Meta-Analysis of DP-SGD across Learning Tasks and parameter-efficient fine-tuning methods using distilBERT

Faizel Khan
University of Minnesota
Minneapolis, Minnesota
khanx370@umn.edu

Max Scheder-Bieschin
University of Minnesota
Minneapolis, Minnesota
sched088@umn.edu

ABSTRACT

The application of differential privacy (DP) in transformer-based large language models (LLMs) such as BERT and GPT has become essential for preserving user privacy while maintaining model performance during transfer learning. However, the added noise and computational overhead associated with DP continue to pose challenges. While parameter-efficient fine-tuning methods and optimal clipping of DP-SGD with appropriate hyperparameters have demonstrated potential [43], optimizing weights, biases, and inner layers in Transformers using DP remains under-explored. In this study, we conduct a comparative analysis on a pre-trained LLM, DistilBERT [31], focusing on the application of four parameter-efficient fine-tuning methods (Gradual Unfreezing [18], Adapter Modules [3], Diff Pruning [14], and BitFit [45]) in conjunction with differentially private stochastic gradient descent (DP-SGD)[36]. We assess the performance of these fine-tuning approaches on three distinct GLUE benchmarks (MNLI, QQP, and SST-2) [39] and observe that the impact of DP-SGD varies widely between benchmarks and fine-tuning methods. Our preliminary analysis indicates that Gradual Unfreezing yields the best performance retention, particularly on the QQP benchmark. This investigation aims to provide greater insight into the interaction between DP-SGD and fine-tuning methods, and guide future research in developing more effective strategies for private LLM fine-tuning.

PVLDB Reference Format:
Faizel Khan and Max Scheder-Bieschin. Meta-Analysis of DP-SGD across Learning Tasks and parameter-efficient fine-tuning methods using distilBERT . PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Natural language processing (NLP) has achieved remarkable success across many tasks, including sentiment analysis, inference, and paraphrasing. LLMs like BERT and GPT have significantly advanced NLP by utilizing extensive public pre-training datasets to create robust general models [33] that can be subsequently fine-tuned on smaller, task-specific datasets. While these models greatly benefit from the wealth of available domain-specific data, they can encounter privacy concerns when the fine-tuning data is sourced from datasets containing sensitive information such as lists of company customers, their behaviors, or financial information.

One of the most popular and well studied privacy-conserving techniques is to add DP [11] to the stochastic gradient descent algorithm. This is formally known as DP-SGD or Differentially-Private Stochastic Gradient Descent [36], and is an adaptation of the widely-used minibatch stochastic optimization technique in deep learning designed to incorporate DP. This modification ensures that sensitive information in the training data remains protected while still enabling the learning process. By adding carefully calibrated noise (randomization) to the gradients during the optimization process, DP-SGD helps preserve the privacy of individual data points, ensuring that the model's outputs do not reveal any specific information about the training data.

In this way, DP-SGD strikes a balance between effective learning and maintaining data privacy. This approach allows for flexibility in the privacy guarantees of the models with a trade-off between protecting sensitive information in the training data (privacy) and maintaining the model's performance (utility). When applying DP-SGD to the training process, noise is introduced to mask individual data points' influence on the model's parameters. However, this noise can also degrade the model's performance, as it may obstruct the learning of meaningful patterns from the data, introducing a substantial performance overhead [1, 4, 6].

This approach has seen adoption in many areas of NLP and the impact that it has on model performance and training overhead varies by application and model architecture [2, 27]. This impacts especially profound when the number of parameters being trained in large. Therefore, Yu et al. [44] introduces a meta-framework to introduce differential privacy while fine-tuning LLMs. They explore parameter-efficient techniques like LoRA [19], and Compacter [25] to fine-tune their models, initially trained on general datasets, for domain-specific applications where privacy is crucial. By introducing parameter-efficient techniques, their architecture focuses on a smaller set of parameters or layers for fine-tuning. With this approach the amount of noise added due to DP can be better managed, and it allows for a more optimal balance between privacy protection and the model's performance, leading to a more accurate model with strong privacy guarantees while also. These parameter-efficient techniques, also achieved success in minimizing the training overhead associated with DP-SGD.

However, the introduction of parameter-efficient fine-tuning techniques also introduces an additional layer of complexity in

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

evaluating the performance implications of incorporating DP-SGD in NLP models. In non-private settings, there has been a lot of work in developing advanced parameter-efficient methods that minimize the layer complexities and leverage the weights and biases [3, 14, 18, 19, 25, 28, 45]. This area of research has not seen as much activity in the a private setting however. To address this research gap, we carry out a comparative assessment of four parameter-efficient fine-tuning techniques, examining their performance in combination with DP-SGD [36] on a pre-trained LLM, DistilBERT [31].

In this paper, we have selected the following parameter-efficient fine-tuning techniques:

(1) Gradual Unfreezing [18]: This technique incrementally unfreezes the model, starting from the last transformer layer, to fine-tune both transformer-based and output-layer parameters while mitigating the risk of overfitting. Enhanced by a slanted triangular learning rate and component-wise gradient clipping, it stabilizes the fine-tuning process and optimizes model accuracy.
(2) Adapter Modules [3]: A technique that introduces additional lightweight adapter modules within the frozen transformer layers, which require only around 3% of the original model parameters, allowing the model to learn task-specific information while keeping the pre-trained transformer parameters fixed, while privately fine-tuning a LLM.
(3) Diff-Pruning [14]: A method aimed at reducing the number of weights that change during the fine-tuning process by separating fixed pre-trained weights and task-specific "diff" values, focusing on transformer-based weight parameters. It extends the base model through a task-specific difference vector and enforces an L0-norm penalty on this vector. The technique approximates L0 regularization through stochastic gates, enabling differentiable optimization.
(4) BitFit [45]: A sparse fine-tuning technique that focuses on optimizing only the bias parameters in the transformer layers, significantly reducing the number of parameters to be updated during fine-tuning. Despite involving a small fraction of the network's parameters, BitFit shows promising results on multiple tasks.

To evaluate the effectiveness of these techniques, we use several GLUE tasks [39], including MNLI, QQP, and SST-2, as benchmarks for assessing private model training performance. We explore the impact of these techniques on the privacy-utility trade-off, model performance, and training overhead across the specified GLUE tasks. Our experiments and analyses provide valuable insights into the benefits and limitations of each technique in the context of privacy preservation. Furthermore, our findings contribute to a better understanding of the intricate balance between privacy, efficiency, and performance in NLP models, paving the way for more effective, secure, and privacy-aware applications of LLMs in real-world scenarios.

2 BACKGROUND

2.1 GLUE

The General Language Understanding Evaluation (GLUE) [39] benchmark serves as an essential collection of tasks designed to assess NLP models' performance on various natural language understanding aspects. By encompassing nine tasks, including single-sentence tasks, similarity and paraphrase tasks, and inference tasks, GLUE encourages the development of models capable of transferring knowledge across multiple tasks. This comprehensive benchmark enables researchers to systematically compare different models' performance and track progress in the NLP field.

Among the tasks incorporated in the GLUE benchmark, we focus on MNLI, QQP, and SST-2 as combined they encompass all three dimensions of language understanding: single-sentence analysis, similarity and paraphrase detection, and inference tasks. SST-2 (Stanford Sentiment Treebank) focuses on sentiment analysis, requiring NLP models to predict the sentiment of sentences derived from movie reviews. This task is crucial for understanding how well models can process subjective information in the text [35]. Another task, QQP (Quora Question Pairs), evaluates models' ability to recognize semantically equivalent question pairs, reflecting their understanding of sentence semantics and paraphrasing. This ability is vital for applications such as question-answering systems, search engines, and chatbots [34]. Lastly, the MNLI (Multi-Genre Natural Language Inference Corpus) task involves predicting the relationship between a premise and a hypothesis sentence, assessing NLP models' reasoning skills, and understanding sentence relationships [41].

The GLUE benchmark has gained widespread adoption within the NLP research community, with its leaderboard showcasing various state-of-the-art models' performance. It's popularity has led to many spin-off benchmarks [38] [21] [40]. Nevertheless it remained the preeminent benchmark for NLP tasks.

2.2 Differential Privacy

Researchers have been developing large machine learning models that rely on private datasets for training purposes. However, it has been demonstrated that this approach can be insecure, as specific training examples may be extracted, potentially compromising the privacy of the data involved[7]. Moreover, adversarial training data extraction attacks can potentially recover individual training examples. Some studies have even managed to extract personally identifiable information from language models [8]. These attacks and other leaks of sensitive data can be combated by training models to guarantee Differential Privacy (DP) [11]. Differential privacy ensures that the influence of any individual training sample is limited to prevent it from later being extracted.

DP is a mathematical framework that originated in a 2006 paper by [11] to ensure the privacy of individual data points in a dataset while enabling aggregate analysis. The main idea is to introduce noise into the data or computation process so that the results of any analysis do not reveal information about individual data points. A mechanism M satisfies ϵ-DP if for any pair of neighboring datasets D and D', and for any possible output x, the following condition holds: $P[M(D) = x] \leq exp(\epsilon) \cdot P[M(D') = x]$. The amount of noise added depends on the sensitivity of the function being computed and the desired privacy level, represented by the privacy budget (ϵ). A smaller ϵ offers stronger privacy, while a larger ϵ allows for more accurate results at the cost of reduced privacy.

In addition to the privacy budget, other important parameters include the failure probability, which is a small positive constant representing the probability of privacy failure, and the choice of noise distribution, such as Laplace or Gaussian distributions. The composition theorems, sequential composition, and parallel composition enable the analysis of cumulative privacy loss when multiple differentially private mechanisms are applied sequentially []. DP can be applied globally, where noise is added to the output of a function computed over the entire dataset, or locally, where noise is added to each individual's data before computing the function. In addition to the privacy budget, other important parameters include the failure probability, which is a small positive constant representing the probability of privacy failure, and the choice of noise distribution, such as Laplace or Gaussian distributions. The composition theorems, sequential composition, and parallel composition enable the analysis of cumulative privacy loss when multiple differentially private mechanisms are applied sequentially [20]. DP can be applied globally, where noise is added to the output of a function computed over the entire dataset, or locally, where noise is added to each individual's data before computing the function.

DP has gained popularity in recent years due to its robust privacy guarantees and its adaptability to various applications, such as statistical databases and machine learning algorithms [11]. By carefully selecting the privacy budget and considering other important parameters, DP provides formal privacy guarantees for the protection of individual privacy, ensuring that sensitive information remains confidential while still allowing for useful data analysis. We use this method through DP-SGD optimizer, explained in the next section.

2.3 DP-SGD

Differential Privacy Stochastic Gradient Descent (DP-SGD) is a differentially private variant of the popular Stochastic Gradient Descent (SGD) optimization algorithm used in training machine learning models, particularly deep neural networks. DP-SGD was introduced as a method to provide privacy guarantees during the model training process while maintaining a high level of utility (CITE: by Martin Abadi et al. in a 2016 paper). DP-SGD works by applying noise to the gradients computed on mini-batches of the training data during each training step, thereby ensuring that the learned model does not memorize or leak individual data points.

The key components of DP-SGD include the gradient clipping step and the addition of noise to the clipped gradients. Gradient clipping limits the sensitivity of the gradient updates by bounding the L2 norm of each gradient vector, which in turn helps control the amount of noise required to achieve a given privacy level. The noise is typically drawn from a Gaussian or Laplace distribution, and the scale of the noise is determined by the desired privacy budget (ϵ) and the sensitivity of the function.

Important parameters in DP-SGD include the clipping threshold, the batch size, the noise scale, and the privacy budget (ϵ). The clipping threshold determines the maximum L2 norm for the gradients, while the batch size and noise scale influence the privacy-utility trade-off. Larger batch size can lead to better utility, but it may also require more noise to achieve the same privacy level. Similarly, increasing the noise scale can improve privacy but may affect model accuracy.

DP-SGD has gained prominence as it allows for the training of machine learning models with robust privacy guarantees. By carefully selecting the parameters and incorporating DP into the optimization process, DP-SGD enables the development of models that respect individual privacy while maintaining a high level of utility for various applications, such as medical data analysis and personalized recommendation systems.

2.4 Parameter-efficient Fine-tuning

LLMs such as BERT are pre-trained from massive and diverse public data sets. These models can then be specialized or fine-tuned for specific tasks using much smaller, domain-specific data sets. An example of a fine-tuning application would be to take a general BERT model and train it on an online store's users purchasing histories and train it to recommend products.

There are a number of challenges with using LLMs. Due to the size of many BERT or GPT models, it's difficult to fine-tune a full model as they contain 100s of millions of parameters. With privacy guarantees added through DP-SGD, this task becomes even more daunting. Not only does DP-SGD impact the required computational power, but as mentioned above, the noise in the model grows with the number of parameters and can overwhelm the signal in larger models. Therefore it is critical to find ways to effectively fine-tune models using only a small percentage of their total parameters.

Parameter efficient fine-tuning is an active area of research as it benefits both private and non-private models. It is not understood exactly why some proposed methods are so effective in obtaining model accuracy with only a small percentage of parameters being updated [9]. Identifying which parameters significantly impact model decision making is beneficial for developing new parameter-efficient fine-tuning methods, and for better understanding which applications will benefit most from certain parameter-efficient approaches. [13] [5]

3 OUR STUDY

3.1 Model Implementation

In this study, we expand upon the meta-framework proposed by [44] for privately fine-tuning LLMs. The core idea of this framework is to exploit a small fraction of the original number of parameters in pre-trained language models and fine-tune only these selected parameters using differentially private stochastic gradient descent (DP-SGD), while keeping all or most of the original pre-trained model parameters frozen.

Fine-tuning just a small fraction of parameters with DP-SGD might introduce excessive noise due to the privacy-preserving mechanism. However, this noise will not impact the pre-trained parameters in most directions. As a consequence, the model parameters are constrained to remain within a specific manifold, even in the presence of noise in the parameter updates.

This key property of fine-tuning only a small fraction of parameters is crucial in ensuring that the model maintains stability and does not diverge, despite the added noise required for privacy preservation. By leveraging this approach, the framework achieves

a balance between maintaining the utility of the model and protecting the privacy of individual data points during the fine-tuning process.

In the following section, we discuss the specific parameter-efficient techniques employed for fine-tuning within our study.

3.2 Fine-Tuning Techniques

After conducting a comprehensive literature review, we identified four distinct parameter-efficient fine-tuning techniques to incorporate into our study. Our selection criteria aimed to encompass a variety of efficiency levels and diverse parameter-efficient approaches for fine-tuning. In our analysis, we focused on transformer-based parameters, such as weights and biases, as well as output layer parameters, to gain a broader understanding of their impact on the fine-tuning process.

Transformer-based parameters play a crucial role in determining the overall performance of LLMs like BERT. By selectively fine-tuning these parameters, it is possible to achieve better performance and reduce the amount of computational resources required during the fine-tuning process as compared to full fine-tuning [15]. Weights and biases in the transformer layers capture the model's general knowledge acquired during pre-training, and fine-tuning them can help adapt the model to the specific task at hand.

Output layer parameters, on the other hand, are essential for mapping the model's internal representations to the final task-specific predictions. Fine-tuning these parameters allows the model to produce more accurate and dimensionally correct, predictions for the target task, thereby enhancing its overall performance.

Our selected parameter-efficient fine-tuning techniques consist of the following:

3.2.1 Gradual Unfreezing. The first Fine-tuning technique implemented is known as Gradual Unfreezing. This technique works by incrementally unfreezing the model starting from the last transformer layer, as this has been shown to contain the most specific knowledge of the model [42]. Gradual Unfreezing focuses on both transformer-based parameters and output-layer parameters, ensuring that the model gains task-specific knowledge while maintaining the general knowledge acquired during pre-training.

By combining Gradual Unfreezing with DP-SGD, we can create a privacy-preserving fine-tuning process. Unfreezing only a limited number of layers one at a time can help reduce the risk of overfitting and maintain a balance between preserving privacy and achieving good performance. Gradually unfreezing layers ensures that the noise added during the DPSGD process is distributed across multiple layers, thus minimizing the impact of the noise on the model's overall performance [30].

We follow the approach of Howard and Ruder [18] and start with only the last layer unfrozen, train for one epoch, then unfreeze another layer. We deviate from their approach due to the use of the smaller DistilBERT architecture and then train for 3 epochs with the top two layers unfrozen because entire layers of the transformer architecture need to be unfrozen this approach, while less than a full model training, requires a significant number of parameters to be updated as compared to the other approaches we included in our study.

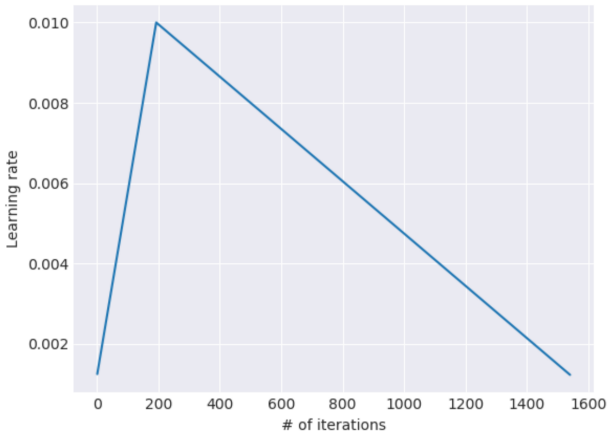

Figure 1: An example of the slanted learning rate implementation. The learning rate begins at a baseline value before rapidly increasing with the number of training iterations before more slowly settling back down. The rate of change caries with the total number of training iterations.

In our implementation, we utilize the slanted triangular learning rate as used in ULMFit. This approach adjusts the learning rate as a function of the number of iterations similar to as shown in Fig. 1. The intuition behind this learning rate schedule is that we want the model to find a decent parameter space relatively quickly in the training regime before refining the model parameters more carefully.

Gradual Unfreezing is the least parameter-efficient method that we implement. We selected it due to its middle-ground between true parameter-efficient fine-tuning methods and a full fine-tuning approach.

3.2.2 Adapter Modules. Adapters are another parameter-efficient fine-tuning approach where the original weights and biases of the model are kept frozen. Each transformer layer is then modified with the insertion of an adapter module. We selected this fine-tuning technique as the authors of the original adapter model paper claim performance within 0.4% of full fine-tuning with 96.4% fewer parameters being trained [3].

There are many different adapter module architectures that can be implemented as long as the number of outputs for the adapter module is equal to the number of inputs. The majority of architectures implement bottleneck architecture to limit the number of parameters needing training. The adapters work by first projecting the original features into lower dimensional space, applying a non-linearity, and then projecting the lower dimensional space back into the original dimensions. The down-projection is a parameter that can be modified with a trade-off between performance and parameter efficiency. In our implementation, we use a reduction factor of 16 with a ReLU non-linearity function.

In addition, the placement of the adapter module can vary. This includes placing adapter modules after both the multi-head attention and feed-forward block in each transformer layer [17], placing

an adapter module only after the feed-forward block in each transformer layer [29] or placing adapter modules in parallel to the original transformer layers [15]. For our implementation, we leverage the approach proposed by **?**] where the adapter modules is placed only after the feed-forward block in each transformer layer.

A way of boosting the performance of adapter modules is to train new layer normalization parameters. This can be done both before and/or after the adapter module(s) and builds upon the work of conditional batch normalization [10]. Our implementation balances performance vs parameter efficiency by performing pre-trained layer normalization and residual connection after the adapter module [28].

3.2.3 Diff Pruning. Diff-Pruning is a method aimed at reducing the number of weights that change during the fine-tuning process of pre-trained language models like BERT without modifying the pre-trained model weights. By concentrating on the transformer-based weight parameters, Diff-Pruning allows for a more controlled and efficient fine-tuning process. This can be particularly beneficial when combined with DP-SGD, as the reduced number of updated parameters can lead to less noise being added during the fine-tuning process, resulting in improved privacy preservation and potentially better model performance.

It focuses on separating fixed pre-trained weights (θ_{pre}) and task-specific diff values (δ_τ) as shown in eqn 1. Initially, all δ_τ weights are set to 0, and the goal is to limit the number of non-zero δ_τ values during training. This constraint is expressed as an L0 penalty in the loss function.

$$\theta_\tau = \theta_{pre} + \delta_\tau \tag{1}$$

To tackle the non-differentiability of the L0 penalty term, the authors employ a technique called differentiable L0 regularization. They use the reparameterization trick, constructing a mask (z) via a parametrized distribution called the hard-concrete distribution as shown in eqn 2. The mask (z) is then incorporated into the model parameters, and the L0 penalty term is calculated accordingly. At inference time, the random part of z is removed, making it deterministic.

$$\delta_\tau = z_\tau \odot w_\tau, \quad z_\tau \in \{0,1\}^d, \quad w_\tau \in \mathbb{R}^d \tag{2}$$

The Diff-Pruning training procedure consists of extending the loss function with the L0 penalty and updating the weights and alpha values during the backward pass. After training for some epochs, magnitude pruning is applied to the diff values, keeping only the top percentage of weights based on their magnitude. The model is then trained for additional epochs to recover any lost performance due to pruning.

By incorporating Diff-Pruning into a typical PyTorch model pipeline, it is possible to control the exact degree of sparsity in the δ values. Though newer techniques like FISH Mask [37] have shown better performance, Diff-Pruning remains an elegant approach to sparsifying large models and has the potential for further research.

3.2.4 BitFit. BitFit [45] is a sparse fine-tuning technique that focuses on optimizing only the bias parameters of a neural network instead of fine-tuning the entire model. As outlined in the original paper, the authors formulated a method specifically designed for models pre-trained on BERT. Consequently, this approach is readily adaptable to DistilBERT, which we employed in our experiment. This novel approach leaves the weight matrix W of each linear or convolutional layer unchanged and selectively fine-tunes the bias vector b. By doing so, BitFit significantly reduces the number of parameters to be updated during fine-tuning. Therefore, it allows for a more efficient and effective fine-tuning process. When combined with DP-SGD, updating only the biases can lead to better privacy preservation while still maintaining competitive performance. By updating fewer parameters, the amount of noise added to the model during the fine-tuning process can be reduced, potentially resulting in improved privacy and performance.

The BERT encoder is composed of L layers. Each of these layers, denoted by L, begins with M self-attention heads. Every self-attention head, represented by (m, l), possesses key, query, and value encoders. Each of these encoders takes the shape of a linear layer.

$$Q^{m,l}(x) = W_q^{m,l}x + b_q^{m,l} \tag{3}$$

$$K^{m,l}(x) = W_k^{m,l}x + b_k^{m,l} \tag{4}$$

$$V^{m,l}(x) = W_v^{m,l}x + b_v^{m,l} \tag{5}$$

In this case, x represents the output from the preceding encoder layer. For the first encoder layer, x stands for the output from the embedding layer. These outputs are then integrated using an attention mechanism that does not necessitate the introduction of new parameters.

$$h_1^l = att(Q^{1,l}, K^{1,l}, V^{1,l}, ..., Q^{m,l}, K^{m,l}, V^{m,l}) \tag{6}$$

$$h_2^l = Dropout(W_{m1}^l \cdot h_1^l, b_{m1}^l) \tag{7}$$

$$h_3^l = g(W_{LN_1}^l \odot \frac{(h_2^l + x) - \mu}{\sigma} + b_{LN_1}^l) \tag{8}$$

$$h_4^l = GELU(W_{m2}^l \cdot h_3^l, b_{m2}^l) \tag{9}$$

$$h_5^l = Dropout(W_{m3}^l \cdot h_4^l, b_{m3}^l) \tag{10}$$

$$out^l = g(W_{LN_2}^l \odot \frac{(h_5^l + h_3^l - \mu}{\sigma} + b_{LN_2}^l) \tag{11}$$

Further, it involves specifying the bias parameters to be optimized within the model's named parameters. By freezing all parameters $W^{(\cdot)}$ and $g^{(\cdot)}$ and fine-tuning only the additive bias terms $b^{(\cdot)}$, BitFit achieves transfer learning performance comparable to or even better than fine-tuning the entire network. It is also possible to fine-tune only a subset of the bias parameters, such as those associated with the query and the second MLP layer (only $b_q^{(\cdot)}$ and $b_{m2}^{(\cdot)}$), and still achieve accuracies that rival full-model fine-tuning. Therefore, this approach has practical utility for deploying multi-task fine-tuned models in memory-constrained environments and paves the way for research on the role of bias terms in pre-trained networks and the dynamics of the fine-tuning process.

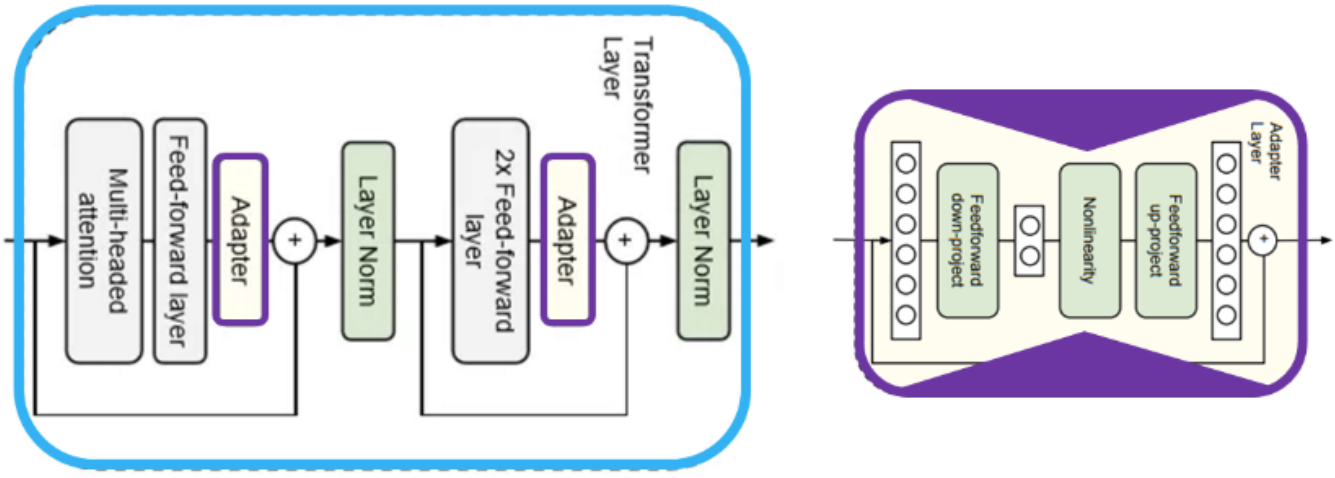

Figure 2: An example of the slanted learning rate implementation. The learning rate begins at a baseline value before rapidly increasing with the number of training iterations before slowly settling back down.

By updating just about 0.05% of the model parameters, BitFit demonstrates similar or better performance than full fine-tuning in low and medium-data scenarios for BERT models with less than 1 billion parameters. However, when the method is applied to larger networks such as T0-3B [22, 32], BitFit significantly under-performs full fine-tuning.

4 EXPERIMENTAL SETUP

Our experiments were carried out on high-performance computing hardware to ensure the effective implementation of our fine-tuning techniques. We utilized NVIDIA A100 Tensor Core GPUs for running all experiments and obtaining our throughput performance results. The choice of these advanced GPUs ensured optimal processing power and speed, crucial for the complex computations involved in fine-tuning LLMs.

For all our training runs, we standardized our batch size to 128. We also conducted tests with smaller batch sizes of 64 and 32, but these resulted in lower accuracy rates compared to when a batch size of 128 was used. Our observation aligns with the evidence presented in recent studies [2, 12, 16, 26, 27], which indicate that DP-SGD tends to yield superior results when utilized with larger batch sizes.

The hyperparameters for all our training runs were also kept consistent to ensure comparability across different techniques. Specifically, we used a learning rate of 5e-3 and a total of 20 epochs for each run. To manage the trade-off between privacy and utility in our DP-SGD implementation, we set a target epsilon (ϵ) of 0.5 and a target delta (δ) of 1e-5. These hyperparameters were chosen to strike a balance between model accuracy and privacy preservation, in line with our objective of evaluating parameter-efficient fine-tuning techniques for their potential application in privacy-sensitive NLP tasks.

The models we implemented were built on the PyTorch framework and leverage tools published by the company HuggingFace. For implementing differential privacy we utilized Opacus and Pyvacy. Where possible we kept our dataset creation and training pipelines the same for ease of replication and troubleshooting.

5 RESULTS

The implementations of the different parameter-efficient fine-tuning techniques against each selected task were analyzed against two metrics: accuracy and the associated training time. For training time, we report the time taken to complete one epoch. For gradual unfreezing this value was taken for the training of the second and final unfrozen layer as training time increases as more layers are unfrozen. We view these results in absolute terms as well as in change due to the addition of differential privacy.

5.1 Accuracy

In Fig. 3, we see a side-by-side comparison of the accuracy of the different fine-tuning techniques across tasks, both with and without DP-SGD. We can immediately see that the accuracy of our models generally decreases with the addition of privacy guarantees. This is expected due to the randomization added by the DP-SGD algorithm. We also see that the model accuracy aligns with what is seen in literature where it is lowest for MNLI and highest for SST. It is unclear at this time why the performance of Adapter module fine-tuning performs so poorly for the SST dataset, however it is important to note that performance does not correlate with the size of the datasets.

In Fig. 4 we see the %change in accuracy between the models with and without DP-SGD. This perspective highlights some additional insights. We can clearly see that MNLI, the inference task dataset has the biggest accuracy drop when DP-SGD is added, whereas SST, the sentiment task, and QQP see a lesser, similar, drop. It is unclear as to why this is. One theory is that MNLI has 3 labels (neutral, contradiction, and entailment) as compared to two labels for the other two datasets. It is also clear from this Figure that BitFit's

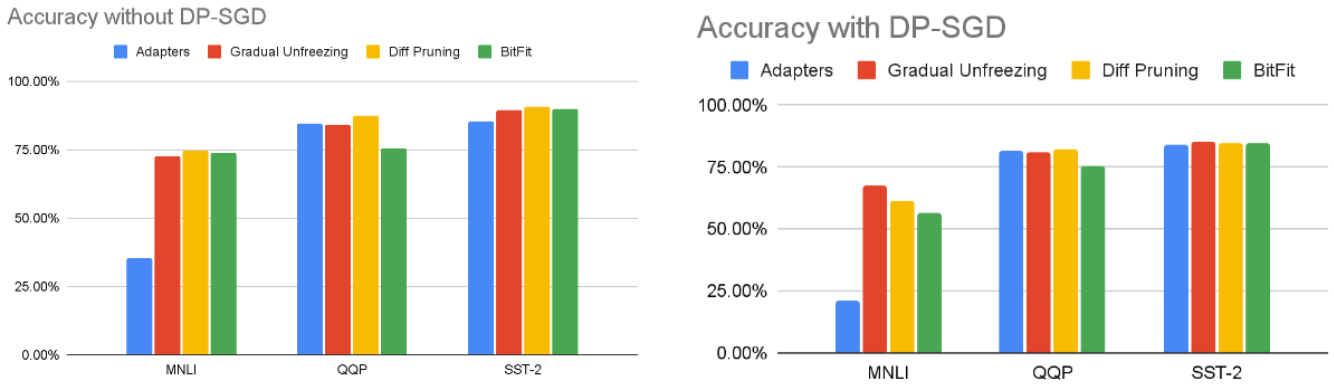

Figure 3: A side-by-side comparison of the accuracy of fine-tuning approaches across tasks with and without DP-SGD added. Accuracy for unfreezing was taken after the top two layers were unfrozen

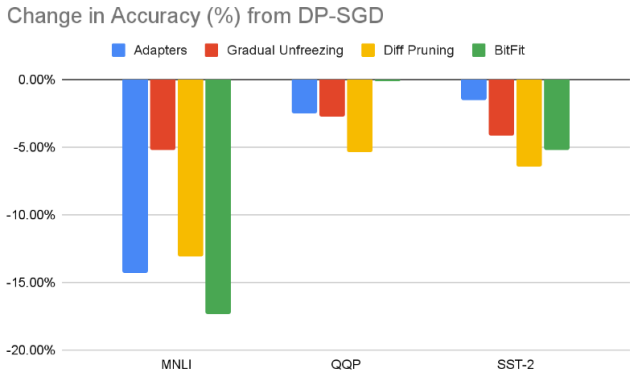

Figure 4: %Change in accuracy for the different GLUE tasks by fine-tuning method.

accuracy does not drop for the QQP task. This performance outlier and the reason for the larger drop in MNLI's performance both warrant further investigation.

Finally, we see that the performance drop for Gradual Unfreezing is the smallest (excluding outliers), which aligns with literature and is intuitive as the approach trains the most parameters during fine-tuning.

5.2 Training Time

In Fig. 5, we see a side-by-side comparison of the training time of the different fine-tuning techniques across tasks, both with and without DP-SGD. We can see that the training time follows a consistent pattern across the model types between tasks. MNLI and QQP both have significantly higher training times both with and without privacy guarantees. We hypothesize that the leading cause of this difference is due to the size of the datasets and seconarily due to the fact that the SST dataset consists of single sentences and a label, while the other two datasets contain two sets of sentences and a label.

It is intuitive that Gradual Unfreezing takes more training time than Diff Pruning and BitFit due to the larger number of parameters that are trained during fine-tuning. For this metric we obtianed the training time after the top two DistilBERT transformer layers were unfrozen as time increases with the number of unfrozen layers.

Fig. 6 shows the %change in training time. We see that the training time change is more impacted by the fine-tuning method than the task itself. We see general consistency of fine-tuning performance change across tasks though note that Diff Pruning is an outlier for SST-2.

6 DISCUSSION

6.1 Fine-tuning Technique Selection

The different parameter-efficient fine-tuning techniques have strengths weaknesses for implementation that are not immediately obvious from the results shown. These must be considered in addition to training time and accuracy for the most effective implementation.

Gradual Unfreezing, while providing the highest accuracy in the results section, also requires training of the most parameters and, therefore, the most computed. In applications that are limited in available computing, it would be appropriate to consider other techniques. The amount of computing associated with gradual unfreezing also increases as the number of layers being trained increases. This is relevant for larger models where more than two layers may need to be unfrozen to achieve the desired accuracy.

One benefit from adapters that warrants further exploration outside the scope of this paper is that since the original parameters are fixed it can be extended to new tasks easily without impacting (or forgetting) previous ones. This is especially useful when many different tasks may be selected by users and is a benefit that goes further than just the accuracy and training time associated with this fine-tuning approach. Adapter modules also contain a number of implementations and hyperparameters as discussed above that can modify implementations to best fit their application and available hardware.

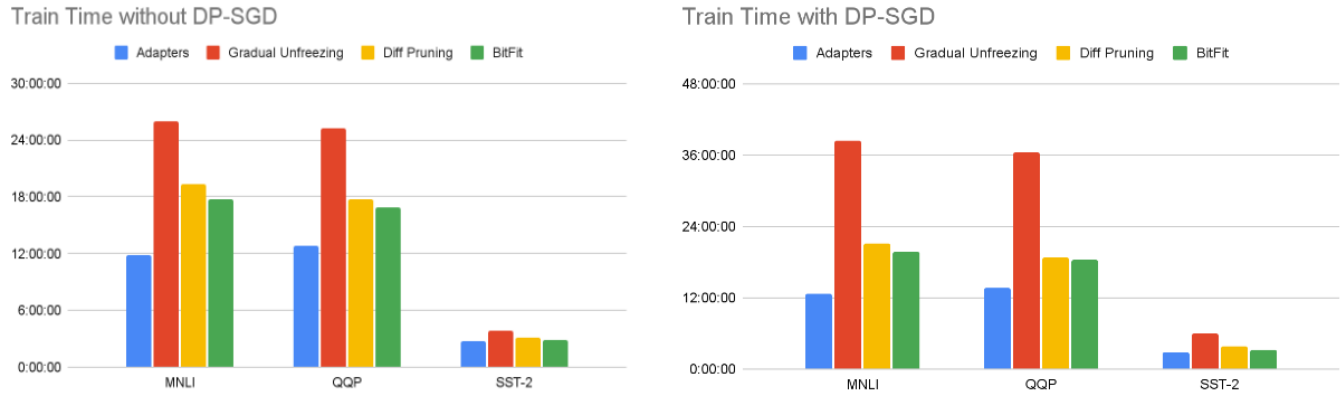

Figure 5: A side-by-side comparison of the per-epoch training time of fine-tuning approaches across tasks with and without DP-SGD added. Training time for unfreezing was taken after the top two layers were unfrozen.

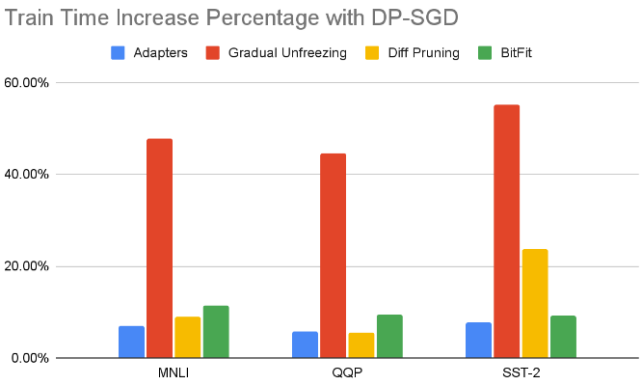

Figure 6: %Change in per-epoch training time. Note that this is more consistent across tasks than the absolute times due to the epoch size varying between GLUE tasks.

6.2 Challenges

As we explored the use of DP-SGD in conjunction with parameter-efficient fine-tuning techniques for language models, we faced several challenges and identify areas for improvement in the broader field of privacy-preserving NLP.

First, we found that although DP-SGD is gaining popularity, there is a scarcity of resources available for its implementation in LLMs. This creates a barrier for researchers and practitioners seeking to adopt privacy-preserving methods in their projects, limiting the widespread adoption of these techniques.

During our study, we encountered difficulties in locating comprehensive resources for implementing DP-SGD. Despite the existence of libraries designed to facilitate the use of DP-SGD, we found that their lack of active maintenance raises concerns about the potential accuracy loss associated with this technique. Additionally, the libraries discovered do not have flexibility in how they are implemented. This severely hinders their ability to be used with popular tools such as HuggingFace's library. These problems highlight the need for ongoing support and development of these tools to ensure their reliability and effectiveness in the context of language models.

Moreover, there is a growing demand for better modular DP-SGD libraries catering to those who are new to the subject. Providing accessible, well-documented, and easy-to-use libraries would significantly lower the entry barrier for researchers and practitioners interested in exploring privacy-preserving techniques in NLP. This, in turn, would promote the development and application of privacy-preserving methods, leading to more secure and privacy-aware systems.

In light of these challenges, we argue that there is a compelling need for more research and development efforts aimed at improving the availability, accessibility, and maintenance of resources for implementing DP-SGD in LLMs more inclusive and privacy-aware community in the field of NLP and facilitate the adoption of privacy-preserving methods across a wide range of applications.

7 FUTURE WORK

While our study has made progress in understanding the impact of parameter-efficient fine-tuning techniques with DP-SGD on model accuracy, there is still room for improvement. In future work, we suggest to explore novel methods and loss functions for fine-tuning to enhance accuracy while maintaining privacy guarantees.

One promising direction is the investigation of a recently published study by Lu et al. [23] that proposes a new loss function for fine-tuning aimed at improving accuracy in conjunction with DP-SGD. This research addresses the challenges faced by existing differential privacy approaches in deep learning, such as limited applicability of objective functions, unsatisfactory privacy-utility trade-offs due to excessive noise in each epoch, and uncertain utility improvement in output perturbation.

Lu et al. [23] proposes an output perturbation framework that injects DP noise into a randomly sampled neuron at the output layer of a baseline non-private neural network trained with a convexified loss function. This framework controls the overall noise injection and leverages the exponential mechanism for noise sampling, leading to a tighter upper bound on the global sensitivity of model parameters under a black-box setting.

Another aspect of future work involves addressing potential privacy leakage risks in deterministic parameter selection methods

like Diff-Pruning and BitFit. We plan to explore the use of the Sparse Vector Technique (SVT) [24], which can help mitigate such risks by adding a layer of privacy-preserving randomness to the process.

Future work should also involve the empirical evaluation of this framework on various real-world datasets, comparing the privacy-utility trade-off with existing DP-SGD implementations in terms of accuracy loss to baseline non-private models. These experiments should also explore the impact of varying the hyperparameters utilized as improvements may be obtained by customization between tasks and architectures.

By pursuing these avenues, we hope to develop innovative approaches that strike a balance between privacy and utility, significantly contributing to the advancement of privacy-preserving deep learning applications in the field of NLP.

8 CONCLUSION

The process of applying DP with DP-SGD to LLMs is not well understood yet. There aren't many resources to help people working in this field know how to balance privacy protection and model accuracy for different tasks. In our study, we share early findings on how different efficient techniques can be used to fine-tune LLMs while keeping data private. We looked into four techniques – Gradual Unfreezing, Adapter Modules, Diff Pruning, and BitFit – and how they affect model performance when used with DP-SGD.

Our results show that the effect of DP-SGD can vary a lot depending on the method used and the task at hand. This shows that we need more research to understand these complex interactions. We hope that by showing the possibilities of combining privacy and efficiency in LLMs, we can inspire more research in this important area. We aim to help build a better understanding of how to fine-tune LLMs privately and help create more efficient and privacy-protecting strategies for real-life uses.

ACKNOWLEDGMENTS

This work was performed for the course CSCI 8980: Advanced Topics in Databases - Data Systems for Dirty, Private, and Federated Data at the University of Minnesota. We would like to thank Professor Chang Ge, whose invaluable support and insightful guidance have been instrumental throughout the course of this project. We are also appreciative of the access to cloud services provided by The Minnesota Supercomputing Institute, which were indispensable in facilitating our computational experiments. Additionally, we would like to thank the many researchers cited below who chose to make their work open-sourced to enable rapid advancement of and transparency within the field of NLP and Differential Privacy.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/2976749.2978318

[2] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021. Large-Scale Differentially Private BERT. arXiv:2108.01624 [cs.LG]

[3] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, Scalable Adaptation for Neural Machine Translation. *CoRR* abs/1909.08478 (2019). arXiv:1909.08478 http://arxiv.org/abs/1909.08478

[4] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization, Revisited. *CoRR* abs/1405.7085 (2014). arXiv:1405.7085 http://arxiv.org/abs/1405.7085

[5] Samyadeep Basu, Daniela Massiceti, Shell Xu Hu, and Soheil Feizi. 2023. Strong Baselines for Parameter Efficient Few-Shot Fine-tuning. arXiv:2304.01917 [cs.CV]

[6] Mark Bun, Jonathan Ullman, and Salil Vadhan. 2018. Fingerprinting Codes and the Price of Approximate Differential Privacy. arXiv:1311.3158 [cs.CR]

[7] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. 2018. The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets. *CoRR* abs/1802.08232 (2018). arXiv:1802.08232 http://arxiv.org/abs/1802.08232

[8] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting Training Data from Large Language Models. *CoRR* abs/2012.07805 (2020). arXiv:2012.07805 https://arxiv.org/abs/2012.07805

[9] Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-Efficient Fine-Tuning Design Spaces. arXiv:2301.01821 [cs.CL]

[10] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. 2017. Modulating early visual processing by language. arXiv:1707.00683 [cs.CV]

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

[12] Friedrich Dörmann, Osvald Frisk, Lars Nørvang Andersen, and Christian Fischer Pedersen. 2021. Not All Noise is Accounted Equally: How Differentially Private Learning Benefits from Large Sampling Rates. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6. https://doi.org/10.1109/MLSP52302.2021.9596307

[13] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2022. On the Effectiveness of Parameter-Efficient Fine-Tuning. arXiv:2211.15583 [cs.CL]

[14] Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. arXiv:2012.07463 [cs.CL]

[15] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. arXiv:2110.04366 [cs.CL]

[16] Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021. Learning and Evaluating a Differentially Private Pre-trained Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 1178–1189. https://doi.org/10.18653/v1/2021.findings-emnlp.102

[17] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*.

[18] Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned Language Models for Text Classification. *CoRR* abs/1801.06146 (2018). arXiv:1801.06146 http://arxiv.org/abs/1801.06146

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

[20] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The Composition Theorem for Differential Privacy. arXiv:1311.0776 [cs.DS]

[21] Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, Pengcheng Wang, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, Ruofei Zhang, Winnie Wu, Ming Zhou, and Nan Duan. 2021. GLGE: A New General Language Generation Evaluation Benchmark. arXiv:2011.11928 [cs.CL]

[22] Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022. Generated Knowledge Prompting for Commonsense Reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 3154–3169. https://doi.org/10.18653/v1/2022.acl-long.225

[23] Zhigang Lu, Hassan Jameel Asghar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. 2022. A Differentially Private Framework for Deep Learning With Convexified Loss Functions. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2151–2165. https://doi.org/10.1109/tifs.2022.3169911

[24] Min Lyu, Dong Su, and Ninghui Li. 2016. Understanding the Sparse Vector Technique for Differential Privacy. *CoRR* abs/1603.01699 (2016). arXiv:1603.01699 http://arxiv.org/abs/1603.01699

[25] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. arXiv:2106.04647 [cs.CL]

[26] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJ0hF1Z0b

[27] Harsh Mehta, Abhradeep Thakurta, Alexey Kurakin, and Ashok Cutkosky. 2022. Large Scale Transfer Learning for Differentially Private Image Classification. arXiv:2205.02973 [cs.LG]

[28] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 46–54.

[29] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. arXiv:2005.00052 [cs.CL]

[30] Théo Ryffel, Francis Bach, and David Pointcheval. 2022. Differential Privacy Guarantees for Stochastic Gradient Langevin Dynamics. arXiv:2201.11980 [stat.ML]

[31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL]

[32] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. arXiv:2110.08207 [cs.LG]

[33] Murray Shanahan. 2023. Talking About Large Language Models. arXiv:2212.03551 [cs.CL]

[34] Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural Language Understanding with the Quora Question Pairs Dataset. arXiv:1907.01041 [cs.CL]

[35] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://aclanthology.org/D13-1170

[36] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. [n. d.]. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. 245–248. https://doi.org/10.1109/GlobalSIP.2013.6736861

[37] Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training Neural Networks with Fixed Sparse Masks. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=Uwh-v1HSw-x

[38] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. arXiv:1905.00537 [cs.CL]

[39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *CoRR* abs/1804.07461 (2018). arXiv:1804.07461 http://arxiv.org/abs/1804.07461

[40] Shu wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung yi Lee. 2021. SUPERB: Speech processing Universal PERformance Benchmark. arXiv:2105.01051 [cs.CL]

[41] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana). Association for Computational Linguistics, 1112–1122. http://aclweb.org/anthology/N18-1101

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? arXiv:1411.1792 [cs.LG]

[43] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. 2022. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR)*.

[44] 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. 2021. Differentially Private Fine-tuning of Language Models. *CoRR* abs/2110.06500 (2021). arXiv:2110.06500 https://arxiv.org/abs/2110.06500

[45] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. *CoRR* abs/2106.10199 (2021). arXiv:2106.10199 https://arxiv.org/abs/2106.10199