



# PrunePrivyTune: Accelerating Language Models with Pruning and Differentially Private Fine-Tuning

Sonakshi Garg<sup>1,2</sup> · Vicenç Torra<sup>1</sup>

Received: 27 February 2025 / Revised: 1 January 2026 / Accepted: 11 February 2026  
© The Author(s) 2026

## Abstract

Large Language Models (LLMs) have demonstrated exceptional capabilities in language understanding and generation, but their large-scale architecture poses significant challenges in deployment and inference, such as increased computational demands and slower processing times. While various techniques like model pruning, knowledge distillation, and quantization have been developed to compress LLMs, they often result in task-specific compression, limiting the model's versatility. Additionally, LLMs face privacy risks due to their potential to memorize and reproduce sensitive training data, raising concerns when deployed in real-world applications. To address these challenges, we propose a novel methodology *PrunePrivyTune* that combines efficient model compression with privacy preserving fine-tuning. Our approach leverages pairwise cosine similarity to identify redundant layers in transformer models, enabling structural pruning that reduces model size without compromising performance. After pruning, we apply Low-Rank Adaptation (LoRA) with DPSGD to fine-tune the model. This ensures that fine-tuning process is both efficient and privacy-preserving, outperforming training and preventing the model from memorizing sensitive data. Later on, we generated synthetic data using the fine-tuned model and subsequently conducted a training data extraction attack to assess the model's privacy vulnerabilities, in terms of perplexity and BERTScore. Our framework demonstrates that the proposed methodology effectively reduces the inference time through model compression and pruning compliments privacy, followed by private fine-tuning. Additionally, our privacy risk assessment indicates that integrating DP successfully mitigates the risk of the model's memorization. This approach upholds strong privacy guarantees, making it highly suitable for real-time applications and deployment in sensitive domains where data confidentiality is paramount.

**Keywords** Model compression · Pruning · Differential privacy · Low-rank adaptation · Training data extraction attack

---

Editors: Yonggang Zhang, Zhen Fang, Mengyue Yang, Kaifeng Lyu, Hangyu Li, Sharon Li, Kun Zhang.

---

Extended author information available on the last page of the article

## 1 Introduction

Recently, Large Language Models (Devlin, 2018; Thoppilan et al., 2022; Touvron et al., 2023) have shown exceptional skills in understanding and generating language. As these models grow in size, they become more capable of handling complex tasks (Brown, 2020; Chowdhery et al., 2023) and even display new, unexpected abilities. However, despite their impressive performance, LLMs present challenges when it comes to deployment and inference. Their large scale architecture requires significant computational resources, leading to issues like longer processing times and other related concerns. For example, when performing tasks like next-word prediction or sentence completion, we need the response to be generated in microseconds for practical use. However, the large size of the model significantly slows down the inference time, making it less efficient for real-time applications. Various techniques have been proposed to address these problems, including model pruning (Hanson & Pratt, 1988), knowledge distillation (Buciluă et al., 2006), and quantization (Gong et al., 2014; Wu et al., 2016).

While these methods have been successful in reducing the number of parameters in the models without significantly affecting performance, they often focus on compressing models for specific domains or tasks, known as task-specific compression. For example, a pre-trained language model might be fine-tuned on a specific dataset, such as one from the GLUE benchmark (Wang et al., 2018) for classification tasks, and then distilled into a smaller model focused on that single task. However, applying this approach to compress LLMs could limit their versatility, making them suitable only for one task rather than being effective across a broad range of tasks.

LLMs, like many neural network architectures also face privacy challenges due to their ability to memorize and reproduce parts of their training data (Carlini et al., 2019, 2021). This characteristic stems from the way these models are trained, which can retain and inadvertently reproduce specific data sequences, particularly when exposed to sensitive information during training. In LLMs, techniques like prompt engineering can exploit this memorization, potentially extracting confidential or personally identifiable information from the model, which poses a critical risk when models are deployed in real-world applications.

In Natural Language Understanding (NLU), protecting the privacy of users' sensitive data is paramount, as input text or its vector representations can inadvertently expose private information (Coavoux et al., 2018; Li et al., 2018). To address this critical concern, Differential Privacy (DP) is employed during the training process to safeguard individual data points from being disclosed. By integrating DP, noise is added to the model's training procedure, ensuring that the influence of any single data point is minimal and thus reducing the risk of data leakage. Specifically, DP can be implemented through the use of the DPSGD optimizer, which modifies the standard Stochastic Gradient Descent (SGD) by clipping gradients to limit the sensitivity of the loss function and adding calibrated noise to these gradients. This approach effectively obscures the contribution of individual data points, making it difficult to infer private information from the model's outputs. As a result, service providers can train NLU models while maintaining strong privacy guarantees, ensuring that users' sensitive information remains protected throughout the training process. This not only enhances user trust but also helps avoid potential legal and reputational risks associated with data breaches.

Even with DPSGD applied during training, language models still face privacy challenges, such as vulnerability to inference-time attacks (Shokri et al., 2017) where sensitive information can be inferred from model outputs. There's also a trade-off between privacy and model utility, as the noise added to protect privacy can degrade performance. Additionally, repeated queries can weaken the privacy guarantees, and fine-tuning large models while maintaining strong privacy protections is also challenging, potentially leading to unintended data exposure. Current approaches may not fully eliminate the risk of data leakage when the model is queried in specific ways or when fine-tuned on sensitive datasets, highlighting the need for continued advancements in privacy-preserving techniques. Thus, the proposed methodology addresses the following challenges.

### 1.1 Complexity of Optimal Pruning

Determining which weights or neurons to prune is complex. Strategies like magnitude-based pruning, where weights with small magnitudes are removed, might not always capture the most important parameters. More sophisticated techniques, such as those based on sensitivity analysis or learned pruning, require additional computational resources and can be more difficult to implement.

### 1.2 Memorization of Training Data

Language models often exhibit a tendency to memorize sensitive training data, which can lead to privacy breaches and non-compliance with GDPR regulations. This memorization risk undermines data privacy by potentially exposing personal information, thereby necessitating effective strategies to mitigate such privacy leakage in compliance with regulatory standards.

### 1.3 Overhead of Post-pruning

Following model pruning, it is often necessary to retrain or fine-tune the model to recover from potential performance degradation. While private fine-tuning—ensuring that this process maintains privacy—is not extensively explored in existing literature, this paper addresses this gap by investigating and developing effective methods for privacy-preserving fine-tuning after pruning.

Therefore, the primary aim of this work is to design a unified framework that enables efficient and privacy-preserving compression of LLMs, without sacrificing their general-purpose capabilities.

To tackle the aforementioned challenges of model compression by pruning while preserving users' privacy, we propose our methodology: *PrunePrivyTune*. Our approach involves computing pairwise cosine similarities across all layers, rather than limiting the analysis to consecutive layers, to perform structural pruning and eliminate redundant layers from the model. After pruning, we employ LoRA (Low-Rank Adaptation) for fine-tuning, combined with Differential Privacy-SGD (DPSGD). LoRA is a parameter-efficient technique that updates only a subset of the model's parameters, significantly accelerating the fine-tuning process compared to full retraining. Concurrently, DPSGD introduces noise to gradients, ensuring that sensitive data is not memorized during fine-tuning, thereby incorporating

privacy-preserving measures. We evaluate the model's performance on several datasets, focusing on accuracy. Subsequently, we generate synthetic data from the fine-tuned model and conduct a training data extraction attack by comparing the original and synthetic texts. We assess the model's privacy vulnerabilities using perplexity and BERTScore metrics.

Therefore, this paper introduces *PrunePrivyTune*, a novel methodology that integrates structural model pruning with privacy-preserving fine-tuning. Unlike prior approaches, our method performs global structural pruning by computing pairwise cosine similarities across all layers to identify redundancy, rather than relying on local or magnitude-based heuristics, and combines LoRA-based parameter-efficient fine-tuning with Differentially Private SGD, addressing privacy risks introduced during post-pruning adaptation—an aspect largely unexplored in existing literature.

The main objectives of the proposed work are as follows.

- O1: To develop a global structural pruning approach based on pairwise-cosine similarity for identifying redundant layers, and to compare its effectiveness with traditional pruning strategies.
- O2: To evaluate whether LoRA-based fine-tuning can achieve comparable or superior post-pruning recovery to full retraining, while significantly reducing computational cost and training overhead.
- O3: To integrate Differential Privacy into the fine-tuning stage and assess its impact on model utility and resistance to data leakage relative to non-private baselines.
- O4: To generate synthetic data from the compressed and differentially private models and evaluate its utility and privacy properties, including potential leakage and similarity to the original training data.

The main contributions of the paper are as follows.

- To propose a novel methodology: *PrunePrivyTune* that integrates structural pruning with differentially private LoRA fine-tuning to enhance model efficiency while preserving privacy.
- To utilize pairwise cosine similarity to identify redundant layers in transformer models and apply pruning based on varying sparsity levels.
- To provide a comprehensive comparison of our approach with other model compression techniques, such as knowledge distillation, and benchmark against existing baselines.
- To assess the privacy vulnerabilities of the trained model, we conduct a training data extraction attack.
- To demonstrate that our framework effectively reduces model size while also mitigating the risk of the model's memorization of sensitive data. This results in a more compact model that retains performance while significantly lowering the risk of data leakage.

The remaining part of the paper is organized as follows. Section 2 describes the existing works related to our approach. Section 3 discusses the important concepts that are used. Section 4 presents the proposed methodology. Section 5 depicts the experimental setup. Section 6 discusses the results obtained and Sect. 7 presents the conclusion.

## 2 Related Works

In this section, we discuss the existing studies that are closely related to our work.

### 2.1 Model Compression

Language models have garnered significant attention in recent years, leading to an increased focus on reducing the size of their parameters and inference time. To address these challenges, various model compression techniques have been explored, which can be broadly categorized into network pruning (Hanson & Pratt, 1988), knowledge distillation (Buciluă et al., 2006), quantization (Gong et al., 2014; Wu et al., 2016), and other methods like early exit strategies (Xin et al., 2020) or dynamic token reduction (Ye et al., 2021). Our emphasis is on pruning, particularly structural pruning, which involves the removal of entire filters from the neural network, offering better hardware efficiency. Structural pruning can be achieved through various approaches, including l1-based pruning (Han et al., 2015), first-order importance estimation (Hou et al., 2020), hessian-based techniques (Kurtic et al., 2022), and the optimal brain surgeon method. Different works have been proposed using different unit of pruning, such as entire layers, multi-head attention, or feed-forward layers. Pruning can also be thought of as analyzing the notion of redundancy in transformer models, which could be layer-level redundancy or neuron-level redundancy (Dalvi et al., 2020). LLM-Pruner (Ma et al., 2023) computes the importance of channel-wise weights to perform structural pruning and then fine-tunes the pruned model using LoRA. However, each channel may contain crucial information and pruning them can degrade the performance. Also, they didn't consider if any sensitive data was used in training, and its privacy implications. Sparse-GPT (Frantar & Alistarh, 2023) performs unstructured pruning on weights, while compensating for weights that are not pruned. They performed one-shot pruning, but many research works demonstrate that fine-tuning with LoRA saves computational time and efficiency.

### 2.2 Privacy-Preserving Compression

PATE (Papernot et al., 2018) performed knowledge distillation in which an ensemble of teacher model is trained on disjoint private data and the student model is trained by noisy aggregation of teacher's response. However, its performance is inferior to DPSGD for complex datasets (Yu et al., 2021). Mireshghallah et al. (2022) studied pruning and knowledge-distillation model compression techniques with the notion of privacy as well. They found that pruning results into better model accuracy than knowledge distillation. Garg and Torra (2024) studied task-specific knowledge distillation with DP, achieved better model performance than (Mireshghallah et al., 2022), but the approach is task-specific and it could not be generalized. We have compared our approach with both (Garg & Torra, 2024; Mireshghallah et al., 2022) showing that pruning performs better than distillation.

## 3 Preliminaries

In this section we explain the most important concepts that are used in this paper.

### 3.1 Pruning

Pruning (Zhu & Gupta, 2017) is a popular model compression technique that focuses on reducing the size and computational complexity of neural networks by removing less significant parameters or entire components. The primary goal of pruning is to make models more efficient in terms of memory usage, inference time, and energy consumption, while maintaining similar performance levels. Unlike quantization, which reduces the precision of the weights (e.g., converting from 32-bit to 8-bit), or distillation, which transfers knowledge from a larger model to a smaller one, pruning specifically eliminates redundant parts of the model. This approach can be particularly advantageous in scenarios where storage and computational resources are limited, such as deploying models on edge devices or mobile platforms. Pruning offers a more interpretable and flexible method of compression, as it directly removes less important connections or structures from the network, potentially leading to a more straightforward trade-off between efficiency and accuracy compared to quantization or distillation. Pruning is favored for having a balance between accuracy and model size.

Different pruning strategies exist, each suited to different stages of the model life cycle. Unstructured pruning (Kwon et al., 2020) removes individual weights based on their magnitude, leading to sparse matrices but can be harder to optimize in hardware. Structured pruning (Wang et al., 2019), on the other hand, removes entire components such as neurons, filters, or attention heads, resulting in a smaller, denser network that is easier to accelerate. Pruning can be done before training (pre-training pruning), during training (gradual pruning), or after training (post-training pruning). The choice of when to prune often depends on the model's performance and the specific application requirements. Methods like magnitude-based pruning (Lee et al., 2020) remove parameters with the smallest absolute values, assuming they contribute least to the output. Gradient-based pruning (Yeom et al., 2021) uses information from the gradients to determine which parameters are least important for the model's performance. Pruning often involves iterative cycles of pruning and retraining to recover any lost performance, ensuring the pruned model remains effective while being significantly more efficient.

The pre-train, prune, and fine-tune paradigm (Xu et al., 2021) is a model optimization strategy where a large model is first pre-trained on a vast dataset to learn general representations. After pre-training, pruning is applied to remove redundant or less important parameters, reducing the model's size and complexity. Finally, the pruned model is fine-tuned on a specific task or dataset to recover any lost performance and further adapt the model to the target application. This approach balances the benefits of large-scale pre-training with the efficiency gains from pruning, making it a practical method for deploying resource-efficient models.

### 3.2 Differential Privacy

Differential Privacy (DP) (Dwork, 2006) is an industry standard privacy model for ensuring data anonymization.

**Definition 1** ( $\epsilon, \delta$ )-Differential Privacy: Consider two datasets as neighboring if they differ by only one record (either by the addition or removal of a single data point). A mechanism

$A$  is said to be  $(\epsilon, \delta)$ -differentially private if, for any two neighboring datasets  $D$  and  $D'$ , and for any subset  $S$  of the output range of  $A$ , the following inequality holds:

$$P[A(D) \in S] \leq \exp(\epsilon) \times P[A(D') \in S] + \delta. \quad (1)$$

Here,  $\epsilon$  and  $\delta$  control the strength of the privacy guarantee, with smaller values providing stronger privacy. A key property of differentially private mechanisms is that any post-processing (data-independent transformation) of their output remains differentially private with the same privacy guarantees.

### 3.2.1 Differential Privacy in ML

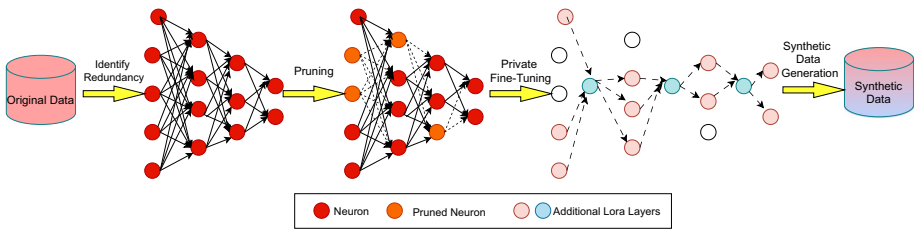
In machine learning, DP can be introduced at various stages, such as at the input level, during model training (DP-Training), or when serving predictions (model inference) (Ponomareva et al., 2023). For non-convex loss functions, one of the most effective DP-Training methods is gradient-noise injection, such as Differentially Private Stochastic Gradient Descent (DP-SGD) (Abadi et al., 2016). This approach limits the sensitivity of the loss function by clipping per-example gradients and adding Gaussian noise to the aggregated clipped gradients to ensure privacy. The noise level is tied to the clipping norm (which controls sensitivity) and the strength of the  $\epsilon$  guarantee. A similar strategy could be adopted to other optimizers as well.

## 4 Methodology

In this section we describe the proposed methodology: *PrunePrivyTune*. It is divided into three parts: efficient pruning, private fine-tuning and then data synthesis using the fine-tuned model. In our methodology, pruning is conceptualized as the removal of redundant layers in the network, guided by pairwise cosine similarity computations. Initially, hidden states are extracted, followed by the calculation of pairwise cosine similarity between these states. Pruning is then executed based on the average pairwise cosine similarity. To mitigate the potential loss of information due to pruning, the model is subsequently either retrained or fine-tuned. Given that retraining can be computationally intensive, we opt for fine-tuning the pruned model using DPSGD. The fine-tuned model is then employed to generate synthetic text. Later on, training data extraction attack is performed to analyze the privacy vulnerabilities and compared the original dataset with the synthetic dataset generated. The methodology is diagrammatically depicted in Fig. 1 with step by step explanation described in the next sub-sections.

### 4.1 Pruning

The objective of pruning is to minimize the redundancy in the model while maintaining similar performance. The idea is that if two consecutive layers are very similar, one of them might be redundant and can be pruned. This method emphasizes incremental knowledge changes across layers. Let  $L$  be the set of all layers in the model. For each pair of consecutive layers  $(\ell, \ell + 1)$ , the cosine similarity is denoted by  $\text{cos-sim}(x^{(\ell)}, x^{(\ell+1)})$ , where  $x^{(\ell)}$



**Fig. 1** PrunePrivyTune: an approach for efficient pruning and private fine-tuning

denotes the  $\ell$ -th hidden state. The objective function is to minimize the sum of cosine similarities for the pruned layers, subject to the constraint that the pruned model maintains a performance metric  $\mathcal{P}$  within a threshold  $\Delta$  of the original model’s performance  $\mathcal{P}_0$ . Formally, the optimization problem can be written as:

$$\begin{aligned} \min_{S \subseteq \mathcal{L}} \quad & \sum_{\ell \in S} \frac{1}{2} (\cos\text{-sim}(x^{(\ell)}, x^{(\ell+1)}) + \cos\text{-sim}(x^{(\ell)}, x^{(\ell-1)})) \\ \text{subject to} \quad & \mathcal{P}(S) \geq \mathcal{P}_0 - \Delta \end{aligned} \tag{2}$$

where  $S$  is the set of pruned layers,  $\mathcal{P}(S)$  is the performance of the model after pruning the layers in  $S$ ,  $\mathcal{P}_0$  is the original model’s performance and  $\Delta$  is the allowable degradation in performance. The step by step explanation of the proposed methodology for pruning is found in the next sub sections.

### 4.1.1 Extract Hidden States

LLM mainly use a transformer architecture, which is made up of several transformer encoder-decoder layers. The effect of each layer can be analyzed as a transformation of the previous hidden state. These layers follow a residual structure, meaning each layer modifies the input it receives. Let us suppose that for each  $\ell$  layer  $f$ , and the parameters  $\theta^{(\ell)}$ , the impact of layer  $f$  can be written as:

$$x^{(\ell+1)} = x^{(\ell)} + f(x^{(\ell)}, \theta^{(\ell)}) \tag{3}$$

In this equation, layer  $f$  adds a transformation  $f(x^{(\ell)}, \theta^{(\ell)})$  to the input  $x^{(\ell)}$ . Therefore, we can understand the importance of each layer in LLMs by looking at how much it changes the input hidden states.

### 4.1.2 Compute Pairwise Cosine Similarity

For each pair of layers, with states  $(x, x')$ , the cosine similarity is computed as follows:

$$\cos\text{-sim}(x, x') = \frac{\sum_{i=1}^L \sum_{j=1}^d x_{i,j} \cdot x'_{i,j}}{\sqrt{\sum_{i=1}^L \sum_{j=1}^d (x_{i,j})^2} \cdot \sqrt{\sum_{i=1}^L \sum_{j=1}^d (x'_{i,j})^2}} \tag{4}$$

### 4.1.3 Identification of Redundant Layers

We define the *redundancy metric* to evaluate the redundancy of each layer. To do this, we compute pairwise cosine similarity of each layer with its adjacent layers. For a given layer  $\ell$ , the average cosine similarity is calculated as follows.

$$\text{avg\_cos}_\ell = \frac{1}{2} (\text{cos-sim}(x^{(\ell-1)}, x^{(\ell)}) + \text{cos-sim}(x^{(\ell)}, x^{(\ell+1)})) \quad (5)$$

The average cosine similarity quantifies how similar the transformations are between a given layer and its adjacent layers. A higher value indicates that the layer has a higher degree of similarity with its neighbors, suggesting potential redundancy.

Then, we define *threshold-based pruning*. To determine which layers to prune, set a threshold value  $\tau$  for the average cosine similarity. Layers with an average similarity above this threshold are considered redundant.

$$\text{redundant layers} = \{\ell \mid \text{avg\_cos}_\ell > \tau\} \quad (6)$$

where  $\tau$  is the predefined threshold value. Layers that meet this criterion are candidates for pruning, as they are likely to be less critical in terms of unique transformation and might be contributing to redundancy in the model.

### 4.1.4 Prune Redundant Layers

In this step, the pruning procedure is performed by removing the redundant layer based on the computed similarity and adjusting the connections of the preceding and succeeding layers. The proposed pruning method can be executed by following the steps described above, as detailed in Algorithm 1.

## 4.2 Private Fine-Tuning

Once the model is pruned, it needs to be either re-trained or fine-tuned to recover from the information loss and potential performance degradation. Fine-tuning is typically preferred over re-training because it leverages the pre-existing learned features, making it more efficient and faster while often yielding better performance with less computational cost. Numerous methods have been developed to update pre-trained models without modifying all the model's weights. In this work, we focus on LoRA (Low-rank Adaptation) (Hu et al., 2021). It works by keeping all the pre-trained model weights fixed and adding trainable low-rank decomposition matrices within each dense layer (MLP and Attention). This approach reduces the number of trainable parameters compared to full fine-tuning.

However, fine-tuning carries the risk of memorizing the training data, as models can learn and store specific data points. To mitigate this risk, DP can be employed, which helps in reducing the potential for data memorization. We utilized DP-LoRA tuning to privately fine-tune the pruned model on downstream tasks. LoRA is particularly effective because it updates only a small subset of parameters compared to full fine-tuning, making it computationally efficient. Instead of updating the entire weight matrix  $W$ , LoRA introduces low-rank

matrices  $L$  and  $R$  such that the updated weight matrix becomes  $W + LR$ . During fine-tuning, only the matrices  $L$  and  $R$  are updated, significantly reducing the number of trainable parameters and enhances computational efficiency especially when combined with DPSGD. In previous work on private training of LLMs, the entire set of weights was adjusted during private training, a process that is both computationally intensive and often yields suboptimal results when used with DP-SGD, as explained step-by-step in Algorithm 2. Therefore, fine-tuning with DP is generally more effective than re-training with DP. Furthermore, there is limited research on performing private fine-tuning with LoRA, which underscores the novelty and potential advantages of this approach.

### 4.3 Data Synthesis Using the Fine-Tuned LLM

Given that transformer models are known for their potential to memorize training data, applying DP is crucial to mitigate risks of memorization. After fine-tuning the model with DP, we generate synthetic data to evaluate the privacy risks associated with the fine-tuned model. We evaluate whether the model memorizes its training data and can regenerate it or not.

While autoregressive language models are commonly used for text generation, recent advancements have demonstrated the effectiveness of masked language models (MLMs) like BERT in this context as well. In this approach, the model generates text by predicting masked tokens, and is refined iteratively to improve output quality. Specifically, masked language models predict masked tokens in the input text and then iteratively re-mask and refine the predictions for least likely tokens (Ghazvininejad et al., 2019). This technique has shown promising results in machine translation and other NLP tasks.

Applying a similar methodology, we leverage a fine-tuned MLM for synthetic data generation using the following approach. This approach for generating synthetic data involves leveraging a fine-tuned masked language model. The model is adapted to handle MLM tasks, where certain tokens in the input text are masked and the model's objective is to predict these masked tokens. The process begins by selecting input sentences from the downstream task, where a portion of the words are randomly masked, prompting the model to predict the missing tokens. The model predicts these masked tokens iteratively, using a strategy called top-k sampling to introduce diversity by selecting one of the top-k most probable tokens for each masked position. This iterative refinement continues until all masked tokens are replaced, resulting in a fully generated sentence. By repeating this process across numerous sentences, a large and varied synthetic dataset is created, which can be used to enhance training data for various natural language processing tasks. We evaluate the synthetically generated dataset with the real dataset with the help of training data extraction attack.

---

**Require:** Model  $M$ , DataLoader  $D$ , Threshold  $\tau$ , Performance Function  $P$ , Max Degradation  $\Delta$

```

1: Extract Hidden States( $M, D$ ):
2:
3:   Initialize hidden states  $H = []$ 
4:   for each batch in  $D$ :
5:      $x \leftarrow \text{batch}[\text{input\_ids}]$ 
6:      $\text{outputs} \leftarrow M(x, \text{output\_hidden\_states}=\text{True})$ 
7:      $\text{hidden\_states} \leftarrow \text{outputs\_hidden\_states}$ 
8:     Append  $\text{hidden\_states}$  to  $H$ 
9:   end for
10:  Return  $H$ 
11: Compute Cosine Similarity( $H$ ):
12:  Initialize pairwise cosine similarity  $C = []$ 
13:  for sample in  $H$ 
14:    for  $\ell = 1$  to  $L - 1$ 
15:      Append  $\text{cos-sim}(x^{(\ell)}, x^{(\ell+1)})$  to  $C[\ell]$ 
16:    end for
17:  end for
18:  Return  $C$ 
19: Identify Redundant Layers( $C, \tau$ ):
20:  Initialize  $\text{redundant\_layers} = \emptyset$ 
21:  for  $\ell = 1$  to  $L$ 
22:     $\text{avg\_cos}_\ell = \frac{1}{2} (\text{cos-sim}(x^{(\ell-1)}, x^{(\ell)}) + \text{cos-sim}(x^{(\ell)}, x^{(\ell+1)}))$ 
23:    If  $\text{avg\_cos}_\ell > \tau$  then
24:      Add  $\ell$  to  $\text{redundant\_layers}$ 
25:    end if
26:  end for
27:  Return  $\text{redundant\_layers}$ 
28: Prune Layers( $M, C, \tau, P, \Delta$ ):
29:  Initialize  $P_0 \leftarrow P(M)$ , pruned layers  $S = []$ 
30:  for  $\ell = 1$  to  $L - 1$ 
31:    If  $C[\ell] > \tau$  then
32:      Prune layer  $\ell + 1$  from  $M$ , append to  $S$ 
33:       $P_{\text{new}} \leftarrow P(M)$ 
34:      If  $P_{\text{new}} < P_0 - \Delta$  then
35:        Restore layer  $\ell + 1$  to  $M$ , remove from  $S$ , BREAK
36:      end if
37:    end if
38:  end for
39:  Return  $M, S$ 
40: main():
41:   $H \leftarrow \text{Extract Hidden States}(M, D)$ 
42:   $C \leftarrow \text{Compute Cosine Similarity}(H)$ 
43:   $\text{redundant\_layers} \leftarrow \text{Identify Redundant Layers}(C, \tau)$ 
44:  Return  $\text{Prune Layers}(M, C, \tau, P, \Delta)$ 

```

---

**Algorithm 1** Pruning redundant layers in a model

## 5 Experimental Setup

In this section, we present the experimental setup which includes description of the dataset used, different baselines that we used for comparison, privacy analysis and risk assessment of our approach.

### 5.1 Dataset Description

We first outline the datasets and tasks used in our experiments. We selected different tasks from the GLUE benchmark (Wang et al., 2018), a widely recognized evaluation benchmark for assessing the performance of natural language understanding models in NLP. Our experiments focused on the following tasks: SST-2 (Single-Sentence Sentiment Classification), RTE (Recognizing Textual Entailment), MRPC (Microsoft Research Paraphrase Corpus), and CoLA (Corpus of Linguistic Acceptability). These tasks were chosen to represent a diverse set of challenges, varying in the size of their training datasets. For example, SST-2 has the largest training set with 67K examples, while RTE has the smallest, with just 2K examples, providing a comprehensive range for evaluation in our experiments. The datasets used in this work are already pre-divided into training, validation, and test sets, and we used these splits directly without further partitioning.

---

**Require:** Weight matrix  $W$ , Low-rank matrices  $L$  and  $R$ ,  $\eta$ , Noise scale  $\sigma$ , Clipping norm  $C$ , Dataset  $\mathcal{D}$ ,  $\epsilon$

- 1: Initialize  $L$  and  $R$  with small random values
- 2: **for** each minibatch  $B \subset \mathcal{D}$  **do**
- 3:     **Compute gradients for  $L$  and  $R$ :**

$$\nabla L_B, \nabla R_B \leftarrow \frac{1}{|B|} \sum_{i \in B} \nabla L_i, \nabla R_i$$

- 4:     **Clip gradients to norm  $C$ :**

$$\nabla L_B \leftarrow \frac{\nabla L_B}{\max(1, \frac{\|\nabla L_B\|_2}{C})}, \quad \nabla R_B \leftarrow \frac{\nabla R_B}{\max(1, \frac{\|\nabla R_B\|_2}{C})}$$

- 5:     **Add noise to the gradients:**

$$\tilde{\nabla} L_B \leftarrow \nabla L_B + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}), \quad \tilde{\nabla} R_B \leftarrow \nabla R_B + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$$

- 6:     **Update  $L$  and  $R$  using noisy gradients:**

$$L \leftarrow L - \eta \tilde{\nabla} L_B, \quad R \leftarrow R - \eta \tilde{\nabla} R_B$$

- 7:     **end for**

- 8:     **Return** updated weight matrix  $W + LR$
- 

**Algorithm 2** DPSGD LoRA fine-tuning

## 5.2 Model Architecture

The proposed model is based on the BERT-base architecture, consisting of an embedding layer followed by a stack of Transformer encoder blocks and a task-specific classification head. Initially, the model contains 12 encoder layers, with a hidden dimension of 768 followed by a task-specific classification head. To enable structural compression, hidden representations from all layers are extracted and analyzed using pairwise cosine similarity. Layers exhibiting high average similarity across the network are considered redundant and removed based on a predefined sparsity level.

Unlike conventional pruning approaches that focus on magnitude-based heuristics or adjacent-layer comparisons, our method performs global layer-wise similarity analysis. This means that it allows for the identification of structurally redundant components throughout the network. This results in a reduced-depth Transformer while preserving representational capacity.

After pruning, the compressed model is fine-tuned on the described dataset using our introduced private LoRA based technique. Differential Privacy is enforced through gradient clipping and the addition of Gaussian noise to gradients, following the DP-SGD paradigm. This mitigates memorization of sensitive training data during post-pruning adaptation.

In the proposed framework, structural pruning reduces the number of transformer encoder layers, thereby decreasing the total parameter count. During fine-tuning, all remaining parameters of the pruned model are updated using differentially private SGD, while no additional parameters are introduced. Consequently, the number of trainable parameters is reduced compared to the original model due to layer removal, whereas the non-trainable parameters correspond to the pruned (removed) components of the original architecture.

The computational cost of the proposed pruning strategy is dominated by the computation of pairwise cosine similarities between transformer layers. Given  $L$  layers and a hidden dimension  $d$ , this step scales as  $O(L^2d)$  since the number of layers in transformer models is relatively small, this cost is negligible compared to training or fine-tuning, making the proposed pruning approach computationally efficient.

## 5.3 Baselines

We conducted a comprehensive comparison of our approach with several baseline methods. We compared our approach with state-of-the-art BERT-base model (Devlin, 2018) which comprises of twelve transformer layers, with BERT-small model (Turc et al., 2019), and also with DistilBERT model (Sanh et al., 2019) which has six transformer layers and already pre-distilled. We also compared with other model compression techniques such as knowledge distillation and pruning. For this, we compared with an existing work (Miresghallah et al., 2022). They used knowledge-distillation with DPSGD for privacy preserving model compression (DPKD). They also proposed privacy-preserving pruning strategy (DPIMP). There is also an existing work TSKD (Garg & Torra, 2024) that used task-specific knowledge distillation with DP for private model compression.

We chose to use BERT models as they are foundational models with a relatively smaller architecture, making them easier to interpret and understand. While more recent models like Llama models are widely used due to their advanced capabilities, their significantly larger model size adds complexity, making it harder to analyze their inner workings. Since

BERT and llama models share a similar transformer-based architecture, focusing on attention mechanisms for language understanding, the proposed approach would also work for llama models.

## 5.4 Privacy Analysis

We employ DP to safeguard sensitive data during model training, specifically using a DPSGD optimizer. The privacy guarantee in DP is characterized by the parameter  $\epsilon$ , often referred to as the privacy budget. In our approach, the  $\epsilon$  value directly corresponds to the  $\epsilon$  used in DPSGD. A crucial aspect of our methodology is that each training example  $x_i$  is utilized only once during the training process. This is important because, in standard DP mechanisms, repeated use of the same data point leads to cumulative privacy loss, a concept known as composition. However, since each training example is used exactly once, there is no iterative exposure, and thus no need to account for composition. Consequently, the privacy budget of the whole process is also  $\epsilon$ . This approach is similar to Local Differential Privacy (Joseph et al., 2018), where each user's data is independently protected before being incorporated into the model. The convergence bounds of our DP-SGD LoRA fine-tuning algorithm align with those established for DP-SGD. Existing theoretical analyses of DP-SGD convergence, such as those in prior studies (Koloskova et al., 2023), remain applicable to our approach.

## 5.5 Privacy Risk Assessment

When models are not trained with privacy-preserving algorithms, they become vulnerable to various privacy attacks such as membership inference attacks (Shokri et al., 2017), model inversion attacks (Fredrikson et al., 2015), and training data extraction attacks (Carlini et al., 2021). In this paper, we focus on evaluating the effectiveness of our privacy mitigation strategy through the lens of the training data extraction attack, as it is the most relevant and direct approach to assess our concerns.

A training data extraction attack aims to reconstruct exact instances from the training data, rather than producing merely similar or approximate examples. This type of attack is particularly effective in detecting whether a language model has memorized specific portions of its training data, which could lead to privacy breaches. To provide a clear context, we define memorization within the scope of language models as follows.

### 5.5.1 Memorization

Memorization is, to some extent, an inherent function of language models since their training objective is to maximize the likelihood of the training data. For example, language models need to “memorize” the correct spelling of words to perform well. When certain words or sentences appear multiple times in the training data, the model might memorize them to achieve higher accuracy and likelihood. However, if the model memorizes and reproduces a word or sentence that is sensitive and appears only once in the entire dataset, this indicates memorization that leads to privacy leakage. This scenario is particularly concerning as it exposes sensitive information that was not intended to be learned or shared by the model.

## 5.5.2 Threat Model

We consider an adversary with black-box access to the language model. This means the adversary can compute the probability of arbitrary sequences  $f_{\theta}(x_1, \dots, x_n)$  and generate text or obtain next-word predictions based on these probabilities. However, the adversary does not have access to the model's internal parameters, such as the individual weights or hidden states (e.g., attention vectors). This type of attack is highly realistic in real-world scenarios, especially considering that many language models, including GPT models, are often trained on sensitive data. Through careful prompt engineering, an adversary can generate synthetic text that may inadvertently reveal sensitive information. This risk highlights the importance of implementing robust privacy-preserving techniques during model training. The adversary's primary objective is to extract specific instances of memorized training data from the model. The effectiveness of the attack is evaluated based on the sensitivity of the extracted information, with the assumption that more sensitive or unique examples represent a greater privacy risk. Consequently, the attack's strength is measured by the degree of privacy compromise, specifically focusing on how well the adversary can retrieve highly private or unique training examples.

## 5.5.3 Privacy Attack Evaluation

We utilize two evaluation measures to quantify privacy. The first is a natural likelihood measure, the perplexity of a sequence, which quantifies how well the language model predicts the given data (Carlini et al., 2021). Specifically, given a sequence of tokens  $x_1, \dots, x_n$ , the perplexity  $P$  is defined as:

$$P = \exp \left( -\frac{1}{n} \sum_{i=1}^n \log f_{\theta}(x_i | x_1, \dots, x_{i-1}) \right) \quad (7)$$

Perplexity captures the model's uncertainty regarding the sequence, where a lower perplexity indicates that the model assigns higher average probabilities to the tokens in the sequence, suggesting the model is less "surprised" by the sequence. This implies that the model has a stronger predictive capacity over the given data.

Another key metric is BERTScore (Zhang et al., 2020), which uses pre-trained contextual embeddings from our fine-tuned model to measure the similarity between candidate and reference sentences via cosine similarity. BERTScore correlates well with human judgments at both sentence and system levels. It also calculates precision, recall, and F1 scores, providing a nuanced evaluation of language generation tasks.

## 6 Results and Discussion

In this section, we delve into the results from our experiments, analyzing the impact of various techniques and strategies on model performance. We first emphasize the importance of pairwise cosine similarity in Sect. 6.1, then compare the effectiveness of retraining in Sect. 6.2, effect of pruning rate on accuracy in Sect. 6.3 and fine-tuning methods in Sect. 6.4,

focusing on how these approaches influence model accuracy and provide a comparison between them in Sect. 6.5. Then, we also compare our methodology with existing baselines in Sect. 6.6, and discuss the advantages of redundancy based pruning in Sects. 6.7, and 6.8 provides the analysis of data synthesis using privacy attack.

## 6.1 Significance of Pairwise Cosine Similarity

In Table 1 we compare two different techniques of identifying redundancy in different layers. We compared the pairwise cosine similarity technique that we used in our approach for identification of redundant layers with the traditional cosine similarity. We computed cosine similarity between consecutive hidden states. It computes the average cosine similarity between hidden state  $\ell$  and hidden state  $\ell + 1$  for all layers. The list of similarities is then used to prune the least important layers based on the highest similarity values. The model prunes the layers with the highest cosine similarity values between consecutive layers.

The results show that using pairwise cosine similarity for pruning generally leads to better accuracy than using regular cosine similarity, both with and without DP. We used the privacy budget of  $\epsilon = 1$  when DP is used. Specifically, the models pruned with pairwise cosine similarity consistently achieve higher accuracy across the datasets compared to those pruned with regular cosine similarity. Additionally, LoRA fine-tuning tends to improve performance compared to standard training, with the greatest improvements observed when combining pairwise cosine similarity pruning with LoRA fine-tuning. When DP is applied, models using pairwise cosine similarity pruning and LoRA fine-tuning perform the best, highlighting the effectiveness of this combination in maintaining accuracy while ensuring privacy.

These results raise an intriguing question: why does the accuracy of tasks from the chosen framework—pairwise-cosine similarity and LoRA fine-tuning—perform better with DP than without it? We hypothesize that this improvement stems from the sequential application of pruning, DP mechanisms, and fine-tuning. Pruning redundant layers using pairwise similarity first reduces the number of parameters, ensuring that only the most impactful features are retained. This simplification makes gradient clipping for DP more effective, as it focuses on controlling the magnitude of updates for meaningful parameters, preventing overly large updates that could dominate learning. With fewer gradients to update, the noise added during the DP step is distributed over a smaller set of critical parameters, reducing the effective noise per parameter. This targeted application of noise acts as a regularizer, further aiding in avoiding over-fitting to the training data. Finally, LoRA fine-tuning adapts the model within a low-rank subspace which limits the degree of freedom and ensures that

**Table 1** Comparison of cosine similarity vs Pairwise-cosine similarity

Pruning strategy	Training	DP	SST-2	RTE	MRPC	COLA
Cosine	Train	–	88.75	62.45	80.21	77.93
Pairwise-Cosine	Train	–	91.05	64.62	82.84	78.14
Cosine	LoRA Fine-Tune	–	89.33	44.76	82.34	68.11
Pairwise-Cosine	LoRA Fine-Tune	–	91.97	47.29	83.71	69.12
Cosine	Train	Yes	88.72	46.93	60.65	62.48
Pairwise-Cosine	Train	Yes	90.11	49.09	63.33	64.21
Cosine	LoRA Fine-Tune	Yes	90.43	47.32	80.28	76.15
Pairwise-Cosine	LoRA Fine-Tune	Yes	92.31	49.45	82.37	79.70

the added DP noise minimally impacts meaningful updates. This combination of pruning, DP noise, and LoRA not only focuses the learning process on essential features but also facilitates more robust generalization, yielding better utility even under the constraints of DP. These results are also aligned with existing studies (Sakuma & Osame, 2017) that in some cases the prediction accuracy improves because of the noise reduction effects of the condensation process. When ML models are resistant to errors, some noise addition does not reduce dramatically the accuracy of the model. In fact, adding noise may results into models that are better from the point of view of generalization.

Later on, we plotted a heat map between different layers as shown in Fig. 2. It depicts the pairwise cosine similarity between the hidden states of different layers in a BERT model, including the embedding layer and all 12 transformer layers. The color gradient, ranging from red to blue, illustrates the degree of similarity: red indicates higher cosine similarity, suggesting that the embedding from the corresponding layers are more alike, while blue signifies lower cosine similarity, indicating greater differences. This is important because it helps in identifying redundant layers, where high similarity (in red color) suggests that two layers may be performing similar functions, making one potentially redundant. This visualization allows us to identify which layers produce similar representations and which layers exhibit distinct characteristics, providing insights into the internal structure and behavior of the BERT model across its various layers.

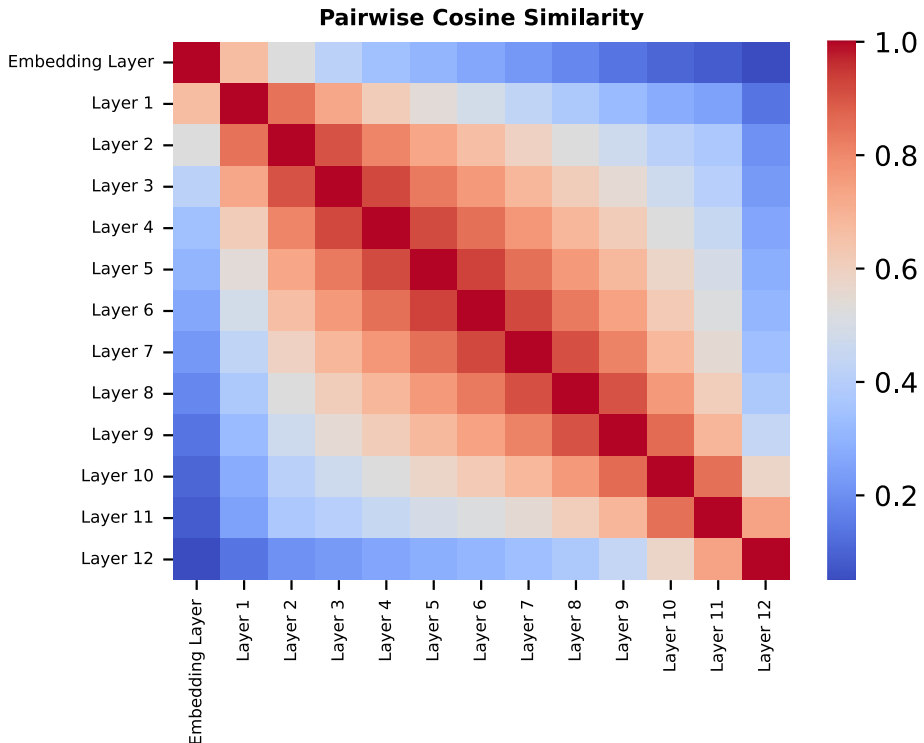


Fig. 2 Pairwise cosine similarity between BERT layers

## 6.2 Comparative Analysis of Model Re-training: With and Without Differential Privacy

We assess the impact of full model retraining following our proposed pruning methodology. After pruning, instead of just fine-tuning the model, we retrain it from scratch. This approach aims to recover any lost information, though it is computationally more expensive compared to fine-tuning.

Table 2 presents the results of retraining the pruned model. We explored various sparsity levels, ranging from pruning 10% of layers (sparsity level = 0.1) to 30% (sparsity level = 0.3). This exploration helps us understand the trade-offs between model size, performance, and privacy. We also evaluated the effects of differential privacy (DP), comparing results with and without DP to assess its impact on model performance. We used the privacy budget of  $\epsilon = 1$  when DP is used. As sparsity increases from 0.1 to 0.3, more layers are pruned, which reduces the model's complexity and inference time. However, this also leads to a slight decrease in accuracy across different tasks. For instance, in the SST-2 dataset, accuracy drops from 91.05% with 0.1 sparsity to 84.60% with 0.3 sparsity when retrained without DP.

The introduction of DP further decreases accuracy beyond the effects of sparsity alone. Interestingly, the impact of DP combined with higher sparsity can be less severe compared to DP with lower sparsity levels. This suggests that higher sparsity might mitigate some of the negative effects of DP. For example, in the MRPC dataset, the accuracy with 0.1 sparsity and DP is 63.33%, while with 0.3 sparsity and DP, it improves to 68.13%. In summary, there is a trade-off between sparsity and model performance. Retraining the pruned model from scratch offers a more thorough recovery of accuracy, but it is more computationally demanding. Additionally, while DP reduces accuracy, its combined effect with higher sparsity may sometimes result in better performance than lower sparsity levels with DP.

### 6.3 The Effect of Pruning Rate on Accuracy

Higher sparsity levels lead to models with fewer parameters, which simplifies the model and reduces the total gradient magnitudes during training. This reduction enhances the efficiency of DP, as fewer gradients are subjected to clipping and noise addition. Simpler, pruned models are inherently less sensitive to noise, allowing the DP noise to act as an effective regularizer rather than a source of distortion. By focusing updates on the most critical parameters, sparsity improves the signal-to-noise ratio of gradient updates, preserving useful information even under DP constraints. Additionally, pruning removes redundant parameters, reducing overfitting risks and encouraging the model to prioritize essential features. This synergy between sparsity and DP creates a more robust training process, where higher

**Table 2** Re-training the model with and without DP

Sparsity	DP	SST-2	RTE	MRPC	COLA	Avg
0.1	–	91.05	64.62	82.84	78.14	79.16
0.2	–	88.79	59.92	84.06	78.14	77.72
0.3	–	84.60	61.73	82.35	73.92	75.65
0.1	Yes	90.11	49.09	63.33	64.21	66.68
0.2	Yes	88.99	50.54	67.64	54.74	65.47
0.3	Yes	83.66	61.73	68.13	65.12	69.66

sparsity levels help the model retain utility despite the privacy-preserving modifications. This positive correlation can be in general observed from Table 2 and it aligns with existing studies (Adamczewski & Park, 2023) where they observed a positive correlation between the pruning rate and the test accuracy when the models are trained with DP.

## 6.4 Comparative Analysis of Model Fine-Tuning: With and Without Differential Privacy

Table 3 illustrates the effects of combining model compression with parameter-efficient fine-tuning on model accuracy. We applied our pruning strategy at various levels of sparsity to reduce the model's size, and then fine-tuned the pruned models using the Low-Rank Adaptation (LoRA) technique. The table shows the model's accuracy across different tasks, both with Differential Privacy (DP) with  $\epsilon = 1$  for privacy preservation and without it. The results reveals that as sparsity increases, there is a noticeable drop in model accuracy for both private and non-private models. However, this accuracy reduction is more pronounced in non-private models. This suggests that while pruning is effective for reducing model size and complexity, it comes with a trade-off in performance, and non-private models are more affected by this trade-off. The application of DP, which introduces noise for privacy protection, generally leads to a decrease in accuracy. Yet, this impact is somewhat mitigated at higher sparsity levels. For instance, in the RTE dataset, the accuracy without DP decreases from 47.29 to 46.57%, whereas with DP, it only drops from 49.45 to 48.73%. This indicates that higher sparsity can help offset some of the accuracy loss attributed to DP.

Overall, the results underline the importance of balancing sparsity, privacy, and fine-tuning strategies. Lower sparsity levels combined with effective fine-tuning methods like LoRA can help preserve model accuracy while achieving model compression and privacy protection.

## 6.5 Training vs. Fine-Tuning

In addition to evaluating our pruning and fine-tuning strategies, we compared full model retraining with parameter-efficient fine-tuning to determine which approach better maintains model performance while preserving privacy. Full model retraining, although exhaustive, enables comprehensive recovery of information lost during pruning but is computationally intensive and time-consuming, especially with DP, where added noise complicates the process. On the other hand, parameter-efficient fine-tuning with methods like LoRA (Low-Rank Adaptation) adjusts only a subset of model parameters, offering a more efficient alternative. LoRA fine-tuning allows for faster adaptation to the pruned model while incorporating privacy safeguards, making it less resource-intensive compared to full retraining.

**Table 3** Performance comparison of parameter-efficient fine-tuning using LoRA with and without DP

Sparsity	DP	SST-2	RTE	MRPC	COLA	Avg
0.1	–	91.97	47.29	83.71	69.12	71.69
0.2	–	89.33	46.93	83.21	68.42	71.97
0.3	–	86.69	46.57	78.38	68.05	71.25
0.1	Yes	92.31	49.45	82.37	79.70	75.95
0.2	Yes	89.44	46.57	80.72	78.25	73.74
0.3	Yes	85.32	48.73	82.11	76.63	73.20

**Table 4** Training vs. parameter-efficient fine-tuning using DP

Training	Sparsity	SST-2	RTE	MRPC	COLA
Train	0.1	90.11	49.09	63.33	64.21
LoRA Fine-Tune	0.1	92.31	49.45	82.37	79.70
Train	0.2	88.99	50.54	67.64	54.74
LoRA Fine-Tune	0.2	89.44	46.57	80.72	78.25
Train	0.3	83.66	61.73	68.13	65.12
LoRA Fine-Tune	0.3	85.32	48.73	82.11	76.63

**Table 5** Comparison with baselines

Model	Training	SST-2	QNLI
BERT-base	Finetune	92.31	87.90
BERT-small	Finetune	79.93	81.96
DistilBERT	Finetune	90.90	87.33
1/2-BERT	DPKD (Miresghallah et al., 2022)	78.5	80.10
1/2-BERT	Structured DPIMP (Miresghallah et al., 2022)	83.3	80.90
SparseBERT	Unstructured DPIMP (Miresghallah et al., 2022)	83.7	82.20
BERT-small	TSKD (Garg & Torra, 2024)	86.23	83.17
BERT-base	<b>PrunePrivyTune</b>	92.31	86.51

Table 4 shows that models fine-tuned with LoRA achieve higher accuracy across datasets (SST-2, RTE, MRPC, and COLA) compared to those trained without LoRA. For example, at a sparsity level of 0.1, accuracy on SST-2 improves from 90.11% to 92.31% with LoRA fine-tuning, and similar improvements are observed across other datasets. This trend is consistent across all sparsity levels, indicating that LoRA fine-tuning is more effective in maintaining or enhancing model performance after pruning. These findings suggest that LoRA's parameter-efficient fine-tuning is superior to traditional retraining methods in balancing model compression.

## 6.6 Comparison with Baselines

Table 5 compares various approaches to model training and performance across several NLP tasks using different models derived from BERT. It includes full fine-tuning of models like BERT-base, BERT-small, and DistilBERT, as well as different compression techniques like knowledge-distillation with zero-shot prompting, structured and unstructured pruning (DPIMP), and task-specific knowledge distillation (TSKD). BERT-base having the most parameters i.e., 109 M, achieves the highest accuracy across tasks like SST-2 and QNLI, but it is the most computationally expensive. In contrast, smaller models like BERT-small, with only 5 M parameters, show a significant drop in performance. DistilBERT offers a middle ground with reduced parameters of upto 66 M and moderate accuracy. The table also includes approaches like 1/2-BERT and SparseBERT, which use pruning techniques, showing how structured and unstructured pruning can recover some performance while reducing the model size. Differentially Private Knowledge Distillation (DPKD) (Miresghallah et al., 2022) resulted in 78.5% accuracy on SST-2 dataset while 80.10% on QNLI which is quite lower than the state-of-the-art BERT models. Nevertheless, Miresghallah et al. (2022)

found that pruning results better performance than distillation with an accuracy of 83.70% on SST-2 and 82.20% on QNLI dataset. Also, another approach Task-Specific Knowledge Distillation (TSKD) (Garg & Torra, 2024), improved the accuracy of their model by 86.23% on SST-2 and 83.17% on QNLI dataset. On the contrary, our approach of *PrunePrivyTune* performs superior to the existing works of private model compression and almost similar to the BERT state-of-the-art model which is quite computationally expensive. Finally, the table shows that the proposed approach using LoRA for fine-tuning BERT-base maintains top performance on several datasets matching the full BERT-base fine-tuning results, illustrating the effectiveness of the proposed method in preserving performance with efficient parameter usage. Prior work on pruning and PEFT techniques, including LoRA, consistently demonstrates that such approaches significantly reduce computational cost and model size while maintaining performance. Thus, based on the structural pruning and LoRA-based fine-tuning applied in our framework, we can qualitatively assert that the proposed method is computationally efficient relative to the uncompressed baseline.

## 6.7 Advantages of Redundancy Based Pruning for DPLoRA

Our pruning method improves upon traditional techniques like magnitude-based pruning, particularly for DPLoRA. Magnitude-based pruning removes layers based on weight magnitude, assuming smaller weights contribute less to the model's output. However, it fails to account for semantic redundancy between layers, leading to potential utility loss when important layers with small weights are pruned. In contrast, our redundancy-based approach uses cosine similarity between hidden states of consecutive layers to identify redundant layers, which contribute minimally to downstream tasks. By pruning only these redundant layers, we minimize utility loss, crucial for preserving accuracy during DP fine-tuning.

Moreover, in DPLoRA, DP noise depends on gradient sensitivity, which is influenced by the number of parameters and updates in pruned layers. Traditional pruning methods ignore this, potentially increasing gradient noise variance and inefficiency. By pruning redundant layers, we reduce gradient sensitivity, minimizing noise injected into gradients and improving the privacy-utility trade-off.

Additionally, the convergence rate of DPSGD is inversely proportional to noise scale and directly proportional to gradient variance. Traditional methods may increase variance, slowing convergence. Our method reduces gradient variance, leading to faster convergence while maintaining DP constraints, making it more efficient for DPLoRA fine-tuning. Furthermore, studies show that pruning itself is a step forward in enhancing model privacy (Huang et al., 2020), and pruning can prevent leakage from membership inference attacks (Wang et al., 2020). When we privately fine-tune the model using DPLoRA, we make the model more private while also reducing model storage and computational costs.

## 6.8 Quantifying Privacy and Memorization in Synthetic Data

Language models tend to memorize the data it is trained on, so to avoid memorization and sensitive information to be leaked DP is employed during training. After initially pruning the model and subsequently fine-tuning it using a privacy-preserving approach, we generate synthetic data with the model to evaluate privacy effectiveness. Table 6 presents an evaluation of the synthetic data generated across various datasets using two metrics dis-

**Table 6** Evaluation of generated synthetic data using perplexity and BERTScore

DP	Metric	SST-2	RTE	MRPC	COLA
No	Perplexity	$6.32 \times 10^5$	$5.27 \times 10^6$	$6.29 \times 10^{-3}$	$4.87 \times 10^1$
	BERT_Precision	0.3701	0.3931	0.3833	0.3425
	BERT_Recall	0.3698	0.3940	0.3852	0.3481
	BERT_F1	0.3697	0.3935	0.3832	0.3460
$\epsilon=1$	Perplexity	$7.10 \times 10^5$	$5.76 \times 10^6$	$1.12 \times 10^{-2}$	$5.22 \times 10^4$
	BERT_Precision	0.3536	0.3613	0.3412	0.3384
	BERT_Recall	0.3500	0.3679	0.3484	0.3392
	BERT_F1	0.3512	0.3646	0.3448	0.3380
$\epsilon=5$	Perplexity	$6.83 \times 10^5$	$5.52 \times 10^6$	$9.10 \times 10^{-3}$	$9.25 \times 10^3$
	BERT_Precision	0.3573	0.3754	0.3650	0.3397
	BERT_Recall	0.3509	0.3848	0.3724	0.3399
	BERT_F1	0.3523	0.3800	0.3678	0.3385
$\epsilon=10$	Perplexity	$6.62 \times 10^5$	$5.35 \times 10^6$	$7.50 \times 10^{-3}$	$1.40 \times 10^3$
	BERT_Precision	0.3694	0.3879	0.3690	0.3412
	BERT_Recall	0.3690	0.3885	0.3922	0.3474
	BERT_F1	0.3702	0.3882	0.3790	0.3442

cussed in Sect. 5: Perplexity and BERTScore. Perplexity gauges the model's confidence in its predictions; lower perplexity signifies that the model is more certain and may have memorized similar training data, while higher perplexity suggests the model is less confident and encountering the data in a more novel context. On the other hand, BERTScore, which includes Precision, Recall, and F1, assesses the quality of synthetic text in terms of similarity to reference text.

The results indicate that as the value of  $\epsilon$  increases, perplexity generally decreases across all datasets. Specifically, with  $\epsilon = 1$ , which we consider the highest level of privacy, the model exhibits the highest perplexity, reflecting greater uncertainty and reduced likelihood of memorizing training data. As  $\epsilon$  increases from 1 to 10, privacy levels decrease, leading to reduced perplexity. This reduction in perplexity implies that the model becomes more confident in its predictions and may retain more information from the training data, indicating higher memorization. In the absence of DP, memorization is most pronounced, resulting in the lowest perplexity. This pattern highlights the trade-off between privacy and model confidence, where stronger privacy constraints (lower  $\epsilon$ ) lead to higher uncertainty and better privacy, while weaker constraints (higher  $\epsilon$ ) result in greater confidence and potential for memorization.

For the BERTScore Precision, Recall, and F1 metrics are observed as  $\epsilon$  increases, indicating that the synthetic data becomes more similar to the original training data. When  $\epsilon=1$ , the synthetic data exhibits the least similarity to the training data, reflecting strong privacy preservation through DP. As  $\epsilon$  increases, privacy protection diminishes, allowing the synthetic data to more closely resemble the original text, which results in higher BERTScores. The highest BERTScore is achieved when no privacy constraints are applied, demonstrating that the synthetic data is most similar to the training data in the absence of privacy guarantees.

## 6.9 Assessment Analysis

In this section, we assess how the objectives outlined in this paper are achieved by summarizing the key observations corresponding to each objective.

O1: The first objective was to develop and evaluate a pairwise structural pruning approach in comparison to traditional pruning strategies. As discussed in Sects. 6.1, our experiments demonstrate that combining pairwise pruning with LoRA-based fine-tuning yields higher accuracy than conventional pruning methods, highlighting the effectiveness of global similarity-based pruning in preserving model performance.

O2: The second objective focused on identifying an efficient post-pruning recovery strategy, comparing full retraining with fine-tuning. As presented in Sects. 6.2, 6.3, 6.4, 6.5, our results indicate that DP-LoRA-based fine-tuning not only achieves superior performance but also significantly reduces computational cost and training time compared to full retraining, demonstrating the efficiency of the proposed parameter-efficient adaptation approach.

O3: The third objective aimed to analyze the impact of incorporating differential privacy during fine-tuning relative to non-private baselines. As shown in Sects. 6.2, 6.4, our findings highlight the trade-off between pruning sparsity, DP privacy level, and post-pruning strategy, providing insights into balancing model utility with privacy preservation.

O4: The fourth objective was to generate synthetic data from the privately fine-tuned model in order to evaluate its memorization capabilities and privacy properties. As discussed in Sect. 6.8, we observe that stronger privacy constraints lead to increased uncertainty in the model's outputs, reducing memorization and enhancing privacy preservation. Overall, these analyses demonstrate that the proposed *PrunePrivyTune* framework effectively addresses model compression, post-pruning adaptation, and privacy preservation while maintaining high performance across tasks.

**Limitation:** This work was conducted using a base-sized BERT model, which enabled efficient experimentation. Extending the proposed pruning and differential privacy techniques to larger models, such as LLaMA or other state-of-the-art LLMs, would require substantially greater computational resources and careful tuning of sparsity levels and DP noise parameters to maintain performance. Additionally, the experiments were performed exclusively on GLUE datasets. While the methodology is potentially applicable to other NLP tasks, its effectiveness in terms of pruning efficiency, model accuracy, and privacy preservation may vary depending on task complexity, dataset size, and the nature of the input data, highlighting potential constraints on generalizability. Furthermore, accurately measuring computational time is challenging for large models due to their scale and training overhead. The results also depend on multiple experimental factors, including the choice of DP epsilon, pruning sparsity levels, and other hyperparameters, making the tuning process labor-intensive and potentially tedious. These aspects underscore the need for careful hyperparameter optimization and motivate future work to systematically explore scalability, efficiency, and generalizability of the proposed framework.

## 7 Conclusion and Future Works

In this paper, we proposed the *PrunePrivyTune* methodology, which aims to reduce the inference time of large language models by employing pruning, a model compression technique. Recognizing that transformer models have a tendency to memorize the data they are trained on, we also introduced a differentially private fine-tuning approach to protect against privacy leakage after the model has been pruned. To evaluate the privacy vulnerabilities of the model, we generated synthetic data from the fine-tuned model and conducted a training data extraction attack to analyze the extent of information retention. We assessed the model's performance in terms of accuracy and privacy vulnerabilities, using perplexity and BERTScore as evaluation metrics. Our framework effectively reduces model size through pruning while also mitigating the risk of the model memorizing sensitive data. This results in a more compact model that retains its performance while significantly reducing the risk of data leakage. For future work, we plan to explore the integration of other possible model compression techniques and adaptive differential privacy mechanisms to further enhance the efficiency and security of large language models such as llama models.

**Acknowledgements** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

**Author Contributions** S. Garg contributed to the idea, methodology, experimentation, initial draft writing, and V. Torra contributed in framing the idea, validation, supervision, reviewing.

**Funding** Open access funding provided by Umea University.

**Data Availability** No datasets were generated or analysed during the current study.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 308–318).
- Adamczewski, K., & Park, M. (2023). Differential privacy meets neural network pruning. [arXiv:2303.04612](https://arxiv.org/abs/2303.04612)
- Brown, T. B. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.
- Buciluă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 535–541).

- Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., & Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 267–284.
- Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., & Oprea, A. (2021). Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, (pp. 2633–2650).
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240), 1–113.
- Coavoux, M., Narayan, S., & Cohen, S. B. (2018). Privacy-preserving neural representations of text. [arXiv:1808.09408](https://arxiv.org/abs/1808.09408)
- Dalvi, F., Sajjad, H., Durrani, N., & Belinkov, Y. (2020). Analyzing redundancy in pretrained transformer models. [arXiv:2004.04010](https://arxiv.org/abs/2004.04010)
- Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, (pp. 1–12). Springer.
- Frantar, E., & Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, (pp. 10323–10337). PMLR.
- Fredrikson, M., Jha, S., & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1322–1333).
- Garg, S., & Torra, V.(2024). Task-specific knowledge distillation with differential privacy in LLMs. In *European Symposium on Research in Computer Security*.
- Ghazvininejad, M., Levy, O., Liu, Y., & Zettlemoyer, L. (2019). Mask-predict: Parallel decoding of conditional masked language models. [arXiv:1904.09324](https://arxiv.org/abs/1904.09324)
- Gong, Y., Liu, L., Yang, M., & Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. [arXiv:1412.6115](https://arxiv.org/abs/1412.6115)
- Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, (Vol. 28, pp. 1135–1143).
- Hanson, S., & Pratt, L. (1988). Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems*, (Vol. 1).
- Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., & Liu, Q. (2020). Dynabert: Dynamic Bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33, 9782–9793.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). Lora: Low-rank adaptation of large language models. [arXiv:2106.09685](https://arxiv.org/abs/2106.09685)
- Huang, Y., Su, Y., Ravi, S., Song, Z., Arora, S., & Li, K. (2020). Privacy-preserving learning via deep net pruning. [arXiv:2003.01876](https://arxiv.org/abs/2003.01876)
- Joseph, M., Roth, A., Ullman, J., & Waggoner, B. (2018). Local differential privacy for evolving data. In *Advances in Neural Information Processing Systems*, (Vol. 31).
- Koloskova, A., Hendrikx, H., & Stich, S.U. (2023). Revisiting gradient clipping: Stochastic bias and tight convergence guarantees. In *International Conference on Machine Learning*, (pp. 17343–17363). PMLR.
- Kurtic, E., Campos, D., Nguyen, T., Frantar, E., Kurtz, M., Fineran, B., Goin, M., & Alistarh, D. (2022). The optimal Bert surgeon: Scalable and accurate second-order pruning for large language models. [arXiv:2203.07259](https://arxiv.org/abs/2203.07259)
- Kwon, S. J., Lee, D., Kim, B., Kapoor, P., Park, B., & Wei, G.-Y. (2020). Structured compression by weight encryption for unstructured pruning and quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (pp. 1909–1918).
- Lee, J., Park, S., Mo, S., Ahn, S., & Shin, J. (2020). Layer-adaptive sparsity for the magnitude-based pruning. [arXiv:2010.07611](https://arxiv.org/abs/2010.07611)
- Li, Y., Baldwin, T., & Cohn, T. (2018). Towards robust and privacy-preserving text representations. [arXiv:1805.06093](https://arxiv.org/abs/1805.06093)
- Ma, X., Fang, G., & Wang, X. (2023). LLM-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36, 21702–21720.
- Mireshghallah, F., Backurs, A., Inan, H. A., Wutschitz, L., & Kulkarni, J. (2022). Differentially private model compression. *Advances in Neural Information Processing Systems*, 35, 29468–29483.
- Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., & Erlingsson, Ú. (2018). Scalable private learning with pate. [arXiv:1802.08908](https://arxiv.org/abs/1802.08908)
- Ponomareva, N., Hazimeh, H., Kurakin, A., Xu, Z., Denison, C., McMahan, H. B., Vassilvitskii, S., Chien, S., & Thakurta, A. G. (2023). How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77, 1113–1201.

- Sakuma, J., & Osame, T. (2017). Recommendation with k-anonymized ratings. [arXiv:1707.03334](https://arxiv.org/abs/1707.03334)
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of Bert: smaller, faster, cheaper and lighter. [arXiv:1910.01108](https://arxiv.org/abs/1910.01108)
- Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, (pp. 3–18). IEEE.
- Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., & Li, Y. (2022). Llama: Language models for dialog applications. [arXiv:2201.08239](https://arxiv.org/abs/2201.08239)
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., & Rodriguez, A. (2023). Llama: Open and efficient foundation language models. [arXiv:2302.13971](https://arxiv.org/abs/2302.13971)
- Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Well-read students learn better: On the importance of pre-training compact models. [arXiv:1908.08962v2](https://arxiv.org/abs/1908.08962v2)
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. [arXiv:1804.07461](https://arxiv.org/abs/1804.07461)
- Wang, Y., Wang, C., Wang, Z., Zhou, S., Liu, H., Bi, J., Ding, C., & Rajasekaran, S. (2020). Against membership inference attack: Pruning is all you need. [arXiv:2008.13578](https://arxiv.org/abs/2008.13578)
- Wang, Z., & Wohlwend, J., Lei, T. (2019). Structured pruning of large language models. [arXiv:1910.04732](https://arxiv.org/abs/1910.04732)
- Wu, J., Leng, C., Wang, Y., Hu, Q., & Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 4820–4828).
- Xin, J., Tang, R., Lee, J., Yu, Y., & Lin, J. (2020). DeeBERT: Dynamic early exiting for accelerating Bert inference. [arXiv:2004.12993](https://arxiv.org/abs/2004.12993)
- Xu, D., Yen, I.E., Zhao, J., & Xiao, Z. (2021). Rethinking network pruning-under the pre-train and fine-tune paradigm. [arXiv:2104.08682](https://arxiv.org/abs/2104.08682)
- Ye, D., Lin, Y., Huang, Y., & Sun, M. (2021). Tr-BERT: Dynamic token reduction for accelerating bert inference. [arXiv:2105.11618](https://arxiv.org/abs/2105.11618)
- Yeom, S.-K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.-R., & Samek, W. (2021). Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, Article 107899.
- Yu, D., Zhang, H., Chen, W., & Liu, T.-Y. (2021). Do not let privacy overbill utility: Gradient embedding perturbation for private learning. [arXiv:2102.12677](https://arxiv.org/abs/2102.12677)
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BertScore: Evaluating text generation with bert. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=SkeHuCVFDr>
- Zhu, M., & Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. [arXiv:1710.01878](https://arxiv.org/abs/1710.01878)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Authors and Affiliations

Sonakshi Garg<sup>1,2</sup> · Vicenç Torra<sup>1</sup>

✉ Sonakshi Garg  
sgarg@cs.umu.se

Vicenç Torra  
vtorra@cs.umu.se

<sup>1</sup> Umeå University, Umeå, Sweden

<sup>2</sup> South Asian University, Delhi, India