under the Hugging Face Trainer [19], which utilizes the AdamW optimizer, η is automatically adapted by the optimizer's learning rate scheduling and warm-up mechanisms.

Input sequences of length m are fed into BERT (shorter sequences are padded and longer ones truncated). We assess the impact of input length using a range of $m \in \{16, 32, 64\}$, with default value 64. We fix parameter $\delta = 10^{-5}$. The privacy budget ranges from $\varepsilon_{\min} = 1.0$ (highest privacy level applied for most sensitive tokens) to $\varepsilon_{\max} = 10$ (lowest privacy level).

For the IGD mechanism parameters, we define a fixed ε variance of $v = 0.3$. To control the position of the privacy peak along the input sequence, we vary the shift parameter $k \in \{-0.5, 0, 0.5\}$, which determines where the maximum privacy budget (i.e., $\varepsilon_{\min}$) is applied. A negative value of k shifts the privacy peak toward the beginning of the sequence (left), zero centers it in the middle, and a positive value shifts it toward the end (right).

Sanitization layer width d (i.e., output dimensionality of the projection layer placed between the encoder and the classification head) is varied in the range $d \in \{200, 500, 700, 900\}$.

A. Results and Analysis

We use two comparison benchmarks: (i) **DP-Forward** [10] introduces a privacy-preserving representation learning technique by directly perturbing embedding matrices in the forward pass of language models. This approach satisfies stringent local differential privacy requirements for both training and inference data; (ii) **ANADP** [20] proposes an adaptive individual differential privacy mechanism by assigning heterogeneous noise magnitudes to each model parameter. These two baselines offer complementary strengths: one focuses on structured intermediate representation perturbation, while the other provides fine-grained, per-parameter adaptivity.

As stated in Section III-C, our approach incorporates three distinct settings related to our part of LCP mechanism, each differing in how attention scores are handled.

- **RAW:** In this setting, we directly utilize the raw attention scores produced by the pre-trained model without any additional perturbation, which leads to privacy leakage.
- **ATTEN_DP:** This variant applies an additional differential privacy mechanism to the output attention scores, with a fixed privacy budget of $\varepsilon_{\text{att}} = 1.0$.
- **POSITION:** We input to the BERT model positional embeddings only, without token embeddings. This leads the model to produce attention scores that are purely position-based, thereby discarding content-sensitive information.

For protecting labels, we rely on the **LabelDP** mechanism [21], [22], [23], which was originally introduced in [21] and later adopted in the DP-Forward framework [10].

First, we vary the value of the shift parameter k to observe how the model performance responds to changes in the position of the privacy peak along the input sequence. The privacy allocation across the sequence is governed by the IGD mechanism discussed in section III-B. Varying k shifts the peak of this distribution, allowing us to emphasize different regions of the input (e.g., beginning, middle, or end) with higher privacy protection. Sanitization layer width is fixed to $d = 700$.

Figure 3 illustrates the results for several values of k and input sequence lengths m. Model performance remains largely constant across different values of k. Shifting the privacy peak toward the left, middle, or right of the input sequence does not lead to significant degradation in accuracy. The largest drop in performance is observed under the `ATTEN_DP` mechanism with $m = 64$, where the accuracy decreases from 91.5% when $k = 0$ to 90.8% when $k = 0.5$. Despite this, the model maintains strong predictive performance, indicating robustness to shifts in the privacy emphasis across the sequence.

A more pronounced effect on accuracy is observed for input sequence length m across values $\{16, 32, 64\}$. Consistent with observations reported in the DP-Forward work [10], increasing the input sequence length generally leads to improved model performance. This suggests that longer input sequences enable the model to access more contextual information, which helps compensate for the information loss introduced by the privacy mechanisms.

Figure 4 shows model accuracy results under varying sanitization layer widths $d \in \{200, 500, 700, 900\}$ across the three configurations: `RAW`, `ATTEN_DP`, and `POSITION`. The results indicate that the choice of sanitization width has a moderate effect on model performance.

In contrast to prior work [2], where the sensitivity of the sanitization mechanism was scaled proportionally to $\sqrt{d}$, our findings show that increasing the width does not lead to a substantial drop in accuracy. Our results suggest that the model remains robust across a wide range of sanitization dimensions. This is particularly evident in the `POSITION` configuration, where accuracy remains stable regardless of the width.

In the `RAW` setting, the model achieves its highest accuracy at $d = 200$, with a slight drop as width increases, followed by a small recovery at $d = 900$. Similarly, for `ATTEN_DP`, the best performance is observed at $d = 500$, with accuracy remaining relatively stable across other widths. The `POSITION` configuration exhibits more consistent performance, with minimal variation across different values of d.

Overall, the results suggest that while the sanitization width introduces a bottleneck in the representation, the model remains robust across a broad range of dimensionalities.

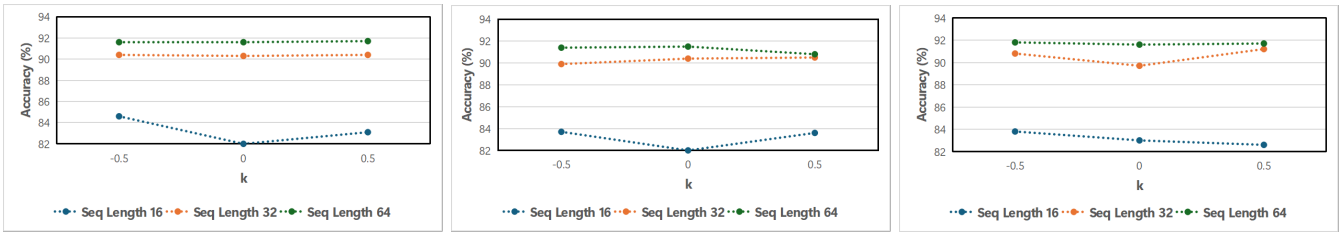
Figure 5 presents the training loss progression across all values of d over training steps. The model with $d = 700$ exhibits a smoother loss decay, despite having a higher loss overall compared to the other configurations. In contrast, widths $d = 200$, 500, and 900 achieve lower final training losses, but their curves show greater fluctuation and instability.

An important trade-off emerges: while different widths may allow the model to better fit the training, they also introduce more variance into the optimization process. Width $d = 700$ appears to offer a more balanced configuration, leading to a stable and consistent learning trajectory.

This behavior correlates with the fact that the encoder outputs a 768-dimensional representation. The sanitization layer acts as a linear projection from this 768-dimensional space to a lower-dimensional bottleneck of size d. When d is much smaller (e.g., 200), this projection aggressively compresses the BERT representation, possibly discarding informative features. On the other hand, when d is significantly larger (e.g., 900), the layer approaches a near-identity transformation, potentially retaining redundant or noisy features.

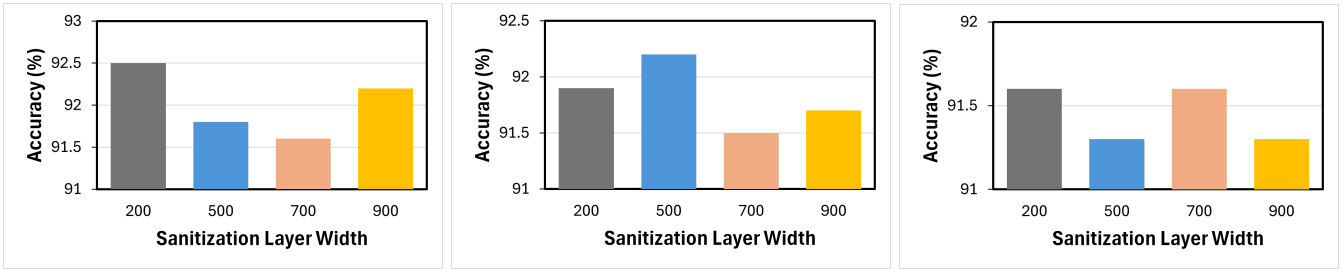
At $d = 700$, the projection dimensionality is close to that of the original BERT output. This near-alignment likely allows the sanitization layer to retain most of the semantic information while introducing mild compression. Additionally, the projection matrix at this width may be better conditioned, leading to smoother gradient flow and more stable optimization under differential privacy. The results suggest that dimensional compatibility between BERT and the sanitization layer contributes to healthier learning dynamics.

Table I shows the accuracy under different privacy shift settings (controlled via the parameter k), with ε ranging from



((a)) RAW mechanism ((b)) ATTEN_DP mechanism ((c)) POSITION mechanism

Fig. 3: Model performance against different k settings with different m



((a)) RAW mechanism ((b)) ATTEN_DP mechanism ((c)) POSITION mechanism

Fig. 4: Accuracy for several sanitization layer widths (d) across three privacy settings: `RAW`, `ATTEN_DP`, and `POSITION`.

Fig. 5: Training loss curves for different sanitization layer widths ($d = 200, 500, 700, 900$).

10 to 1. Our results indicate that even in the strictest privacy setting, the accuracy of our method remains stable across all variants, with RAW achieving 91.7%, ATTEN-DP 90.8%, and POSITION 91.7%. Notably, these results substantially outperform DP-SGD, which reports 86.5% accuracy at comparable privacy levels, and ANADP, which achieves 88.18% with $\varepsilon \sim 8$. Furthermore, the DP-Forward paper reports 89.5% at $\varepsilon \sim 2$. This demonstrates that our approach of combining directional MVG noise mechanism with token-level privacy allocation preserves significantly more utility, even under strong privacy guarantees. The robustness of the POSITION variant across all k values also highlights the benefit of aligning noise direction with structural priors.

Next, using fixed values of $k = 0$ and $d = 700$, each configuration was evaluated under three different privacy budgets: low ($\varepsilon \in [1, 3]$), medium ($\varepsilon \in [7, 10]$) and high

TABLE I: Accuracy (%) under different DP mechanisms and privacy shift settings (k).

Method	**ε Range**	**Accuracy (%)**		
		$k = -0.5$	$k = 0$	$k = 0.5$
RAW	$10 \rightarrow 1$	91.6	91.6	91.7
ATTEN-DP	$10 \rightarrow 1$	**91.4**	**91.5**	**90.8**
POSITION	$10 \rightarrow 1$	**91.8**	**91.6**	**91.7**
DP-Forward [10]	~ 2	89.5		
ANADP [20]	~ 8	88.18		
DP-SGD [7], [10]	~ 2	86.5		
Non-Private baseline		93		

($\varepsilon \in [13, 15]$). Higher values of ε lead to weaker privacy guarantees and lower noise levels, yielding better accuracy. This trend was most evident in the **RAW** and **ATTEN_DP** configurations, where accuracy steadily improved or remained stable as ε increased. For instance, in the RAW configuration, accuracy increased from 91.89% at low ε ($[1 \rightarrow 3]$) to 92.20% at medium ε ($[7 \rightarrow 10]$), and remained high at 92.09% in the high ε range ($[13 \rightarrow 15]$). Similar improvements were observed in the ATTEN_DP setting, culminating in the highest accuracy of 92.66% at the highest ε range. As expected, when noise decreases, the model is able to recover more precise representations, thereby achieving higher utility.

TABLE II: Accuracy (%) under different ε for each model.

Method	**ε Range**		
	$[1 \rightarrow 3]$	$[7 \rightarrow 10]$	$[13 \rightarrow 15]$
RAW	91.89	92.20	92.09
ATTEN_DP	91.86	91.86	**92.66**
POSITION	91.40	**91.97**	90.70

B. Non-Monotonic Behavior and Regularization Role of DP

Despite the overall trend, we also identified non-monotonic patterns in specific configurations—most notably in the **POSITION** setting. In this case, the model achieved its highest accuracy of 91.97% at medium ε, but performance declined at both lower and higher privacy budgets. This indicates that the relationship between privacy and utility is not strictly monotonic, particularly when additional architectural components interact with the DP mechanism.

A key factor in this behavior is the presence of a *bounded ReLU activation* in the sanitization layer, which constrains the output to a fixed interval $[0, C]$. This activation serves as a nonlinear bottleneck, causing feature saturation when activations hit the upper or lower bounds. When ε is high (i.e., noise is small), the model can overfit to these saturated outputs, leading to poor generalization. In contrast, moderate noise levels introduced at medium ε injects enough stochasticity to regularize learning, mitigate overfitting, and improve robustness—similar to the effects of dropout or label smoothing.

Thus, in certain setups, DP noise provides a beneficial **regularization effect**. At low ε, noise is too strong and obscures useful signal, leading to underfitting. At high ε, the absence of noise combined with bounded activations may result in overfitting. Intermediate ε values can strike the right balance, allowing meaningful signal retention while still benefiting from the regularization induced by stochasticity.

These insights suggest that the role of DP noise extends beyond privacy protection—it can also serve as a mechanism for implicit regularization, especially in architectures where output representations are tightly bounded or compressed.

To further evaluate the robustness and generalizability of our proposed approach, we conducted an extended set of experiments under a fixed privacy configuration, with the privacy peak position set to $k = 0$ and sanitization layer width $d = 700$. We tested the model under all three proposed privacy settings—**RAW**, **ATTEN_DP**, and **POSITION**—across two transformer-based architectures: **BERT-base** and **DistilBERT**. In addition, we evaluated two distinct fine-tuning paradigms: **full fine-tuning**, and **LoRA** (Low-Rank Adaptation), to assess the compatibility of our privacy mechanism with parameter-efficient training strategies. For comparison, we also report results for non-private training using the same setups. Table III summarizes the performance across these configurations, showing that our approach maintains strong utility under all privacy settings while aligning well with both full and efficient fine-tuning paradigms.

TABLE III: Accuracy (%) across architectures and fine-tuning paradigms (with $k = 0$, $d = 700$).

Model		Private?	RAW	ATTEN_DP	POSITION
BERT-base	Full	Yes	91.6	91.5	91.6
		No	93		
	LoRA	Yes	87.8	88.1	88.3
		No	88.76		
DistilBERT	Full	Yes	89.6	89.3	88.4
		No	89.2		
	LoRA	Yes	86.7	86.6	86.8
		No	87.04		

V. Related Work

The Matrix Gaussian Mechanism (MGM - Section II-D) was introduced in [14]. Bu et al. proposed the Improved Matrix Gaussian Mechanism (IMGM) [24], which tightens further the theoretical privacy bounds. It derives a necessary and sufficient condition for (ε, δ)-DP by constraining the singular values of the noise covariance matrix, enabling tighter composition bounds. Zhang et al. introduced the Matrix Variate Gaussian (MVG) mechanism [4], which extends to matrix-valued queries using noise drawn from a matrix-variate Gaussian distribution. The novelty of this work lies in the concept of directional noise.

Personalized Differential Privacy (PDP) was introduced by Jorgensen et al. [25]. Subsequent approaches, like those by Li et al. [26], introduced privacy-aware and utility-based partitioning strategies that group data with similar privacy budgets for separate DP processing. More recently, adaptive solutions such as AdaPDP [27] dynamically tune noise and sensitivity parameters based on data distributions, user-defined privacy levels, and utility considerations. Two dominant PDP strategies are *sampling*—where high-privacy data is sampled less often—and *grouping*, where similar privacy requirement records are processed together [28], [3]. These ideas have been extended in the PATE framework [29], [30], using upsampling and teacher weighting to control information flow based on privacy levels.

LMO-DP [31] introduces a new framework for differentially private fine-tuning by generalizing the Moment Accountant via Rényi Differential Privacy, allowing for tight privacy composition across a broad range of mechanisms beyond Gaussian—including Laplace, Staircase, Matrix Variate Gaussian, and optimal mixture distributions. LMO-DP approximates sub-optimal DP mechanisms offline through a structured search space of probability density functions, incorporating randomized scale parameters drawn from mixtures of Gamma, Exponential, and Uniform distributions. The result is a strict ε-DP guarantee (with $\delta = 0$), over 90% accuracy at $\varepsilon = 0.3$, and significantly faster convergence in various tasks. Moreover, LMO-DP is orthogonal to existing techniques and integrates smoothly with methods such as Ghost Clipping for low-memory execution.

DP-MLM [32], focuses on privacy-preserving text rewriting rather than model fine-tuning. DP-MLM leverages masked language models (MLMs) such as BERT to perform text rewriting guided by contextual information from the original input. This method differs from traditional decoder-based privatization by relying on masked token prediction, enabling rewrites that maintain semantic coherence while preserving privacy. The contextualization strategy introduced in DP-MLM significantly improves utility under tight privacy budgets, outperforming previous state-of-the-art methods in empirical evaluations across standard benchmarks.

DP-ZO [33] introduced zeroth-order optimization (ZO) [34] for private fine-tuning of large language models. Unlike

gradient-based approaches, DP-ZO estimates gradients indirectly using only scalar loss differences between perturbed model evaluations with flipped signs. This scalar—being the only part derived from private data—is privatized by adding noise to the loss difference. The sensitivity of this update is controlled through clipping, enabling both ε-DP and (ε, δ)-DP guarantees. By removing the need for per-example gradient clipping, DP-ZO achieves strong performance across privacy levels, scales effectively to large models (e.g., 30B and 66B parameters), and simplifies implementation by eliminating the need for backpropagation.

VI. Conclusion

We introduced a forward-pass differential privacy framework for large language models, applying Matrix-Variate Gaussian noise and individualized per-attribute privacy budgets via a novel Layer-Wise Contribution Propagation (LCP) method. Our approach preserves feature correlations and achieves stronger privacy-utility trade-offs than DP-SGD and DP-Forward, as validated on SST-2. However, the method incurs significant computational overhead due to structured noise generation and contribution tracking. Future work will focus on optimizing computational efficiency to support broader scalability without compromising privacy guarantees.

References

[1] B. C. Das, M. H. Amini, Y. Wu, Security and privacy challenges of large language models: A survey, ACM Comput. Surv. 57 (6) (Feb. 2025).

[2] I. A. Monir, G. Ghinita, Differentially-private neural network training with private features and public labels, in: Big Data Analytics and Knowledge Discovery, 2024, pp. 208–222.

[3] F. Boenisch, C. Mühl, A. Dziedzic, R. Rinberg, N. Papernot, Have it your way: Individualized privacy assignment for DP-SGD, Advances in Neural Information Processing Systems 36 (2024).

[4] T. Chanyaswad, A. Dytso, H. V. Poor, P. Mittal, Mvg mechanism: Differential privacy under matrix-valued query, in: ACM Conf. on Computer and Communications Security, 2018, pp. 230–246.

[5] C. Dwork, Differential privacy, in: International Colloquium on Automata, Languages and Programming, Vol. 4052, 2006, pp. 1–12.

[6] C. A. Joseph P. Near, Programming Differential Privacy, 2023.

[7] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proc. of ACM Conf. on Computer and Communications Security, 2016, p. 308–318.

[8] I. Mironov, Rényi differential privacy, in: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), 2017, pp. 263–275.

[9] J. Yang, Z. Ji, L. Xiang, Weights shuffling for improving dpsgd in transformer-based models, arXiv e-prints (2024) arXiv–2407.

[10] M. Du, X. Yue, S. S. Chow, T. Wang, C. Huang, H. Sun, Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass, in: ACM CCS, 2023, pp. 2665–2679.

[11] M. C. Tweedie, Statistical properties of inverse gaussian distributions. i, The Annals of Mathematical Statistics 28 (2) (1957) 362–377.

[12] J. L. Folks, R. S. Chhikara, The inverse gaussian distribution and its statistical application—a review, Journal of the Royal Statistical Society Series B: Statistical Methodology 40 (3) (1978) 263–275.

[13] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, K.-R. Müller, Layer-wise relevance propagation: an overview, Explainable AI: interpreting, explaining and visualizing deep learning (2019) 193–209.

[14] J. Yang, L. Xiang, J. Yu, X. Wang, B. Guo, Z. Li, B. Li, Matrix gaussian mechanisms for differentially-private learning, IEEE Transactions on Mobile Computing 22 (2) (2021) 1036–1048.

[15] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, P. Liang, Lost in the middle: How language models use long contexts, Transactions of the Association for Computational Linguistics (2024).

[16] S. Ravichandiran, Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT, Packt Publishing Ltd, 2021.

[17] T. Ji, P. Li, Less is more: Revisiting the gaussian mechanism for differential privacy, in: USENIX Security Symposium, Philadelphia, PA, 2024, pp. 937–954.

[18] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.

[19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Scao, S. Gugger, A. Rush, Transformers: State-of-the-art natural language processing, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[20] X. Li, R. Zmigrod, Z. Ma, X. Liu, X. Zhu, Fine-tuning language models with differential privacy through adaptive noise allocation (10 2024). doi:10.48550/arXiv.2410.02912.

[21] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, C. Zhang, Deep learning with label differential privacy, Advances in neural information processing systems 34 (2021) 27131–27145.

[22] M. Malek, I. Mironov, K. Prasad, I. Shilov, F. Tramèr, Antipodes of label differential privacy: Pate and alibi, ArXiv (2021).

[23] A. M. Medina, R. I. Busa-Fekete, U. Syed, S. Vassilvitskii, On the pitfalls of label differential privacy, in: NeurIPS 2021 Workshop, 2021.

[24] J. Yang, L. Xiang, W. Li, W. Liu, X. Wang, Improved matrix gaussian mechanism for differential privacy, ArXiv abs/2104.14808 (2021).

[25] Z. Jorgensen, T. Yu, G. Cormode, Conservative or liberal? personalized differential privacy, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 1203–1217.

[26] H. Li, L. Xiong, Z. Ji, X. Jiang, Partitioning-based mechanisms under personalized differential privacy, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Vol. 10234, 2017, p. 615.

[27] B. Niu, Y. Chen, B. Wang, Z. Wang, F. Li, AdaPDP: Adaptive personalized differential privacy, in: IEEE INFOCOM, IEEE, 2021, pp. 1–10.

[28] I. A. Monir, M. I. Fauzan, G. Ghinita, A review of adaptive techniques and data management issues in dp-sgd, Data Engineering (2023) 94.

[29] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, Ú. Erlingsson, Scalable private learning with PATE, ICLR 1050 (2018) 24.

[30] F. Boenisch, C. Mühl, R. Rinberg, J. Ihrig, A. Dziedzic, Individualized PATE: Differentially private machine learning with individual privacy guarantees, Proceedings on Privacy Enhancing Technologies (2023).

[31] Q. Yang, M. Mohammady, H. Wang, A. Payani, A. Kundu, K. Shu, Y. Yan, Y. Hong, LMO-DP: optimizing the randomization mechanism for differentially private fine-tuning (large) language models, CoRR abs/2405.18776 (2024).

[32] S. Meisenbacher, M. Chevli, J. Vladika, F. Matthes, Dp-mlm: Differentially private text rewriting using masked language models, CoRR (2024).

[33] X. Tang, A. Panda, M. Nasr, S. Mahloujifar, P. Mittal, Private fine-tuning of large language models with zeroth-order optimization, in: ICML 2024 Workshop on Foundation Models in the Wild, 2024.

[34] J. C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE transactions on automatic control 37 (3) (1992) 332–341.

[35] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, et al., Differentially private fine-tuning of language models, in: ICLR, 2025.

[36] D. Yu, H. Zhang, W. Chen, J. Yin, T.-Y. Liu, Large scale private learning via low-rank reparametrization, in: International Conference on Machine Learning, PMLR, 2021, pp. 12208–12218.

[37] X. Yue, M. Du, T. Wang, Y. Li, H. Sun, S. S. Chow, Differential privacy for text analytics via natural text sanitization, in: ACL-IJCNLP 2021, Association for Computational Linguistics (ACL), 2021, pp. 3853–3866.

[38] B. Ghojogh, A. Ghodsi, Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey, working paper or preprint (Dec. 2020).

APPENDIX

A. Noise Propagation Theorem

Theorem: Given independent Gaussian noise applied to input attributes, the propagated noise at the output follows a Gaussian distribution where the output noise standard deviation is derived from the squared sum of input noise variances weighted by the squared contribution scores [2].

Proof: Let X_1, X_2, X_3 be independent random variables with noise $\mathcal{N}(0, \sigma_1^2), \mathcal{N}(0, \sigma_2^2), \mathcal{N}(0, \sigma_3^2)$. The output features Y_1, Y_2 are defined as:

$$Y_1 = C_{11}X_1 + C_{12}X_2 + C_{13}X_3 \tag{14}$$

$$Y_2 = C_{21}X_1 + C_{22}X_2 + C_{23}X_3 \tag{15}$$

Since each X_i contains Gaussian noise, the noise component in Y_1 is:

$$C_{11}\mathcal{N}(0, \sigma_1^2) + C_{12}\mathcal{N}(0, \sigma_2^2) + C_{13}\mathcal{N}(0, \sigma_3^2) \tag{16}$$

Using the property that a linear combination of independent Gaussian random variables remains Gaussian, the variance at the output is computed as:

$$\sigma_{out1}^2 = C_{11}^2\sigma_1^2 + C_{12}^2\sigma_2^2 + C_{13}^2\sigma_3^2 \tag{17}$$

Similarly, for Y_2:

$$\sigma_{out2}^2 = C_{21}^2\sigma_1^2 + C_{22}^2\sigma_2^2 + C_{23}^2\sigma_3^2 \tag{18}$$

Taking the square root, we obtain the standard deviation:

$$\sigma_{out1} = \sqrt{C_{11}^2\sigma_1^2 + C_{12}^2\sigma_2^2 + C_{13}^2\sigma_3^2} \tag{19}$$

$$\sigma_{out2} = \sqrt{C_{21}^2\sigma_1^2 + C_{22}^2\sigma_2^2 + C_{23}^2\sigma_3^2} \tag{20}$$

Thus, the theorem is proven.

B. Sensitivity Analysis of Bounded ReLU

Let $Z \in \mathbb{R}^{B\times d}$ denote the output of the sanitization layer, where B is the batch size and d is the output dimensionality. We apply an element-wise bounded ReLU activation to each scalar entry z_{ij} of Z, defined as:

$$\text{BoundedReLU}(z_{ij}) = \min(\max(z_{ij}, 0), C)$$

This operation ensures that each individual output feature is clamped to the range $[0, C]$. The result is a matrix where all scalar values lie within a fixed interval, providing an upper bound on the influence any single data point can have on the output.

We consider neighboring datasets D and D' that differ in a single example, which corresponds to a single row change in the output matrix Z. Since the bounded ReLU activation is applied independently to each scalar entry, any change in the output due to a differing input affects only one row of the matrix.

Each scalar entry z_{ij} in Z is guaranteed to lie within $[0, C]$. Therefore, the maximum possible change in any scalar output across neighboring datasets is:

$$|z_{ij} - z'_{ij}| \leq C$$

As a result, for each output feature (i.e., each column of Z), only a single coordinate (corresponding to one row) may differ, and the maximum ℓ_2 norm of the change in that column vector is:

$$\|z_j - z'_j\|_2 = |z_{ij} - z'_{ij}| \leq C$$

Hence, the ℓ_2 sensitivity of each per-output feature column is:

$$\Delta_f = C$$

This per-column sensitivity bound allows for independent application of differential privacy noise (e.g., Gaussian noise) to each output dimension using standard mechanisms.

C. Dataset

We conduct our experiments using the Stanford Sentiment Treebank (SST-2) [18], a widely used benchmark for binary sentiment classification. SST-2 is part of the General Language Understanding Evaluation (GLUE) benchmark suite, which comprises a collection of tasks that are widely adopted in both natural language processing (NLP) and DP research [10], [35], [36], [37].

SST-2 focuses on binary sentiment classification of movie review sentences, where each example is labeled as either positive or negative. The dataset consists of 67,349 training examples and 872 examples in the development set. Unlike the original SST dataset, neutral sentiments are excluded in SST-2 to simplify the task to a binary setting.

Each instance contains a single sentence, and the classification is based on overall sentiment polarity. This task formulation aligns well with sentence-level privacy-preserving representations and downstream evaluation of private text encoders.

Evaluation Protocol. Following standard practice in the GLUE benchmark, we use the development set as our test set, since the official test set is publicly available without labels. We report classification accuracy as the primary evaluation metric.

All inputs are preprocessed using the BERT tokenizer with a maximum sequence length of 128 tokens. Inputs are truncated or padded as needed. For consistency with prior work, we retain the original tokenization strategy without additional preprocessing.

D. Model Architecture & Parameters

1) Model Architecture: We build our models on top of the Bidirectional Encoder Representations from Transformers (BERT) [16] architecture, a widely used pretrained language model.BERT is designed using a multi-layer bidirectional transformer encoder trained on large-scale corpora using a masked language modeling (MLM) objective and a next sentence prediction (NSP) task.

BERT is available in multiple configurations, differing primarily in depth, hidden dimensionality, and number of attention heads [16], [38]:

- BERT-base: 12 layers (transformer blocks), 768 hidden units, 12 attention heads, 110M parameters.

- BERT-large: 24 layers, 1024 hidden units, 16 attention heads, 340M parameters.
- DistilBERT-base: 6 layers, 768 hidden units, 12 attention heads, 66M parameters (lighter, distilled from BERT, no NSP objective).
- RoBERTa-base: 12 layers, 768 hidden units, 12 attention heads, 125M parameters (trained with dynamic masking and no NSP).
- RoBERTa-large: 24 layers, 1024 hidden units, 16 attention heads, 355M parameters (improved pretraining and trained on 10× more data than BERT).

These variations offer trade-offs between model capacity, training cost, and inference latency. Table IV shows summary of different configurations.

TABLE IV: BERT configurations.

Model	Layers	Hidden Units	Attention Heads	Parameters
BERT-base	12	768	12	110M
BERT-large	24	1024	16	340M
DistilBERT-base	6	768	12	66M
RoBERTa-base	12	768	12	125M
RoBERTa-large	24	1024	16	355M

In this work, we adopt BERT-base-uncased as our backbone encoder. It provides a strong performance baseline while maintaining a reasonable computational footprint, making it a practical choice for privacy-preserving training, where additional computational cost is incurred by privacy mechanisms such as gradient clipping or noise injection.

We initialize BERT-base using pretrained weights from Hugging Face's Transformers library and fine-tune it end-to-end on the SST-2 dataset. The model processes input sentences using WordPiece tokenization, prepending a [CLS] token and appending a [SEP] token as required by the original BERT design. The final hidden representation of the [CLS] token is used for downstream classification [16].

By building upon BERT-base, we ensure comparability with prior work in both standard NLP and differential privacy for text, where BERT-base remains the most commonly used architecture due to its balance of accuracy and efficiency.

2) Parameters: We organize the parameters used in our experiments into three main categories:

- **Training Parameters:** This includes the standard model training hyperparameters such as the learning rate, batch size, number of fine-tuning epochs, and the optimizer used.
- **Differential Privacy Parameters:** Represented by the pair (ε, δ), where δ is fixed and ε is not a single value but instead varies within a range. Specifically, ε is defined by a minimum and maximum bound, denoted as $\varepsilon_{\min}$ and $\varepsilon_{\max}$, respectively.
- **IGD Parameters:** These govern the generation of the ε range. The values of $\varepsilon_{\min}$ and $\varepsilon_{\max}$ are dynamically derived from an Instance-wise Gradient Decay (IGD) mechanism parameterized by k and v.

For the training parameters, we define a fixed training batch size B along with a total number of fine-tuning epochs E. The learning rate η is initialized to η_i. As training progresses under the Hugging Face Trainer [19], which implicitly utilizes the AdamW optimizer, η is automatically adapted. This adaptivity is handled internally by the optimizer's learning rate scheduling and warmup mechanisms.

We also define an input sequence length to BERT, denoted by m, where input sequences shorter than m are padded and those exceeding m are truncated. In the majority of our experiments, m is fixed at 64. However, to assess the impact of input length on model performance, we also evaluate the model using $m \in \{16, 32, 64, 128\}$ and analyze its behavior across these configurations.

Going further with the DP parameters, we fix the failure probability at $\delta = 10^{-5}$. The privacy budget bounds are set to $\varepsilon_{\min} = 1.0$, representing the highest privacy level (applied at the peak of the privacy concerns), and $\varepsilon_{\max} = 10$, representing the lowest privacy level. This configuration enables adaptive allocation of privacy budgets across the training process as governed by the IGD mechanism.

Continuing with the IGD parameters, we define a fixed ε variance of $v = 0.3$. To control the position of the privacy peak along the input sequence, we vary the shift parameter $k \in \{-0.5, 0, 0.5\}$, which determines where the maximum privacy budget (i.e., $\varepsilon_{\min}$) is applied. A negative value of k shifts the privacy peak toward the beginning of the sequence (left), zero centers it in the middle, and a positive value shifts it toward the end (right). This setup allows us to investigate the effect of privacy emphasis across different parts of the sequence.

The last parameter we define is the sanitization layer width, denoted by d, which represents the output dimensionality of the projection layer placed between the encoder and the classification head. In our experiments, we vary $n \in \{200, 500, 700, 900\}$ to study the impact of the sanitization bottleneck size on downstream task performance and privacy-utility trade-offs.

For benchmarking with the DP-Forward paper [10], we used the same values for all common parameters to ensure a fair and consistent comparison.

TABLE V: Summary of experimental parameters.

Category	Parameter	Values
Training Parameters	Batch size (B)	32
	Epochs (E)	3
	Optimizer	$AdamW$
	Learning rate (η)	$\eta_i = 20^{-5}$ (Adaptive)
	Sequence length (m)	16, 32, 64 (default)
DP Parameters	δ	10^{-5}
	$\varepsilon_{\min}$	1.0 (highest privacy)
	$\varepsilon_{\max}$	10
IGD Parameters	Variance (v)	0.3
	Shift (k)	-0.5 , 0 , 0.5
Sanitization Layer	Width (d)	200, 500, 700, 900