Multitask minBERT with Parameter-efficient Fine-tuning

Stanford CS224N Default Project

Zhuoqi (Charlie) Zhang
Department of Statistics
Stanford University
czzq@stanford.edu

Abstract

The project explores the techniques and effects of fine-tuning pretrained BERT to be a multi-class model for a variety of downstream tasks. In particular, it highlights how the careful transformation of embeddings, the structure of the fine-tuning loop, and the choice of loss functions can result in sizable performance gain across tasks without increasing the overall model complexity. Furthermore, it also studies the parameter-efficient fine-tuning methods in this context, showcasing highly desirable trade-off ratios between trainable parameters and performance.

1 Key Information

Mentor: Soumya Chatterjee, **External Collaborators**: None, **Sharing project**: No

2 Introduction

A pretrained Bidirectional Encoder Representations from Transformers (BERT) model offers versatile embeddings for various downstream tasks. In practice, BERT initially undergoes extensive pretraining with massive datasets on generic tasks, like masked token prediction, and then fine-tunes on more specific tasks with their respective smaller datasets.

This pattern of fine-tuning pretrained models has seen wide application, proving effective in performance and more computationally accessible than training models from scratch Qin et al. (2023). However, several challenges arise in the implementation details. First of all, there are numerous decisions and configuration possibilities to go from embeddings to the final task-specific output, such as the further processing of embeddings and the formulation of loss functions based on the characteristics of a given task. Additionally, in a multi-task setting, we may require adjustments to the fine-tuning process so that the embeddings generated are simultaneously generalizable to all downstream tasks of interest. In other words, it may be desirable that the model cycles through batches across multiple task-specific datasets evenly and frequently. Finally, full fine-tuning still involves updating a large number of trainable parameters, so we investigate the effect of parameter-efficient fine-tuning (PEFT) methods that may reduce trainable parameters to a fraction of the original quantity while maintaining minimal performance trade-offs Houlsby et al. (2019).

The remainder of the report delves into the methodologies and experiments aimed at addressing the aforementioned challenges. Under the default final project framework, the multi-task model for this report builds upon the minBERT model with pretrained weights. The enhancements implemented to mitigate these challenges are evaluated across three downstream tasks: sentiment prediction, paraphrase detection, and semantic textual similarity (STS). At the end of the project, the Best Performance model achieved an overall score of 0.778 on the test set. With PEFT, the model was able to reduce its trainable parameters to around 20% while only losing roughly 5% of overall score on the development set.

3 Related Work

The original BERT paper outlines the fundamental information such as model architecture, pretraining, and token configuration Devlin et al. (2019). More importantly, its experiment section highlights the results of many extension possibilities, such as ensembling, additional pretraining, and various feature-based approach to leverage the pretrained embeddings.

However, the topic of multi-task learning was not in the spotlight until around the BERT and PALs paper Stickland and Murray (2019). The Projected Attention Layers (PALs) mechanism explores the option of adding task-specific parameters *alongside*, as opposed to on top of, BERT parameters. In other words, the main BERT parameters are shared across tasks, while extra supplementary parts of the model are dedicated to picking up task-specific signals. The paper also raises the issues with fine-tuning on task-specific datasets in the naive sequential manner and proposes an alternative, sampling-based, approach. Additionally, the MTRec paper, in the scenario of tackling multiple news article related tasks, presented more sophisticated decoding configurations, the idea of "main" and "auxiliary" tasks, and the practice of combining losses from multiple tasks while accounting for potential conflicts in gradient updates Bi et al. (2022).

One detail that is particularly relevant to this project is the input format. For example, to accommodate paired sentence inputs, the original BERT paper proposes the concatenation of the pair into one Devlin et al. (2019). On the other hand, the Sentence-BERT paper recognized that BERT is more suitable for token-level tasks and subsequently adapted it into a Siamese Network architecture, where two sub-networks of BERT share weights and process sentence pairs Reimers and Gurevych (2019). In the process, the paper leverages the fact that paired sentence inputs are usually for tasks that measure similarity in some capacity. As a result, it devises a loss function that penalizes the pair of embeddings based on their level of similarity and the label.

Finally, computational efficiency has been a coveted property and much of it can depend on the concept of parameter efficiency. As an alternative to full fine-tuning, which involves updating all model parameters, parameter-efficient fine-tuning (PEFT) techniques in general drastically lower the number of trainable parameters required with a minor tradeoff in performance. For example, Low-rank Adaptation (LoRA) approximates the weight updates that happen during fine-tuning with low-rank matrix multiplications Hu et al. (2021). Weight-decomposed Low-rank Adaptation (DoRA) builds upon the previous approximation idea, but it first decomposes the pretrained weights into magnitude and direction components in order to induce some of the desirable properties that full fine-tuning exhibits in its weight updates Liu et al. (2024).

4 Approach

The high-level structure of the approaches of this project involves first exploring extensions that can provide the most amount of *performance* improvement to the baseline model. Subsequently, it applies PEFT methods to the most performant model obtained and examines the levels of *efficiency* gain.

4.1 Baseline

The baseline model aims to produce multi-task outputs with minimal extensions. For the sentiment classification logits, the baseline uses $W_s(\text{Dropout}(u)) + b_s$, where u is the [CLS] token embedding and W_s are the trainable weights in the last linear layer for sentiment classification. For the paraphrase and STS logits, the baseline simply outputs the cosine similarity between the pair of [CLS] token embeddings. Lastly, the baseline fine-tunes only on the default sentiment classification dataset.

4.2 Feature Representations

Given a single sentence, the project considers using the default [CLS] token embedding as a "summary" representation of the whole sentence versus applying pooling operations (including *mean* and *max* pooling) on the sequence of wordpiece-level embeddings generated by the last BERT layer, referred to as the "last hidden state (lhs)".

Given a pair of sentences, we extend the ideas above. Let u and v be the representations of their respective sentence, via [CLS] token embedding or pooled embedding, we experiment with the

concatenated subsets of $A = \{u, v, |u - v|, |u + v|\}$ as the input to the final prediction module Reimers and Gurevych (2019).

Given a pair of sentences, we also adopt the suggestion to pass them *simultaneously* through BERT, as $(\text{Sentence1} + [\text{SEP}] + \text{Sentence2})$, sometimes referred to as cross-encoding, where [SEP] is the separator token Devlin et al. (2019). Let s be the cross-encoded embedding of the sentence pair, we add that to the mix of feature set A and the input to the final prediction module could be, for example, $\text{Concat}(u_{\text{lhs}}, v_{\text{lhs}}, s_{\text{lhs}})$.

4.3 Multitask Training Loop

In each epoch, suppose we have three task-specific datasets D_1, D_2, D_3 for fine-tuning, the naive way would be to go through them sequentially as $D_1 \rightarrow D_2 \rightarrow D_3$, exhausting all batches in one dataset before starting on the next. This approach brings up the potential drawbacks of learning on one task for an extended period of time.

Alternatively, we cycle round-robin at the batch level $b_{1i} \rightarrow b_{2j} \rightarrow b_{3k} \rightarrow ...$, such that we train on each task at an even and constant pace, restarting on smaller datasets until the largest dataset is depleted. However, this over-sampling nature could lead to over-fitting to smaller datasets if the size imbalance is huge.

Finally, we examine the sampling-based option. Whenever the model is ready to process the next batch, we sample from the collection of all batches across all tasks with the probability $p_i \propto |D_i|$. We also examine the effect of task data imbalance in this configuration. If we use probability distribution $p_i \propto \sqrt{|D_i|}$, we effectively skew the sampling probabilities toward the smaller datasets Stickland and Murray (2019). Let the epoch conclude whenever a dataset is depleted and this is equivalent to under-sampling the majority dataset.

4.4 STS with Cosine Embedding Loss

The default approach to STS is to treat it as a regression task, get feedback from the mean-squared-error loss, and clamp the final output to the predefined range based on the dataset. However, the nature of the STS task allows the idea of penalizing based on similarity measures between a pair of sentences. One such loss we examine in this project is the Cosine Embedding Loss. With u, v as embeddings and y as the label for the pair, it states

$$\text{Loss}(u, v) = \begin{cases} 1 - \cos(u, v), & \text{if } y = 1 \\ \max(0, \cos(u, v) - \text{margin}), & \text{if } y = -1 \end{cases} \tag{1}$$

In other words, for paired sentences that are labeled as similar, we get a penalty for how dissimilar their embeddings are Reimers and Gurevych (2019). For pairs that are labeled as dissimilar, we get a loss for how similar their embeddings are such that weak similarity below a certain margin does not incur any penalty. In practice, if the original label space is continuous, we experiment with the ways to map them to $\{-1, 1\}$ in order to apply this loss function.

4.5 DoRA

Until this point, combinations of previous approaches experimented under full fine-tuning should have yielded a multi-task model with solid performance. We then apply DoRA to the model with the highest overall score to analyze the effects of this PEFT method Liu et al. (2024). At a high level, this means altering all linear layers in the existing model.

Specifically, let $W_0 \in \mathbb{R}^{d \times k}$ be the trainable weights in the original linear layer. DoRA, leveraging the weight decomposition method, sets the magnitude component $m = ||W_0||_c$ and the directional component $V = W_0$ Salimans and Kingma (2016). Let ΔV be the weight updates that would happen to the directional component under full fine-tuning, by the Low-rank Approximation (LoRA) method, we may approximate $\Delta V = BA$ where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $r \ll \min(d, k)$. In summary, the new weights W' under DoRA would look like

$$W' = m \frac{W_0 + BA}{||W_o + BA||_c} \tag{2}$$

In other words, DoRA decomposes the original weights into magnitude and normalized directional components. During fine-tuning, the magnitude component is updated plainly while the directional

component is updated through LoRA. Parameter-efficiency is achieved by having only m, B, A as the trainable parameters. In experiment, we also observe the effects of the hyperparameter rank r.

5 Experiments

5.1 Data

This project uses some of the datasets recommended by the default project handout. Specifically:

- The Stanford Sentiment Treebank (SST) dataset[1] is the task-specific dataset for sentiment classification. The training set includes 8544 examples. Each input is a movie review and its corresponding label denotes one of the five discrete sentiment categories.
- The Quora dataset[2] is the task-specific dataset for paraphrase detection. The training set includes 283003 examples. Each input is a pair of questions posted on the Quora website and the pair has a corresponding binary label indicating whether the pair is considered duplicate questions or not.
- The SemEval STS Benchmark dataset is the task-specific dataset for semantic textual similarity evaluation Cer et al. (2017). The training set includes 6040 examples. Each input is a pair of sentences and the pair has a continuous similarity score in the interval [0,5] where higher scores indicate stronger similarity.

5.2 Evaluation method

The evaluation method follows the default project leaderboard metrics. In particular, we evaluate both the sentiment classification and the paraphrase detection tasks by accuracy. We evaluate the STS task by the Pearson correlation between the true similarities and the predicted similarities. The metrics for the individual tasks are shown in table columns with shorthand *Sentiment, Paraphrase*, and *STS*. The overall score (shown in table columns with shorthand *Overall*) is calculated as $\frac{1}{3}(\mathrm{Acc}_{\mathrm{sentiment}} + \mathrm{Acc}_{\mathrm{paraphrase}} + (0.5 + \mathrm{Corr}_{\mathrm{sts}}))$. For PEFT experiments, the number of trainable parameters is measured as the proxy for parameter-efficiency.

5.3 Experimental details

The experiments have the same pretrained weights for BERT, the same learning rate of $1e^{-5}$, and the number of epochs fixed to 10. No early stopping or increase in the number of epochs is in place because the model's performance on the development set tends to converge fairly quickly at around epoch 4 to 6. The experimental results that are focused on a single aspect of model configuration have all other parameters held constant.

5.4 Results

5.4.1 Key Results

First, in Table1, we showcase the evaluation metrics of model configurations that considerably improved upon its previous iterations, in chronological order:

- *Key Update 1.* Compared to the baseline, this version includes all three datasets mentioned above. The STS task uses the Cosine Embedding Loss.
- *Key Update 2.* Compared to Key Update 1, this version modifies the fine-tuning loop to use the sampling-based approach to select a batch at each step.
- *Key Update 3.* Compared to Key Update 2, this versions swaps out the use cases of the [CLS] token embedding with mean-pooled token embeddings.
- *Best Performance.* Compared to Key Update 3, sentence pairs are cross-encoded, that is, passed through the encoder together as opposed to separately. Accordingly, STS switches to become a regression task penalized by mean-squared-error loss.

[1] https://nlp.stanford.edu/sentiment/

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model	Overall	Sentiment	Paraphrase	STS
Baseline	0.494	0.471	0.390	0.271
Key Update 1	0.670	0.457	0.778	0.550
Key Update 2	0.681	0.509	0.784	0.519
Key Update 3	0.717	0.500	0.872	0.559
Best Performance	**0.779**	**0.524**	**0.874**	**0.877**
Most Efficient	0.757	0.501	0.846	0.850
Test Leaderboard	**0.778**	**0.524**	**0.875**	**0.871**

Table 1: Performance of notable model configurations. Metrics other than "Test Leaderboard" are based on the development set.

- *Most Efficient.* we swap out each linear layer in the Best Performance model with its DoRA formulation (rank 16) to drastically reduce the number of trainable parameters.

At the end of Table1 is the test leaderboard results of the "Best Performance" model, surpassing the initial expectations. This outcome aligns with one of the sub-themes of the project, which aims to limit the overall model complexity. From a quantitative perspective and considering notable model configurations, we observe the following: (1) Fine-tuning on an ample amount of task-specific data is essential for the pretrained embedding to generalize for a given task; (2) The STS task benefits most significantly from cross-encoding compared to bi-encoding; (3) Careful adjustments to non-architectural components, such as loss functions, embedding choices, and fine-tuning procedures, yield incremental yet consistent improvements. The key results collectively highlight the significance of methodological nuances in optimizing model performance.

5.4.2 Feature-based Experiment Results

Concatenation	STS
(u,v)	0.530
(u,v,|u-v|)	0.559
(|u+v|,|u-v|)	0.553
(u,v,c)	**0.877**

Table 2: Embedding concatenations.

Sampling Method	Probabilities	Overall
Sequential	-	0.767
Proportional	[0.03, 0.95, 0.02]	0.773
Square Root	**[0.13, 0.76, 0.11]**	**0.779**
Balanced	[0.33, 0.33, 0.33]	0.748

Table 3: Overall score versus sampling methods.

By directly comparing the different ways to concatenate relevant embeddings, we get the quantitative results in Table2. The concatenated tensor is the input to the final linear layer, which generates the logits. The u and v are bi-encoded, mean-pooled, last hidden states. It is noteworthy that incorporating information about the sum or difference between the pair of embeddings yields a minor boost in STS correlation, which agrees with the suggestions in the Sentence-BERT paper Reimers and Gurevych (2019). Moreover, having the cross-encoded embedding c in the concatenation is able to increase the correlation by over 0.3, marking the largest gain for the STS task among all strategies.

5.4.3 Training Loop Experiment Results

In Table3, the Probabilities column shows the sampling probabilities for the SST, Quora, and the SemEval STS datasets. Sampling batch-by-batch from all three task-specific datasets (Proportional) allows the model to frequently change the task that is is fine-tuning on. As discussed in the dataset section, the Quora dataset is considerably larger than the other two. Alternatively, when the sampling probabilities are proportional to the square-rooted sizes of the datasets (Square Root), it not only limits the data size disparity to some extent, but also avoids removing too many examples from the majority dataset. This contrasts with the method of forcing an even split among datasets (Balanced), where we witness that the benefit of having access to more data significantly outweighs the appeal of fine-tuning on evenly-sized datasets.

6 Analysis

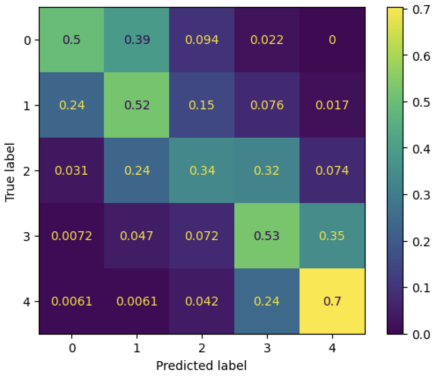

Figure 1: Sentiment confusion matrix.

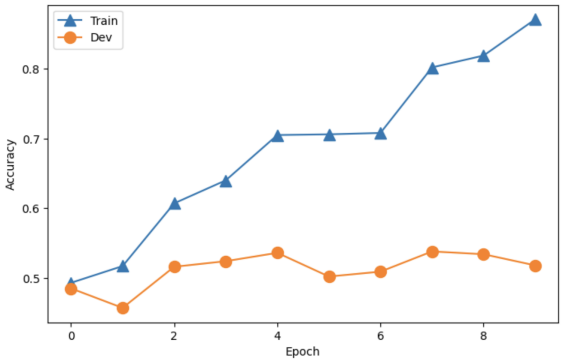

Figure 2: SST accuracy over epochs.

6.1 Task Performance Analysis

Sentiment Classification. This task among the three is the hardest one for the model to improve its performance on. We commonly observe that the development set accuracy converges very early on across all model configurations experimented with in this project. For example, the training curve of the Best Performance model shown in Figure2 exhibits that the model is able to learn peculiarities that exist in the training data but do not generalize well to unseen data. To uncover how and where the model makes mistakes, we inspect the confusion matrix in Figure 1. The raw counts are normalized on the True Label level. For example, True Label 2 (Neutral) has the weakest performance. Among those sentences, 24% were predicted as 1 (Somewhat Negative) and 32% were predicted as 3 (Somewhat Positive). This pattern extends to other labels, where the predictions concentrate *near* the true label, which corresponds to how the heatmap highlights areas *near* the diagonal. Admittedly, the finer we divide the sentiment space, more nuanced examples arise to which even human annotators would not assign labels unanimously.

Mistake	Question 1	Question 2
FP	Why computer vision is hard?	Why computer vision is *computationally* hard?
FN	How is Donald Trump in person?	What is Donald Trump like in person, *away from the press/media*?
FP	How do I fix a laptop that won't turn on?	How do you fix an *HP* laptop that won't turn on?

Table 4: Paraphrase detection error examples.

Paraphrase Detection. Based on the development set for this task, the number of mistakes for each label is proportional to the number of total examples with that label, indicating a balanced performance. We select a few noteworthy examples from the mistakes, shown in Table4, to illustrate some of the subtle differences that are particularly challenging. In the Mistakes column, "FP" refers to false positives, meaning predicting a non-duplicate pair as duplicates, and "FN" refers to false negatives accordingly. As we can see in the paired questions, though the quantifiers may only consist of a single word, *computational* difficulty is much more specific than general difficulties (example 1), much like how fixing an *HP* laptop might require specific expertise compared to fixing any laptops (example 3). On the other hand, longer quantifiers may not change the scope of the question in the case where being *away from the press* is synonymous to being in person (example 2).

Semantic Textual Similarity. Since the Best Performance model treats the task as a regression problem, we can look at the residual plot to discover error patterns. Shown in Figure4, the first thing to notice is

that despite the predicted values are almost always well-contained in the possible range of similarities, simply setting an upper-bound of 5 for output values can provide a minor performance increase. Secondly, we observe the emptiness in the lower-left quadrant, implying the model's relatively higher tendency to over-predict for pairs with low similarities.

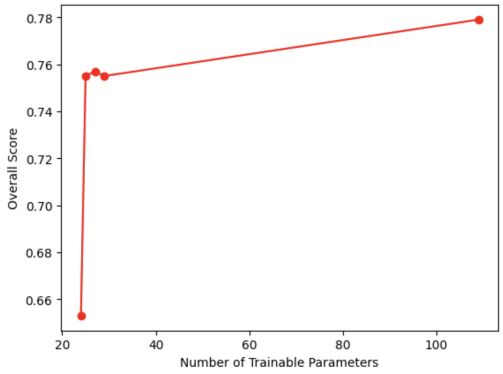

Figure 3: Model size versus performance.

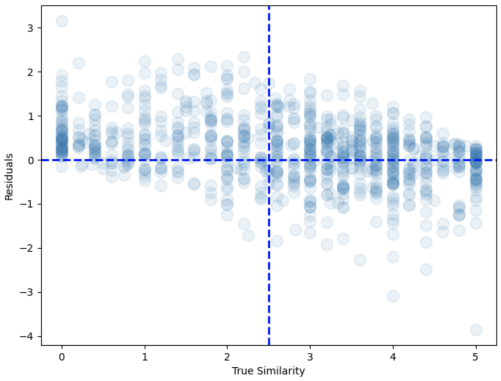

Figure 4: Residual plot of STS predictions.

6.2 PEFT Analysis

In Figure3, we compare the number of trainable parameters with the resulting overall score for models integrating DoRA with rank $r = [4, 8, 16, 32]$, as well as the full fine-tuning model in the upper-right corner. As mentioned earlier, the rank parameter directly controls how "low-rank" we want matrices B and A to be for approximation. A higher rank would imply more parameters to update.

The first observation is that despite $r = 4$ and $r = 8$ only have a roughly 1 million difference in terms of trainable parameters, the gap in their overall scores is significant. Conversely, once the rank exceeds $r = 8$, the performance gains are negligible.

Compared to the full fine-tuning result, DoRA's parameter efficiency is evident. The Best Performance model with full fine-tuning requires a roughly 300% increase in the trainable parameters while only yielding a 3% increase in the overall score. Importantly, within the scope of this project, integrating DoRA does not compromise some desired system properties. For instance, models with DoRA still converge quickly, requiring a similar number of epochs as full fine-tuning models. Additionally, in this current multi-task setting, the performance trade-offs occur evenly across tasks, indicating that none of the current downstream tasks are particularly susceptible to approximation errors in the weight matrices resulting from DoRA.

7 Conclusion

This project demonstrates the collective impact of enhancing small components within a multi-task model. While the availability of task-specific datasets is critical to the performance, refining the training loop, identifying a more suitable loss function, and exploring potential feature representations can be effective and resource-efficient strategies. These enhancements, spread across the deep learning system, can lead to significant improvements, even under constraints of model complexity.

Furthermore, this project showcases how cross-encoding is essential to the learning of the STS task, resulting in a final Best Performance model that achieves an overall test score of 0.778 on the leaderboard. More importantly, applying DoRA reveals that competitive model performance can be attained with only a fraction of the original trainable parameters. This demonstrates that PEFT is desirable when computational resources are limited or when faster iterations through modeling ideas are needed.

Regarding the limitations, it is important to note that the set of downstream tasks are relatively small. For the multi-task model to handle a more complex, diverse, or niche set of tasks, broader experimentation with the model's architecture may be required. Additionally, since DoRA is only

one of the many PEFT methods, a comprehensive comparison of multiple other approaches within this multi-task setting is a reasonable next step.

8 Ethics Statement

One of the ethical concerns in the context of multi-task learning is the interactions across different task-specific datasets. When weights are shared, the transfer of biases across tasks is not yet fully understood. In real-world application scenarios where the number of tasks increases significantly, managing the relationships among datasets and maintaining the integrity of each dataset becomes challenging. Consequently, the undesirable properties of one may extend to impact the whole multi-task system. This concept may extend beyond biases to privacy. That is, even when there are multiple datasets in play, the privacy constraints on each single dataset cannot be relaxed in the hope that being in a multi-task system can obscure where the signals originate. A possible mitigation strategy is to deconstruct complex multi-task objectives into smaller sub-systems that interact with only a few datasets. This allows for a bottom-up approach to combine smaller systems into a more complex form.

The other ethical challenge has to do with PEFT. Parameter efficiency can lead to significant monetary savings due to reduced computation needs. For companies providing deep learning as a service to various customers, switching to PEFT methods can significantly cut operational costs without noticeably affecting the evaluation metrics visible to customers. This scenario may lead companies to withhold benefits from customers by not adjusting prices in accordance with the reduced costs. As PEFT methods evolve, the signals indicating such changes can become even less discernible. Therefore, it is advisable to implement policy updates that regulate appropriate disclosure.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Mtrec: Multi-task learning over bert for news recommendation. In *Findings*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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.

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.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver?

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Tim Salimans and Diederik P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.