Differentially Private Model Compression

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent papers have shown that large pre-trained language models (LLMs) such as BERT, GPT-2 can be fine-tuned on *private data* to achieve performance comparable to non-private models for many downstream Natural Language Processing (NLP) tasks while simultaneously guaranteeing differential privacy. The inference cost of these models – which consist of hundreds of millions of parameters – however, can be prohibitively large. Hence, often in practice, LLMs are *compressed* before they are deployed in specific applications. In this paper, we initiate the study of differentially private model compression and propose frameworks for achieving 50% sparsity levels while maintaining nearly full performance. We demonstrate these ideas on standard GLUE benchmarks using BERT models, setting benchmarks for future research on this topic.

1 Introduction

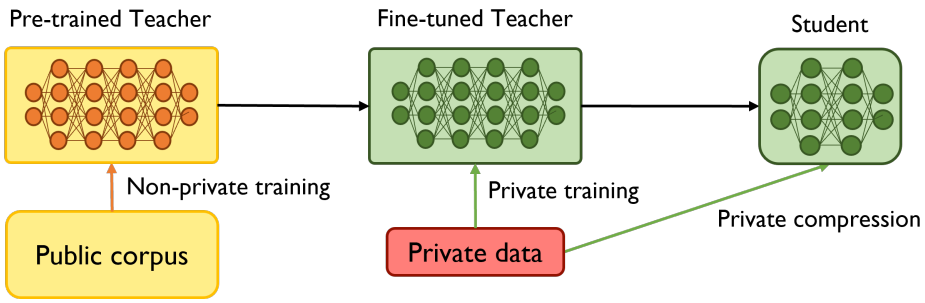

Figure 1: The 3-pronged modern deep learning pipeline: Pre-train on public data, fine-tune on private data, and compress the model to meet the memory and latency requirements of specific applications.

Since the advent of Transformer-based [58] large language models (LLMs) such as BERT [12, 35], GPT families [43, 4], there has been a paradigm shift in the way deep learning models are trained for natural language processing (NLP) applications. LLMs are first *pre-trained* on extremely large and diverse publicly available datasets. The weights are then *fine-tuned* for a specific task of interest using a much smaller *private dataset*, which may consist of sensitive information about the users. Finally, before deploying, the models are compressed (also referred to as sparsified or distilled) to reduce the parameter count. The reason for this final step is that LLMs such as BERT and GPT-2 consist of hundreds of millions of parameters, and hence inference time and memory footprints of the models are too large to be used in many applications. As a concrete example, consider a language model that is deployed for sentence completion task in text editors. In this scenario, clearly, the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

inference (response) time of the model needs to be in orders of milliseconds to be useful. Besides the practical considerations, it is also commonly observed that large deep learning models (even the ones that are not pre-trained) consist of many redundant parameters in the network that can be removed while retaining the full performance [30, 25, 24]. Hence most modern deep learning pipelines use this three-pronged approach of pre-train, fine-tune, and compress as illustrated in Figure 1.

While transformer-based models swayed deep learning research towards making models larger to achieve better performance, several recent works have shown that over-parameterized large models also increase the risk of leaking sensitive information about their training datasets [46, 7, 8]. Over the past few years, training deep learning models guaranteeing differential privacy (DP) [16], a strong notion of data privacy, has emerged as the defacto method to mitigate such information leakage. Exciting recent works have also shown that large pre-trained models when fine-tuned via DPSGD are as good as non-private models for a variety of NLP and image applications [65, 32, 11].

The main purpose of this paper is to understand how private training impacts the modern deep learning pipeline of pre-train, fine-tune, and compress. The main observation is that most widely used model compression algorithms such as Knowledge Distillation (KD) and Pruning, use private datasets to produce compressed models, hence if our goal is to deploy a differentially private compressed model, we should consider their impact on the training process. This leads to the question:

What algorithms should one use to produce compressed private models and how do they impact private fine-tuning via DPSGD?

The goal of this paper is to investigate this question and propose frameworks for private model compression in the context of NLP applications. Although we investigate model compression techniques at the fine-tuning stage using pre-trained models, we would like to emphasize that our frameworks for private model compression are not tied to this setting. They are equally applicable to training deep learning models from scratch and to other application domains such as image classification tasks.

1.1 Our Contributions

- We give a framework for doing model compression using Knowledge Distillation algorithm guaranteeing differential privacy, which we call DPKD. We show that DPKD alone is not enough to transfer the knowledge from large models to compressed models. This loss in the accuracy of compressed models can be mitigated by better initialization of compressed models from the weights of the large models, which itself is a form of knowledge transfer. We propose several *zero-shot, fully private* methods for initialization compressed models using weights of the large models. Our empirical evaluation of these ideas on standard GLUE benchmarks using BERT models show that DPKD approach to the model compression loses an accuracy of 5% compared to the larger models if the compressed model has *half the size of the full BERT model*.
- To overcome the limitations of DPKD algorithm for model compression, we consider a framework for evolving the larger models to compressed models via private adaptation of Iterative Magnitude Pruning (DPIMP). We show that on standard GLUE benchmarks using BERT models, DPIMP framework produces compressed models whose performance is comparable to larger models at 50% unstructured sparsity levels.
- As a byproduct of DPIMP approach for model compression, our work also shows that pre-trained BERT models have sparse subnetworks that can be found via DPSGD that have *almost* matching performances of the private full model, similar to the Lottery Ticket Hypothesis for BERT models in non-private settings [19, 10].

To the best of our knowledge, no prior work have studied model compression techniques of LLMs in private settings. A problem broadly related to model compression is ensemble learning, where the goal is to transfer knowledge from an ensemble of teacher models to a single student model [13]. This problem was studied in the private setting by [42], who proposed the Private Aggregation of Teacher

Ensembles (PATE) framework. In PATE an ensemble of teacher models is trained on *disjoint private data* and the student model is trained by noisy aggregation of teachers' answers. Two recent works combine PATE framework with KD algorithm for doing noisy aggregation of teachers' answers for mobile analytics and text generation problems [36, 55]. The PATE framework and ensemble learning techniques can be applied for fine-tuning (or training) deep learning models. Unfortunately, however, as previous works have shown, the performance of deep learning models trained via PATE are inferior to that of DPSGD for complex datasets [66]. As the performance of a fine-tuned large model on a sensitive dataset is an upper bound on the performance of a compressed model, and we do not know how to fine-tune large models using PATE to match the performance of DPSGD, we do not consider PATE framework for model compression.

2 Preliminaries

Recall the formal definition of differential privacy.

Definition 2.1 (Differential Privacy (DP) [16, 15])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ,δ)-differentially private if for any two neighboring datasets D and D', which differ in exactly the data pertaining to a single user, and for all sets $\mathcal{S}$ of possible outputs:* $\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta$.

We train all our models via DPSGD as the optimizer. We briefly describe the algorithm.

2.1 Training via DPSGD

To train a deep learning model with privacy, the most widely used algorithm is the DP stochastic gradient descent (DPSGD) [48, 3, 1, 45]. DPSGD augments the standard SGD algorithm with per-example gradient clipping and Gaussian noise addition steps. These two steps serve to limit and mask the contribution of a single example. At a high-level, the privacy analysis of DPSGD proceeds by first showing that each iteration of DPSGD is differentially private for some (ϵ, δ), then applying amplification by subsampling and composition across all the iterations. To get the tightest privacy parameters, however, one needs more sophisticated arguments such as the Moments Accountant method [1] or numerical composition algorithms [21].

3 Compressed Models via Knowledge Distillation

One of the most widely used algorithms for compressing models is knowledge distillation (KD) [26, 5, 44]. In this section, we propose a framework for implementing knowledge distillation with DP constraints, and evaluate its effectiveness on standard GLUE benchmarks. We begin by briefly describing how the KD algorithm is applied for compressing models; we refer the readers to [22] for more details. Adopting the naming convention from the literature, for the rest of the paper, we call large pre-trained models as *teacher models* and compressed smaller models as *student models*.

3.1 Non-Private Knowledge Distillation

Let $\mathcal{T}$ be a teacher network with the class probabilities $P_{\mathcal{T}} = \text{softmax}(a_{\mathcal{T}})$ (a.k.a. soft labels) where $a_{\mathcal{T}}$ is the output of last layer before the softmax operation. Similarly, let $\mathcal{S}$ be a student network with parameters $W_{\mathcal{S}}$ and class probabilities $P_{\mathcal{S}} = \text{softmax}(a_{\mathcal{S}})$. The main idea behind KD algorithm is to train $\mathcal{S}$ to mimic the output distribution of the teacher $P_{\mathcal{T}}$ and the true labels. The intuition is that $P_{\mathcal{T}}$ captures the knowledge learnt by the teacher, in particular probabilities assigned by the teacher to labels that are different from the true label. Hinton et al. [26] suggested to use softmax-temperature where probability for class i of the teacher is given by

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

with logits z_i where T controls the smoothness of the output distribution. Setting a higher value for the temperature parameter T produces a softer probability distribution over classes. The same

relaxation is applied to the output of the student network. The student is trained to minimize the weighted combination of the distillation loss and the supervised training loss:

$$\mathcal{L}_{\text{KD}}(\mathcal{W}_{\mathcal{S}}) := \mathcal{H}(\mathbf{y}_{\text{true}}, P_{\mathcal{S}}) + \lambda \cdot \mathcal{H}(P_{\mathcal{T}}, P_{\mathcal{S}}) \tag{1}$$

where $\mathcal{H}$ refers to the cross-entropy and λ is a hyperparameter.

3.2 Differentially Private Knowledge Distillation (DPKD)

A natural way to generalize KD algorithm for private distillation of student model is to train the student with DPSGD to minimize the loss function given by Equation 1. However, such an algorithm fails to produce student models satisfying DP because of the term $\mathcal{H}(P_{\mathcal{T}}, P_{\mathcal{S}})$ in Equation 1. Note that $P_{\mathcal{T}}$ is a function of the entire dataset, hence clipping and adding noise alone is not enough to argue that DPSGD produces a private student. A natural solution to overcome this hurdle is to first train the teacher models with DPSGD and then apply KD. We propose our DPKD framework in Algorithm 1.

Algorithm 1 Differentially Private Knowledge Distillation (DPKD)

Input: Teacher model $\mathcal{T}$, student model $\mathcal{S}$, private data D, privacy budget (ϵ, δ)
Output: Student model $\mathcal{S}$ satisfying (ϵ, δ)-DP
1: Find an allocation of (ϵ_1, δ_1), (ϵ_2, δ_2) and (ϵ_3, δ_3) from the privacy budget (ϵ, δ)
2: Train $\mathcal{T}$ on D with DPSGD using privacy budget of (ϵ_1, δ_1)
3: Initialize $\mathcal{S}$ (possibly privately with privacy budget (ϵ_2, δ_2))
4: Train $\mathcal{S}$ on D to minimize Eq. (1) with DPSGD using privacy budget of (ϵ_3, δ_3)
5: **return** $\mathcal{S}$

As the model parameters produced by DPSGD satisfy privacy guarantees, using the post processing property of DP [17], one can show the following theorem. We omit the proof.

Theorem 3.1. *The output of DPKD algorithm is differentially private with privacy parameters obtained by the adaptive composition of privacy parameters in steps 2, 3, and 4 of Algorithm 1.*

In this work we use numerical composition of privacy mechanisms as given in [21]. Our framework for DPKD raises several interesting algorithmic and hyperparameter tuning questions. The most interesting one is whether one can have DPKD algorithm where the privacy budget is not *wasted* on training the teacher. While this is a nice theoretical question, we show in our experiments that DPKD framework is competitive with respect to teachers that are *not trained with DP*.

3.3 Empirical Evaluation of DPKD Algorithm

In this section we perform experiments to evaluate the effectiveness of DPKD algorithm for compressing models.

Teacher and Student Architectures. Our teacher models are pre-trained BERT [1] models, which consists of 12 transformer blocks. The architecture of compressed models consist of 6 transformer blocks, which we refer as $\frac{1}{2}$-BERT.

Tasks and datasets. Following prior work [67, 32, 65], we experiment with the following set of 4 tasks from the GLUE benchmark [59]: MNLI (Multi-Genre Natural Language Inference Corpus), QQP (Quora Question Pairs), QNLI (Stanford Question Answering Dataset) and SST-2 (Stanford Sentiment Treebank).

Training and privacy parameters. In all our experiments, we set the target privacy budget to be bounded by $\epsilon < 4.25$ (across the tasks), with $\delta = \frac{1}{10N}$ where N is the number of samples in the given dataset. We describe the software and hardware specifications in Appendix A.1 and hyperparameter settings in Appendix A.2.

[1] We use Huggingface's `bert-base-uncased`.

Table 1: Comparison between performance of 6-layer $\frac{1}{2}$-BERT student models with random initialization against full 12-layer BERT teacher models. The first row indicates the performance of fine-tuning full teacher model. All our models have the same privacy budget bounded by $\epsilon < 4.25$.

Model	Initialization	Teacher	Training	MNLI	QQP	QNLI	SST-2	Avg
BERT	Pretrained	-	Finetune	77.8	84.7	87.8	90.5	85.2
$\frac{1}{2}$-BERT	Random	-	Finetune	55.1	74.0	59.4	69.7	64.5
$\frac{1}{2}$-BERT	Random	BERT	DPKD	53.9	73.1	59.2	65.4	62.9

Table 2: Comparison between performance of 6-layer $\frac{1}{2}$-BERT student models against full 12-layer BERT teacher models and pre-trained DistilBERT, under various initialization strategies. For every student initialization method, we compare fine-tuning using DPKD algorithm vs full fine-tuning via DPSGD. All our models have the same privacy budget bounded by $\epsilon < 4.25$.

Model	Initialization	Teacher	Training	MNLI	QQP	QNLI	SST-2	Avg
BERT	Pretrained	-	Finetune	77.8	84.7	87.8	90.5	85.2
$\frac{1}{2}$-BERT	Zero-shot (PT)	-	Finetune	71.7	82.4	83.2	82.7	80.0
$\frac{1}{2}$-BERT	Zero-shot (PT)	BERT	DPKD	72.8	82.6	83.0	82.7	80.3
$\frac{1}{2}$-BERT	Zero-shot (FT)	-	Finetune	71.3	81.8	83.4	82.2	79.7
$\frac{1}{2}$-BERT	Zero-shot (FT)	BERT	DPKD	72.3	82.1	82.9	82.6	80.0
DistilBERT	Pretrained	-	Finetune	73.0	84.3	82.8	87.7	81.9
DistilBERT	Pretrained	BERT	DPKD	72.9	83.7	83.0	86.6	81.5

We start with *random initialization* of the student models to see if DPKD algorithm can be effective in transferring the knowledge from the teacher. We compare the performance of our student model trained with DPKD algorithm with performance of directly fine-tuned student via DPSGD and the performance of full teacher model fine-tuned with DPSGD. Our results are summarized in Table 1. The main takeaway from this experiment is:

- There is a large gap in the performance of students trained using DPKD algorithm compared to the teacher when students are randomly initialized. In fact, directly fine-tuning the student model using DPSGD achieves better performance compared to DPKD algorithm.

We conclude that DPKD alone is not enough to transfer the knowledge from teacher models to compressed student models.

3.4 Better Student Models via Zero-shot Initializations

In our desire to train better performing student models, we explore different initialization strategies. We note that initialization is also a form of knowledge transfer. Here we consider two natural zero-shot initialization strategies:

- Zero-shot (PT): Here we initialize the student model using weights of the pre-trained teacher. In particular, we simply initialize the layers of the student model with 6 layers of BERT teacher model. We follow [45] for the choice of layers.
- Zero-shot (FT): Here we initialize the student model using weights of the privately fine-tuned teacher model. As our DPKD requires teacher to be private as well, one can initialize the student model using the weights of the private teacher without incurring any additional privacy cost.

We compare zero-shot initialization strategies against fully pre-trained student model given by Huggingface's compressed BERT model DistilBERT [45]. DistilBERT is trained using KD algorithm with the full BERT model on public data at the *pre-training stage*. We note that DistilBERT has 6

Table 3: Comparison between performance of 6-layer $\frac{1}{2}$-BERT student models under different teacher models in distillation against full 12-layer BERT teacher models and pre-trained DistillBERT. All our models have the same privacy budget bounded by $\epsilon < 4.25$.

Model	Initialization	Teacher	Training	MNLI	QQP	QNLI	SST-2	Avg
BERT	Pretrained	-	Finetune	77.8	84.7	87.8	90.5	85.2
Student	Pretrained	-	Finetune	73.0	84.3	82.8	87.7	81.9
$\frac{1}{2}$-BERT	Zero-shot (PT)	BERT	DPKD	72.8	82.6	83.0	82.7	80.3
$\frac{1}{2}$-BERT	Zero-shot (PT)	$\text{BERT}_{\text{LARGE}}$	DPKD	72.4	81.1	83.1	81.5	79.5
$\frac{1}{2}$-BERT	Zero-shot (PT)	BERT w/out DP	DPKD	74.2	82.9	84.5	83.0	81.1

transformer blocks and hence has the same architecture as $\frac{1}{2}$-BERT. We emphasize that pre-training student model is computationally expensive, and the main aim of this paper is to develop algorithms where one can transfer knowledge from the teacher to students models without resorting to pre-training the student from scratch. However, DistilBERT serves as the gold standard to compare our zero-shot initialization strategies and also sets the benchmark for comparing ideas in Section 5.

Results Our results are summarized in Table 2. For every student initialization method, we also compare fine-tuning using DPKD algorithm vs simply fine-tuning the student via DPSGD. The main takeaway from these experiments are:

- Zero-shot initialization strategies give large performance improvements to student models and come 1-2% of the performance achieved by pre-trained DistilBERT. Somewhat surprisingly, there is not much difference between our two zero-shot initialization strategies.
- Both pre-trained DistilBERT and student models with zero-shot initialization strategies fall short of matching the performance of teacher models.
- Finally, broadly speaking, DPKD algorithm does not give significant performance boost to the student models compared to directly fine-tuning them.

3.5 Better Models are Better Teachers?

Given the results in the previous section, one may wonder if better teachers can help in improving the performance of the students. In this regard we consider two notions of *better teacher*: (1) A larger teacher model in Step 2 of Algorithm 1 (2) A teacher model that is not DP-trained. We note that the final student model in the second case is not DP; however, these experiments allow us to quantify how much performance gains one could have if one were to come up a new framework of implementing KD in DP setting without requiring teacher to be trained with DP.

Our results are summarized in Table 3. These experiments show that DPKD does not benefit significantly from having better teacher models. Furthermore, our proposed framework of implementing KD in DP framework by training both the teacher and student models via DP is only within 0.8% of the doing KD with fine-tuned teacher without DP.

4 Evolving Teacher to Student Models via Pruning

As the previous section presents, the Knowledge Distillation approach to model compression has two main drawbacks in the private world:

- Drop in accuracy: There is a considerable drop in the accuracy between the teacher and the student models.
- Good initialization of students is crucial: The best performance is obtained by students who already have a good initialization; in our experiments, pre-trained DistilBERT mostly achieved the best student performance.

Finding a good initialization can be challenging in practice. Often, the student architectures are chosen to suit the hardware and latency requirements of the application for which the model is being deployed, using neural architecture search [18]. Hence, finding a good initialization for every student architecture via pre-training can be expensive and in most cases impossible. Our zero-shot initialization strategies alleviate this problem to a certain degree, yet fall short of closing the gap between the teacher and the student performances. Moreover, DPKD requires that (i) the teacher is trained with DPSGD and (ii) the student is distilled via DPSGD. This two-step approach creates additional overheads in terms of training. Given these limitations, it is natural to ask: Can we evolve the teacher to a student model while fine-tuning with DPSGD? In this section, we explore an answer to this via *structured and unstructured pruning* with privacy, which allows us to obtain student models that are as good as the teacher models.

4.1 Model Compression via Pruning

Pruning algorithms are a broad class of model compression techniques where one drops the parameters from a model during or after the training process. Many works have shown that eliminating unnecessary parameters of neural networks via pruning can lead to sparser and compressed models that have shorter inference times without loss in performance [30, 25, 24]. For example, in *magnitude pruning*, one of the most widely used pruning techniques, we prune a fraction of parameters with the lowest magnitude. However, there are several pruning strategies, and we refer the readers to [33] for more details and references.

Pruning can be implemented in both *structured and unstructured* ways. In structured pruning, all the pruned weights belong to a single building block of the model. For example, a 6-layer $\frac{1}{2}$-BERT can be obtained by pruning 6 layers from the full BERT model, which consists of 12 transformer blocks. On the other hand, in unstructured pruning, pruned weights may be spread across all the layers of the network. In unstructured pruning, it is possible to obtain a 50% sparse student model while still having all the 12 layers of BERT. Depending on the hardware architectures, inference latency between models with structured and unstructured sparsity could be quite different. However, in this section, *we use sparsity as the main measure of model compression*, which is also well accepted in the community [33, 27].

4.2 Iterative Magnitude Pruning (IMP)

Private pruning techniques we study in this section are based on the Iterative Magnitude Pruning (IMP) method, which is a specific pruning technique proposed in a recent work on Lottery Ticket Hypothesis [19]. The idea behind IMP is rather simple: As we train a deep learning model, after every N iterations we prune an $\alpha\%$ of the weights with the *lowest magnitude*. We repeat this process until we achieve the desired sparsity. Here, both N and α are hyperparameters that need to be tuned. For example, to achieve 50% sparsity, one can perform $5N$ iterations where after every N iterations additional 10% of the weights with the least magnitudes are dropped. As specified IMP produces unstructured sparsity. However, we consider a simple modification of the IMP algorithm to produce structured sparsity as well.

4.3 Structured DPIMP

We first attempt to obtain a student model from the teacher model via a structured IMP technique, using the following modification: During fine-tuning the teacher model with DPSGD, we progressively drop an *appropriately chosen transformer block* from the teacher model at the end of every N iterations. We repeat this process until we obtain the student model with the required sparsity. The layer to drop is chosen using the following heuristic: Let $\alpha > 0$ be a hyperparameter. At the end of N iterations, fix bottom (by magnitude) $\alpha\%$ of *all* model weights, and denote it by W_{min}. For the ith transformer block, let W_i denote the set of model weights belonging to that block. Among all the transformers blocks we find the block i^* that has the highest number of weights from the set W_{min}; Formally, $i^* := \arg\max_i \{W_i \cap W_{\text{min}}\}$, and we prune the transformer layer i^*. We present this in Algorithm 2.

Algorithm 2 Structured DPIMP

Input: Teacher model $\mathcal{T}$, number of layers to prune L, hyperparams α, N and M
Output: Private student model $\mathcal{S}$ with L layers pruned from $\mathcal{T}$

Set $\mathcal{S} := \mathcal{T}$
for $j = 1$ to L **do**
    Fine-tune $\mathcal{S}$ for N iterations with DPSGD
    Set W_{min} consisting of $\alpha\%$ of the remaining model weights with the least magnitude
    Set W_i as the weights of layer i
    Drop the layer i^* from $\mathcal{S}$ satisfying $i^* := \arg\max_i \{W_i \cap W_{\text{min}}\}$
end for
Fine-tune $\mathcal{S}$ for M more iterations with DPSGD
return $\mathcal{S}$

Table 4: Comparing performance of 6-layer $\frac{1}{2}$-BERT student model produced by structured DPIMP with 12-layer BERT teacher model and pre-trained DistillBERT. All our models have the same privacy budget bounded by $\epsilon < 4.25$.

Model	MNLI	QQP	QNLI	SST-2	Avg
BERT	77.8	84.7	87.8	90.5	85.2
DistilBERT	73.0	84.3	82.8	87.7	81.9
$\frac{1}{2}$-BERT	72.9	83.1	82.5	85.7	81.0

Empirical Evaluation We evaluate our structured pruning algorithm with the same setup described in Section 3.3, and provide the hyperparameters in Appendix A.2. We split the privacy budget equally among all the iterations of the algorithm. Our goal is to produce a student model which has $\frac{1}{2}$ as many layers as the full BERT model. Table 4 shows the results for this setting where we compare structured DPIMP to private fine-tuning of the pre-trained DistilBERT and the full BERT model. The main takeaway from this experiment is:

- DP structured pruning algorithm produces a student model that has performance comparable to that of DistilBERT. Further, it avoids the pre-training cost associated with DistilBERT.

4.4 Unstructured DPIMP

The structured pruning produces student models that are as good as DistilBERT; our next target is to produce student models that are as good as the full teacher BERT model. Towards that, we explore unstructured pruning techniques, in particular differentially private version of the IMP algorithm. As we fine-tune the deep learning models using DPSGD, after every N iterations we prune increments of $\alpha\%$ of the weights with lowest magnitude. The remaining weights are then reset to the original pre-trained initialization. (We do this step to establish connections to Lottery Ticket Hypothesis, see below.) We repeat this process until we achieve the desired sparsity. We note that the initialization of the final student model's non-zero parameters at the end of the pruning step are still non-private, even though weights correspond to pre-trained model weights. This is due to the fact that pruned parameters are found during the fine-tuning process, which makes use of the private dataset. Therefore, the whole process must be performed with DPSGD to produce a private student model. Finally, we perform additional fine-tuning of the pruned student model by using DPSGD for M more iterations. We call the private variation of this procedure Unstructured DPIMP. We formally present this process in Algorithm 3.

Empirical Evaluation We refer to the student model produced by Algorithm 3 as SparseBERT and indicate the sparsity level in brackets. We allocate the privacy budget equally among all iterations of Algorithm 3. Table 5 summarizes our experiments on unstructured pruning via DPIMP. We compare the performance of our student model to pre-trained DistilBERT (whose parameter count is the same as our final student model) and the full-sized BERT teacher model. The main takeaways are:

Algorithm 3 Unstructured DPIMP

Input: Model $\mathcal{T}$ and the sparsity level $S\%$, hyperparams α, N and M
Output: Private Student model $\mathcal{S}$ satisfying the sparsity requirements
1: Set $\mathcal{T}' := \mathcal{T}$
2: **for** $i = 1$ to $\lceil (S/\alpha) \rceil$ **do**
3: Fine-tune $\mathcal{T}'$ for N iterations with DPSGD
4: Prune $(\alpha \times i)\%$ of weights with the lowest magnitude from $\mathcal{T}'$
5: Reset the non-zero weights of $\mathcal{T}'$ to the original $\mathcal{T}$
6: **end for**
7: Fine-tune $\mathcal{T}'$ for M iterations with DPSGD
8: **return** $\mathcal{S} := \mathcal{T}'$

Table 5: Comparing performance of SparseBERT student model produced by unstructured DPIMP with 12-layer BERT teacher and pre-trained DistillBERT. All models have the same privacy budget $\epsilon < 4.25$.

Model	MNLI	QQP	QNLI	SST-2	Avg
BERT	77.8	84.7	87.8	90.5	85.2
DistilBERT	73.0	84.3	82.8	87.7	81.9
SparseBERT (50%)	76.8	84.0	85.7	87.6	83.5

- DPIMP produces a student model that has superior performance compared to DistilBERT.
- The average performance of DPIMP is within 1.7% of the full BERT model. We conclude that unstructured pruning techniques are more effective in closing the gap between the teacher and the student models in the private world. Moreover, pruning methods are computationally cheaper as student models do not require pre-training on public data.
- DPIMP algorithm as described in Algorithm 3 at the end of line 6 finds sparse subnetworks in the pre-trained BERT model that can be fine-tuned to obtain nearly matching teacher performance. This is similar to Lottery Ticket Hypothesis work for BERT networks [10], although in the private world performance of the student network is not matching that of the teacher.

5 Conclusions and Future Directions

In this paper we initiated the study of differentially private model compression via Knowledge Distillation and Pruning, and gave frameworks for implementing both. Our work shows that one can obtain student models whose performance comes within 2% of the full teacher models while having 50% fewer parameters, which can lead to significant reduction in memory and improve inference time. We believe that our work takes the first step in the intersection of private training and model compression techniques, and opens a whole in direction with plenty of interesting and important problems. We highlight some of them here.

- *Better Algorithms For Model Compression*: While we gave natural DP adaptations of KD and pruning algorithms, we believe that there are plenty of new techniques to explore.
- *Better Accounting*: In all our results, we chose very simple strategies for allocating privacy budget across various steps of training and pruning. Theoretical innovations in the form of better accounting can further improve the performance of student models.
- *Lottery Tickets for DP-training?* Both in our experiments on KD and in pruning, we see that models that have "good initialization" lead to dramatic improvements in performance. Moreover our DPIMP algorithm finds sparse subnetworks in pretrained BERT models at 50% sparsity that can be fine-tuned to obtain *nearly* matching teacher performances. This raises an intriguing question akin to Lottery Ticket Hypothesis: Are there good initialization of models where dynamics of DPSGD is similar to SGD? We believe that this is an exciting research direction both from a theoretical point of view and also its implications to practical private training.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM. 2.1, B

[2] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-scale differentially private BERT. *arXiv preprint arXiv:2108.01624*, 2021. B

[3] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 464–473, Washington, DC, USA, 2014. IEEE Computer Society. 2.1, B

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[5] C Bucilua, R Caruana, and A Niculescu-Mizil. Model compression, in proceedings of the 12 th acm sigkdd international conference on knowledge discovery and data mining. *New York, NY, USA*, 2006. 3

[6] Qingqing Cao, Harsh Trivedi, Aruna Balasubramanian, and Niranjan Balasubramanian. DeFormer: Decomposing pre-trained transformers for faster question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4497, 2020. B

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, USENIX Security '19, pages 267–284. USENIX Association, 2019. 1, B

[8] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021. 1

[9] Kamalika Chaudhuri and Daniel Hsu. Sample complexity bounds for differentially private learning. In *Proceedings of the 24th Annual Conference on Learning Theory*, COLT '11, pages 155–186, 2011. B

[10] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020. 1.1, 4.4, B

[11] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022. 1

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[13] Thomas G. Dietterich. Ensemble methods in machine learning. In *MULTIPLE CLASSIFIER SYSTEMS, LBCS-1857*, pages 1–15. Springer, 2000. 1.1

[14] Lifang Ding and Yujiu Yang. SDSK2BERT: Explore the specific depth with specific knowledge to compress BERT. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, page 420–425, Aug 2020. B

[15] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '06, pages 486–503, Berlin, Heidelberg, 2006. Springer. 2.1

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006. Springer. 1, 2.1

[17] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. 3.2

[18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 4, B

[19] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 1.1, 4.2

[20] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021. B

[21] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *arXiv preprint arXiv:2106.02848*, 2021. 2.1, 3.2

[22] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. 3

[23] Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*, 2019. B

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015. 1, 4.1

[25] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992. 1, 4.1

[26] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 3, 3.1

[27] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. 4.1, B

[28] Shlomo Hoory, Amir Feder, Avichai Tendler, Alon Cohen, Sofia Erell, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. Learning and evaluating a differentially private pre-trained language model. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, PrivateNLP '21, pages 21–29, 2021. B

[29] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174, 2020. B

[30] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989. 1, 4.1

[31] Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3187–3199, 2020. B

[32] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *Proceedings of the 10th International Conference on Learning Representations*, ICLR '22, 2022. 1, 3.3, B

[33] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *arXiv preprint arXiv:2101.09671*, 2021. 4.1

[34] Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. MKD: A multi-task knowledge distillation approach for pretrained language models. *arXiv preprint arXiv:1911.03588*, 2019. B

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 1

[36] Lingjuan Lyu and Chi-Hua Chen. Differentially private knowledge distillation for mobile analytics. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1809–1812, 2020. 1.1

[37] Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Quanlu Zhang, Yaming Yang, Yunhai Tong, and Jing Bai. LadaBERT: Lightweight adaptation of BERT through hybrid model compression. In *Proceedings of the 28th International Conference on Computational Linguistics*, page 3225–3234, 2020. B

[38] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR '18, 2018. B

[39] Subhabrata Mukherjee and Ahmed H. Awadallah. Distilling BERT into simple neural networks with unlabeled transfer data. *arXiv preprint arXiv:1910.01769*, 2019. B

[40] Subhabrata Mukherjee and Ahmed H. Awadallah. XtremeDistil: Multi-stage distillation for massive multilingual models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 2221–2234, 2020. B

[41] Matan Ben Noach and Yoav Goldberg. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889, 2020. B

[42] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. 1.1

[43] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019. 1

[44] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 3

[45] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 2.1, 3.4, B

[46] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017. 1

[47] Kaitao Song, Hao Sun, Xu Tan, Tao Qin, Jianfeng Lu, Hongzhi Liu, and Tie-Yan Liu. LightPAFF: A two-stage distillation framework for pre-training and fine-tuning. *arXiv preprint arXiv:2004.12817*, 2020. B

[48] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*, GlobalSIP '13, pages 245–248, Washington, DC, USA, 2013. IEEE Computer Society. 2.1

[49] Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. In *International Conference on Learning Representations*, 2020. B

[50] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4314–4323, 2019. B

[51] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 830–844, New York, NY, USA, 2021. Association for Computing Machinery. B

[52] Thierry Tambe, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, and Gu-Yeon Wei. Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. B

[53] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. B

[54] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv preprint arXiv:1903.12136*, 2019. B

[55] Zhiliang Tian, Yingxiu Zhao, Ziyue Huang, Yu-Xiang Wang, Nevin Zhang, and He He. Seqpate: Differentially private text generation via knowledge distillation. 2021. 1.1

[56] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data). In *Proceedings of the 9th International Conference on Learning Representations*, ICLR '21, 2021. B

[57] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*, 13, 2019. B

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017. 1

[59] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018. 3.3

[60] Moshe Wasserblat, Oren Pereg, and Peter Izsak. Exploring the boundaries of low-resource BERT distillation. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, page 35–40, 2020. B

[61] Xing Wu, Yibing Liu, Xiangyang Zhou, and Dianhai Yu. Distilling knowledge from pre-trained language models via text smoothing. *arXiv preprint arXiv:2005.03848*, 2020. B

[62] Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. Nas-bert: Task-agnostic and adaptive-size bert compression with neural architecture search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1933–1943, New York, NY, USA, 2021. Association for Computing Machinery. B

[63] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763, 2019. B

[64] Yichun Yin, Cheng Chen, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Autotinybert: Automatic hyper-parameter optimization for efficient pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5146–5157, 2021. B

[65] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, ICLR '22, 2022. 1, 3.3, B

[66] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1.1

[67] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. In *Proceedings of the 38th International Conference on Machine Learning*, ICML '21. JMLR, Inc., 2021. 3.3

[68] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8bit BERT. In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019. B

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] , see Sections 1.1 and 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] , we are studying and evaluating differentially private model compression.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A] , See Section 3. The paper is empirical, we only recite existing theoretical work that we build our experiments on.

(b) Did you include complete proofs of all theoretical results? [N/A] We are not introducing new theoretic notions.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] , we have included the dataset names (publicly available) and all of our hyperparameters and package versions. See Sections 3.3, A.1, and A.2.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] , See Sections 3.3, A.1, and A.2.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] .

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] , see Section A.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]