Low Rank Adaptation for Multitask BERT

Stanford CS224N Default Project

Johannes Fuest
Department of Statistics
Stanford University
jfuest@stanford.edu

Marco Tacke
Department of Engineering
Stanford University
mtacke@stanford.edu

Abstract

Fine-tuning large language models to simultaneously perform well on different downstream tasks poses an important challenge in natural language processing (NLP) [1], as saving different models for every downstream task is increasingly infeasible, given their ever-growing number of parameters [2]. In this paper, we test to what extent this issue can be resolved by augmenting models with Low Rank Adaptation (LoRA)[3]. This technique allows fine-tuning of large-language models (LLMs) on different NLP tasks at a fraction of the capacity requirements of three separate models [3]. We compare its performance to that of a basic version of the Bidirectional Encoder Representations from Transformers (BERT) model [4]. We define performance as average accuracy on sentiment analysis, paraphrase detection, and semantic similarity scoring. We find that LoRA exhibits strong performance in a multitask setting while also offering reduced computational complexity. Our findings are consistent with those of the original authors and further offer new insights into the suitability of LoRA in a multitask setting.

1 Key Information to include

- Mentor: Shai Limonchik

2 Introduction

Transformer models and the concept of self-attention [5] have raised the bar in all advanced NLP applications [6]. Models that have been trained on a large and universal corpus of text have proven to be especially useful for both generalist and domain-specific applications [7]. While effective, this pre-training is computationally expensive and has prompted the emergence of so-called large transformer language models with a high number of parameters, such as BERT [8] with 110 million trainable parameters and GPT-3 [9], which is currently the largest single transformer language model with 175 billion parameters [6].

In order to solve specific NLP tasks, these models require a domain- and task-specific second training phase after the universal pre-training. This so-called fine-tuning typically entails minor adjustments to all the learned parameters, which substantially improves the model performance for the specific task but makes it lose its universal functionality and applicability to different NLP tasks [10]. Combined with the storage requirements induced by a large number of model parameters, storing a fine-tuned large language model for each individual NLP task of an application on one device becomes infeasible [11]. This creates the need for a new, leaner approach to fine-tuning, which does not alter the foundational model but still enables the model to pick up on domain- and task-specific patterns and exhibit strong performance across the board.

Hu et al. [6] propose a solution to these conflicting goals that capitalizes on the structure of transformer layers. Transformer layers are composed of multiple matrices with full rank, denoted by W_q (queries), W_k (keys), W_v (values), and W_o (outputs). Aghajan et. al [12] argue that the effective changes in the matrices during the fine-tuning process have a lower *intrinsic dimension*. The LoRA methodology capitalizes on this, reducing the adaptation during fine-tuning to changes in low-rank decompositions of a matrix with the same dimensions as the frozen matrix from the transformer layer. With the original frozen matrix denoted as $W_o \in \mathbb{R}^{d\times k}$, the changes made during fine-tuning are denoted as $\Delta W \in \mathbb{R}^{d\times k}$. The latter is represented with a low-rank decomposition $\Delta W = BA$ with $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ where the parameter r is set to be substantially smaller than the dimensions d and k. During training, the W_o and ΔW are added coordinate-wise, but only A and B receive parameter updates. This overcomes the need to calculate and store gradients for the larger W_o matrix. Thus, instead of altering the foundational model parameters, these are frozen and not changed during fine-tuning. This approach greatly reduces the computational costs of fine-tuning, as the number of trainable parameters in the decomposition matrices is far smaller than the number of parameters in the model. Using this approach, the storage requirements of each fine-tuned NLP application, which is based on the same foundational model, are greatly reduced, as the foundational model must only be stored once alongside the smaller domain-specific decomposition matrices.

Known drawbacks and limitations. While the advantages of LoRA in terms of computational efficiency and storage requirements are obvious, there are also some known drawbacks. First, LoRA increases the complexity of the fine-tuning process and the pipeline. While regular fine-tuning only entails loading existing parameters and adjusting them with conventional training methods, LoRA introduces a more complex model architecture with matrices that do not follow the convention of large models, such as BERT, of using an identical number of hidden layers throughout the model [8]. The introduction of the new parameter r induces the need for additional hyperparameter tuning. While LoRA accelerates the training by 25%, it is not faster by a sufficient magnitude to compensate for the additional time commitment of hyperparameter tuning. Another drawback is that the approach relies on a well-balanced pre-training corpus. It will be able to amplify certain relations in the data, but it will not be able to introduce knowledge about an entirely new domain, as the original parameters are frozen and still dominate the added low-rank ΔW. A further limitation already mentioned in the original LoRA paper is that it is "not straightforward to batch inputs to different tasks with different A and B in a single forward pass if one chooses to absorb A and B into W to eliminate additional inference latency" [6].

Despite these disadvantages, we see the enormous potential of this technology in its ability to maintain the versatility of large language models while enabling time-efficient fine-tuning of multiple NLP tasks based on one model, which also leads to a substantial reduction in incremental storage requirements for each NLP tasks. In this project, we explore the tradeoff between this and the performance of LoRA, in order to holistically assess its suitability to multitask finetuning of the BERT model. We will do this by implementing LoRA in BERT models with varying degrees of sophistication and comparing its performance against traditionally finetuned BERT models. Identifying the conditions under which LoRA performs comparably well will allow for a more nuanced understanding of its applicability and contribution and could facilitate suitable deployment of lightweight multitask models in future use cases.

3 Related Work

There has been plenty of work studying challenges posed by multitask learning [13] and offering solutions [14]. LoRA is part of a body of work that focuses on the challenge of reducing the storage and computation required to train large language models for multitask learning. Other approaches in this area have used parameter subsetting and masking [15], parallelization techniques [16], pruning [17], as well as modifying the existing transformer architecture [18].

While LoRA has been applied in some studies in the NLP area to reduce computational requirements [19] and to exploit its interpretability for image data [20, 21], it has, to the best of our knowledge, not been applied in a multitask learning application.

A large part of this project focuses on modifying BERT and adding various improvements to it and comparing these improvements to LoRA. There have been many efforts that implement and extend such improvements of BERT. These have come in the areas like attention modification [22] and knowledge distillation [23] [24]. Our baselines are inspired by this type of research and include modifications of BERT that use cosine similarity loss.

Finally, our work draws on work from the general domain of multitask learning and multitask fine-tuning [25] in particular, as these ideas form the underpinnings of the baselines we compare LoRA to.

4 Approach

In order to conduct a holistic analysis of the tradeoff between performance, storage and computation offered by LoRA, our approach is to compare it to multiple baseline models, all of which are modifications of BERT. We finetune each baseline, as well as LoRA, and compare their storage requirements, training times, and accuracies. We describe each baseline below.

To test the performance of LoRA in a highly optimized multitask setting similar to a production environment, we apply a number of multitask modeling approaches to compare LoRA against (see Experimentation). This includes gradient surgery to avoid detrimental gradient interference by projecting "a task's gradient onto the normal plane of the gradient of any other task that has a conflicting gradient." [25]

For each of the model, add three separate heads for each subtask. Each head receives the last hidden BERT-layer of the `[CLS]`-token. For the tasks that require two inputs, we concatenate the individual BERT outputs of each input and feed the concatenation through the respective heads. We found that using two linear layers with Dropout and a ReLU activation function [26] substantially improves the performance compared to a single linear layer, while adding more layers did not yield substantially better performance.

Baselines. For our first baseline , we freeze the pre-trained BERT [4] and only train the task-specific heads. We use cross entropy as baseline loss function for the sentiment classification and for the paraphrase detection task. Cross entropy is defined as

$$L(y, \hat{y}) = -\sum_{j=1}^{C} y_j \log(\hat{y}_j) \tag{1}$$

where $\hat{y} \in \mathbb{R}^C$ are predicted probabilities and $y \in \mathbb{R}^C$ are the true label probabilities. For the ordinal similarity prediction task, we use mean squared error loss, defined as

$$L(y, \hat{y}) = (y - \hat{y})^2 \tag{2}$$

where $\hat{y}$ is the predicted value, and y is the true label. This baseline is at the lower end of the spectrum of the tradeoff we are examining, as the freezing of the BERT layers minimizes both the storage and computation requirements of training this model compared to the others in our analysis. Consequently, we expect this baseline to exhibit the worst performance out of our test models.

Our second baseline will be a fully fine-tuned BERT model that uses gradient surgery [25]. In this model, the BERT parameters are unfrozen during training. In order for the gradient surgery method to work, we stretch the datasets to be the same length artificially and then train on a batch of data for each task sequentially. Using the same loss functions as described above, we then perform gradient surgery, using an implementation by Wei Tseng [27]. This approach ensures that conflicting gradients in our loss functions do not get in the way of each other. The algorithm proceeds as follows:

```
Algorithm: Gradient Surgery
Require: Model Parameters $\theta$, $Minibatch\ \mathcal{B} = \{\mathcal{T}_k\}$

$g_k \leftarrow \nabla_\theta \mathcal{L}_k \theta \quad \forall k$
$g_k^{PC} \leftarrow g_k \quad \forall k$
for $\mathcal{T}_i \in \mathcal{B}$ do
    for $\mathcal{T}_j \in \mathcal{B} \setminus \mathcal{T}_i$ do
        if $g_i^{PC} \cdot g_j < 0$ then
            $g_i^{PC} \leftarrow g_i^{PC} - \frac{g_i^{PC} \cdot g_j}{||g_j||^2} g_j$
        end if
    end for
end for
return $\Delta\theta = g^{PC} = \sum_i g_i^{PC}$
```

Unlike the first baseline, training this model requires updating every single parameter of the original model, making it very computationally expensive. However, since the BERT layers are shared between all three tasks, this model is on par with the first baseline in terms of storage requirements. This baseline is expected to perform significantly better than the first baseline.

Next, we want to test how sophisticated improvements to a standard BERT model translate to LoRA. Thus, our third baseline will be identical to the second baseline with one extension. For semantic textual similarity scoring, we will use cosine similarity loss, defined as

$$L(x_1,\ x_2) = \ (cos(x_1, x_2) - y)^2 \tag{3}$$

where x_1 and x_2 are the model's output embeddings and y is the label divided by five and $cos(x_1,\ x_2)$ measures the cosine similarity between the embeddings. This loss function may be able to improve performance on semantic textual similarity, as it forces the embeddings of a model into a shape where similar inputs for a given task yield a higher cosine similarity and vice versa. It is unclear if this performacnce gain also translates to LoRA.

The fourth and final baseline we test LoRA against will be a version of BERT with separate, unfrozen parameters for each task. This version requires triple the computation and triple the storage of the second baseline. Barring any benefits from parameter sharing however, we would expect it to exhibit the best performance of the four baselines.

LoRA Implementations. In order to test LoRA in combination with BERT, we implemented a `LoraLinear` layer from scratch. This module amends the logic of a regular `Linear` layer by functionalities that store A and B matrices (see Introduction) as parameters and automatically handle the (de-)composition, prediction, and training of the combined layer. Unlike the LoRA implementation described by the authors of the original LoRA paper, this module natively supports multiple sets of A and B matrices specifically for multitask applications when specifying a list of subtasks during initialization and a specific subtask during evaluation. We provide functionalities that specifically make the gradients of the A and B parameters compatible with PyTorch's autograd functionality while freezing all other BERT parameters and the option to deactivate the LoRA functionalities, allowing for traditional fine-tuning for testing purposes. However, to explore whether sharing the A/B-matrices can also have a beneficial effect, we also implemented a version of LoRA that uses gradient surgery to only train one low rank decomposition$\Delta W = BA$ per W_0. Varying between these two options, as well as using cosine and MSE loss for the sts task, we thus arrive at four distinct versions of LoRA we test. By varying between the baseline loss and cosine loss as well as parameter sharing of the A and B matrices, we end up with four LoRA configurations to configure (see Table 1).

Research approach. Having built the baseline models, we can now proceed to comprehensively compare each of them to LoRA by taking not only the model performance, but also resulting storage requirements and training durations into account. The different characteristics of the baseline models in this approach will allow us to present a holistic analysis of the usability of LoRA for multitask finetuning. We provide an overview of the models we use in table 1 below.

5 Experiments

Datasets. For our experiments we use three datasets, each corresponding to one downstream NLP task we test on. For sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset [28], for paraphrase detection, we use the Quora dataset [29], and for semantic textual similarity scoring, we use the SemEval STS Benchmark dataset [30].

Evaluation Method. Our models will be simultaneously evaluated on their performance on three NLP tasks. For sentiment analysis and paraphrase detection, we will use accuracy of our predictions,

Model Name	**BERT Layers**	**Learning Rate**	**LoRA**	**STS LOSS**	**Shared BERT Layers**
Baseline 1:	Frozen	$1e^{-3}$	✗	MSE	✗
Baseline 2:	Unfrozen	$1e^{-5}$	✗	MSE	✓
Baseline 3:	Unfrozen	$1e^{-5}$	✗	Cosine	✓
Baseline 4	Unfrozen	$1e^{-5}$	✗	Cosine	✗
LoRA 1	Frozen	$1e^{-5}$	✓	MSE	✗
LoRA 2	Frozen	$1e^{-5}$	✓	Cosine	✗
LoRA 3	Frozen	$1e^{-5}$	✓	MSE	✓
LoRA 4	Frozen	$1e^{-5}$	✓	Cosine	✓

Table 1: Experiment Results

defined by

$$\frac{\#\, Correct\ Predictions}{\#\, Total\ Predictions} \tag{4}$$

For semantic textual similarity scoring, we will use pearson correlation of predictions and results as an evaluation metric, defined by:

$$corr(X,Y) = \frac{Cov(X,Y)}{sd_X * sd_Y} \tag{5}$$

where X are predctions and Y are labels. We will further evaluate our models on their average performance across the three tasks, which will be a simple average of the three tasks' respective evaluation metrics.

Experimental Details. For each of the described baselines, as well as the LoRA implementation, we run the same experiment. When sharing parameters, we stretch the sst and the sts datasets until they are the same size as the Quora dataset. We do this, because an equal number of batches for each task is required for gradient surgery to function. We then train on the datasets using a batch size of 8 before testing the performance of our models on an unseen test set for each task. For baseline 1, we found a learning rate of $1e^{-3}$ to be most suitable, all other experiments were run on a learning rate of $1e^{-5}$. Throughout each experiment, we log training speeds and accuracies over time, as well as the size of the final model. The results of this are summarized in table 1.

Results. Table 2 reports the performances of our LoRA implementations, as well as the baselines we tested against. The worst overall performance was exhibited by baseline 1, in which only the heads of our model were finetuned to the three NLP tasks.

Experiment	**Average Performance**	**SST Accuracy**	**Paraphrase Accuracy**	**STS Correlation**
Baseline 1:	43.53%	38.42%	67.27%	24.89%
Baseline 2:	55.13%	**50.77%**	78.94%	35.69%
Baseline 3:	**63.32%**	49.23%	**79.31%**	58.41%
Baseline 4	62.42%	49.96%	78.46%	58.85%
LoRA 1	57.95%	49.51%	76.73%	47.61%
LoRA 2	62.23%	48.68%	76.95%	**61.61%**
LoRA 3	57.93%	47.13%	69.10%	57.55%
LoRA 4	61.52%	48.77%	69.22%	66.58%

Table 2: Performance Results

Experiment	Incremental Storage Requirements (3 Tasks)	Bundled Storage Requirements
Baseline 1:	1.78M	111.78M
Baseline 2:	111.78M	111.78M
Baseline 3:	111.78M	111.78M
Baseline 4	331.78M	331.78M
LoRA 1	5.10M	115.10M
LoRA 2	5.10M	115.10M
LoRA 3	2.89M	115.10M
LoRA 4	2.89M	112.89M

Table 3: Computational Results

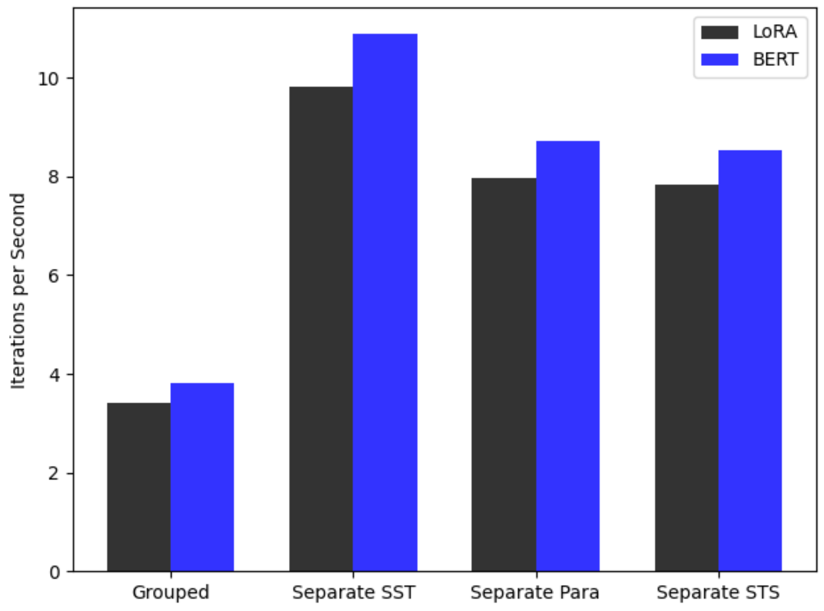

Figure 1: Training times

6 Analysis

Overall Performance Overall, our empirical performance testing confirmed our initial hypotheses. Our performance results are displayed in Figure 2. First, baseline 1 exhibited the worst performance, which is expected, as only the heads were trained on the specific tasks, with BERT layers frozen. This basic finding reiterates the need for fine-tuning in the first place. In general terms, we find that LoRA exhibits comparable performance to the related BERT baselines. This overall performance finding is in line with that of the original authors.

Choice of Loss Functions. As expected, using cosine similarity loss was able to improve performance on semantic textual similarity, as it forces the embeddings of a model into a shape where similar inputs for a given task yield a higher cosine similarity and vice versa. We were able to show this improvement for the regular unfrozen BERT as well as LoRA, indicating that LoRA is generally applicable to both commonly used and more sophisticated objective functions.

Storage Requirements. In order to fit multiple NLP applications that are based on the same large language model on one device, the store necessary for each application is crucial. We measure the

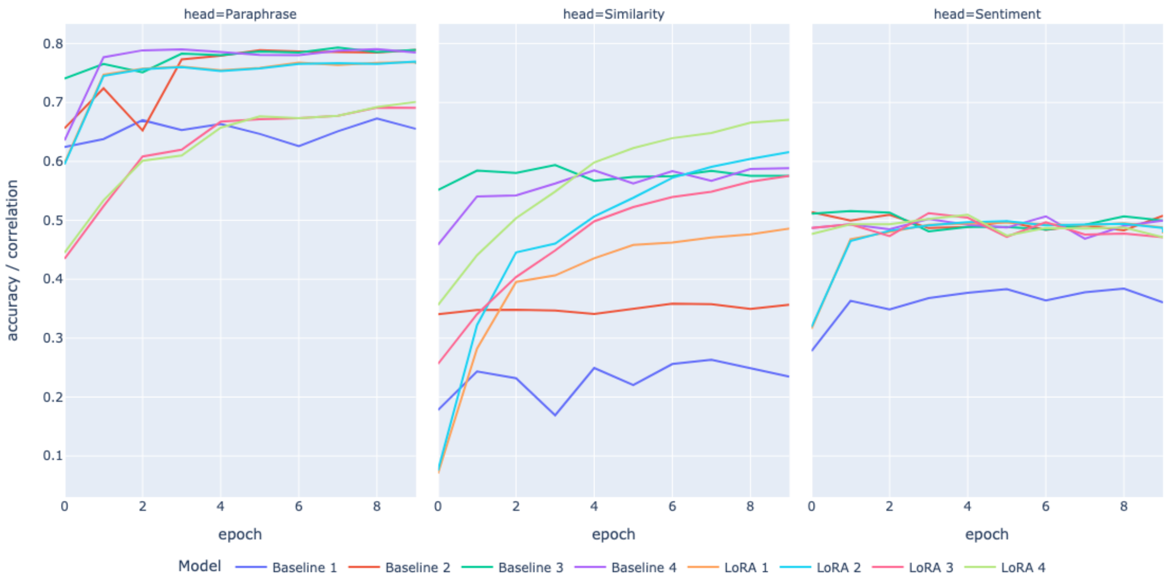

Figure 2: Performance graphs

storage requirements in number of parameters and differentiate between two scenarios: A scenario in which a copy of the BERT model is already present on the device for other NLP tasks and only the incremental storage requirements of the three additional NLP tasks at hand is of concern, and a scenario in which we need to store a solitary application that is independent of other NLP applications already on the device. The first column is particularly relevant for a world with a small number of large language models that serve as the basis for a large number of task-specific NLP applications. Only adding heads to a frozen BERT requires the lowest number of parameters (1.78M) to store but also clearly shows the lowest performance across all tasks. Training separate BERT models shows the highest overall performance but also requires the storage of about 332 million parameters. The best LoRA configuration achieves 98% of the performance improvement of the best-performing baseline compared to the worst-performing baseline while only requiring an incremental storage of 4.5% of the respective additional parameters.

Furthermore, it is worth noting that the usage of independent LoRA parameters for each NLP task allows workflows in which specific "plug-and-play" applications can be added, updated, or amended independent of each other. As an example, updating model parameters of the similarity model on a new and improved Quora dataset would be possible without the need to retrain all three specific models, which would not be possible in the case of gradient surgery, rendering the usage of LoRA more risk-averse and future-proof for real world NLP applications.

Code Complexity and Parameter Tuning. When analyzing the codebase that was built up for this project, the majority of complexity for the trainer of multitask models evolves around manually weighting loss functions or even implementing gradient surgery as an integrated approach for this task. A joint training furthermore required us to make compromises in terms of learning-rate, independent early-stopping, and number of epochs. While we generally kept these parameters constant for our final runs, we saw the potential of decoupling the optimization processes, as the training processes of separate LoRA parameters are effectively independent of each other. Furthermore, we found that the smaller dimension of the A/B matrices has a substantial impact on the performance, introducing the need for an additional hyperparameter to be considered for training.

Parameter Sharing. Both for the unfrozen BERT as well as LoRA, we can see a distinct difference between sharing parameters across all tasks and handling those separately. While shared parameters seem to have an advantageous effect on the similarity and paraphrase tasks, which are structurally very similar, the conceptually different sentiment tasks suffers slightly from sharing parameters with the other two tasks. Using LoRA generally allows the creation of specific shared tasks groups. Here, this could mean sharing the A/B-matrices for the paraphrase and similarity tasks while allowing the sentiment task to train an independent set of parameters. While similar approaches are generally thinkable without the usage of LoRA, the incremental storage requirements of each additional task with a separate parameter space of around 110M is unfeasible for many applications.

Training duration. Our training speed results are displayed in Figure 1. All measurements here were made on the same virtual machine. While the authors of the original paper found training speed to be increased by 25%, we were only able to achieve a training-speed increase of roughly 10%. All measurements were undertaken on AWS's EC 2 instance of type "Deep Learning AMI GPU PyTorch 1.13.1 (Ubuntu 20.04)" with an NVIDIA A10G GPU. This difference could be due to inefficiencies in the overhead of our training loop and is thus not an indication of a substantial difference to the original authors' findings. As noted by the authors, this training speed-up aspect of LoRA is far less significant than its reduced storage requirements.

7 Conclusion

This project successfully implemented variations of BERT that simultaneously performed different NLP tasks and compared them to similar implemenatations that used our custom version of LoRA. We find that LoRA offers a significant advantage with regards to storage requirements and training time. This confirms the findings of the original authors. Our analysis further establishes that LoRA is highly suitable for multitask settings, exhibiting comparable performance to our baselines across all tasks.

Future Work There are various directions that further examinations of LoRA in a multitask learning context could take. First, experiments in settings with higher number of distrinct NLP tasks may accentuate the differences in training time and storage between LoRA and conventional fine-tuning. Experimenting on a more varied selection of datasets would also cement the generality of our findings. One potential interesting source for such datasets is offered by ChatGPT, which is able to synthetically create datasets. We include an example of this in the appendix. Furthermore, all of the models in our analysis could be further improved in their performance by adding more enhancements, such as regularization, hyperparameter tuning, and techniques such as warm-up. While applying this to all models in our experiments is out of scope for this project, we found initial promising results by applying warm-up to our shared parameter LoRA implementation and increasing the batch size to 24, resulting in test leaderboard scores of 52.35%, 72.74%, and 74.45% on sst, sts, and para respectively.

References

[1] Zhihan Zhang, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. *arXiv preprint arXiv:2204.03508*, 2022.

[2] Canwen Xu and Julian McAuley. A survey on dynamic neural networks for natural language processing. *arXiv preprint arXiv:2202.07101*, 2022.

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[7] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[12] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[13] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-task learning in natural language processing: An overview. *arXiv preprint arXiv:2109.09138*, 2021.

[14] Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. Massive choice, ample tasks (machamp): A toolkit for multi-task learning in nlp. *arXiv preprint arXiv:2005.14672*, 2020.

[15] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. *CoRR*, abs/2109.05687, 2021.

[16] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019.

[17] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *CoRR*, abs/1910.04732, 2019.

[18] Jonathan Pilault, Amine Elhattami, and Christopher J. Pal. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. *CoRR*, abs/2009.09139, 2020.

[19] Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, et al. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. *arXiv preprint arXiv:2207.08988*, 2022.

[20] Yang Chen, Yudu Li, and Zongben Xu. Improved low-rank filtering of mr spectroscopic imaging data with pre-learnt subspace and spatial constraints. *IEEE Transactions on Biomedical Engineering*, 67(8):2381–2388, 2019.

[21] Loreen Ruhm, Johanna Dorst, NI Avdievich, Andrew Wright, Christian Mirkes, Jonas Bause, Klaus Scheffler, and Anke Henning. 31p mrsi of the human brain at 9.4 t: Metabolic imaging applying low-rank denoising. In *27th Annual Meeting and Exhibition of the International Society for Magnetic Resonance in Medicine (ISMRM 2019)*, 2019.

[22] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR, 2019.

[23] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[24] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351, 2019.

[25] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[26] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[27] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[28] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[29] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs. In *https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs*, 2016.

[30] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

8 Appedix:

The below data is a short excerpt of data returned by ChatGPT4 when prompted with "Generate me a list of 500 pairs of sentences and add a score from 0 to 5 on a continuous scale of how similar they are".:

No.	Sentence 1	Sentence 2	Score
1	The sun is shining brightly today.	It's a sunny day outside.	4.5
2	She baked a delicious apple pie.	He made a tasty banana bread.	3.5
3	The car engine needs repair.	The motorcycle engine is malfunctioning.	4.7
4	The ocean waves crashed against the shore.	The desert sands shifted in the wind.	1.8
5	The night sky was filled with stars.	The dark room made it hard to see.	1.7
6	I need to buy some dog food.	She's purchasing cat food for her pet.	4.4
7	Chocolate chip cookies are my favorite.	I love oatmeal raisin cookies.	4.0
8	He played table tennis at the community center.	She enjoyed a game of ping pong with her friends.	4.6
9	He lost his wallet at the amusement park.	She misplaced her purse at the theme park.	4.0
10	She planted roses in her garden.	He cultivated tulips in his backyard.	3.7
11	The baby giggled when tickled.	The infant laughed when being tickled.	4.8
12	He wore a blue shirt and black pants.	She donned a red dress and white shoes.	2.5
13	I'm going for a run in the park.	I'll be jogging at the park today.	4.3
14	She studied hard for the math exam.	He prepared diligently for the history test.	3.6
15	The movie was fantastic, and I enjoyed it.	I loved the film; it was amazing.	4.5
16	The author wrote a compelling novel.	The writer penned an engaging book.	4.7
17	I can't find my keys, and I'm late.	I misplaced my wallet, and I need to hurry.	3.8
18	The city was bustling with activity.	The urban area was alive with action.	4.6
19	Her new hairstyle looked stunning.	His fresh haircut was quite stylish.	3.4
20	I'm learning to play the guitar.	I've started taking piano lessons.	2.7
21	The rain poured down all day.	It was a nonstop downpour.	4.4
22	The children played in the park.	The kids had fun at the playground.	4.2
23	She picked up a gallon of milk from the store.	He grabbed a loaf of bread at the supermarket.	3.6

No.	Sentence 1	Sentence 2	Score
24	The concert was so much fun.	I had a blast at the live performance.	4.3
25	They went on a hike in the mountains.	The group trekked through the forest.	3.2

Using such data to train