Leveraging PEFT Strategies for Improved Performance and Efficiency in minBERT

Stanford CS224N Default Project

Mateo Quiros Bloch
Department of Symbolic Systems
Stanford University
`mquiros@stanford.edu`

Lara Seyahi
Department of Computer Science
Stanford University
`seyahi@stanford.edu`

Susan Ahmed
Department of Computer Science
Stanford University
`suahmed@stanford.edu`

Abstract

In the realm of Language Models, the computational and storage demands of large pre-trained models often hinder their practical application, particularly when only specific downstream tasks are of interest. Parameter-efficient fine-tuning (PEFT) methods, such as those proposed by the LoRA and prefix tuning frameworks, present a solution to this challenge by enabling effective model fine-tuning with a minimal increase in parameter count. This study evaluates the PEFT method for fine-tuning the minBERT model on multi-task learning, targeting sentiment analysis, paraphrase detection, and semantic textual similarity. Using datasets such as Stanford Sentiment Treebank, CFIMDB, Quora, and SemEval STS Benchmark, we aim to assess LoRA and prefix tuning's efficiency. We hypothesize that PEFTs, particularly with LoRA, will maintain or enhance minBERT's performance while significantly reducing parameter count. We also compare LoRA variants, including hyperparameter tuning, and its application to different layers, to determine their impact on performance and tradeoffs. The results show that the model with our final LoRA version, applied to the self-attention layers, resulted in a decreased accuracy. The results also show that with prefix-tuning, the accuracy decreased.

1 Key Information to include

- Late Days: Our group is sharing late days (2 from Lara and 1 from Mateo) for an extra late day.
  TA Mentor: Aditya Agrawal
  Team Contributions: Mateo adapted the code base to enable LoRA fine-tuning and its different variations. Susan worked on adapting the code to enable prefix tuning. Lara focused on research of methodologies and analysis of results. We all participated in the implementation of the minBERT, the write-ups and poster.

2 Introduction

Natural Language Processing (NLP) has achieved significant advancements with the advent of pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers). These models have set new benchmarks in various NLP tasks, such as text classification, named entity recognition, and question answering. However, fine-tuning such large models for specific tasks can be resource-intensive and time-consuming, often requiring substantial computational power and large amounts of labeled data. To address these challenges, researchers have proposed Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA) and other techniques. These methods aim to reduce the number of parameters that need to be updated during fine-tuning, thereby decreasing the computational load and resource requirements. Moreover, PEFT methods allow effective model fine-tuning, often achieving similar or better performance than full fine-tuning [1].

Resource Efficiency:
One of the most significant advantages of using PEFT methods like LoRA is the substantial reduction in computational resources required for fine-tuning. Traditional fine-tuning involves updating a large number of parameters in models like BERT, which necessitates considerable GPU power and memory. PEFT methods, by updating only a fraction of the parameters or using low-rank adaptations, minimize this requirement, making it feasible to fine-tune large models on more modest hardware (Xu et, al 2023).

Speed and Time Efficiency:
By reducing the number of parameters that need to be adjusted, PEFT methods accelerate the fine-tuning process. Fine-tuning all the pre-trained parameters can become time-consuming (Xu et, al 2023). This leads to faster convergence times, enabling quicker iterations and experimentation.

In our approach, we use PEFT methods, LoRA and prefix tuning, to fine-tune our minimalist version of the BERT model (minBERT) efficiently. Firstly, we implemented the minBERT model and obtained baseline accuracy metrics for the sentiment classification task on the SST and CFIMDB datasets. Secondly, we implemented baseline PEFT techniques, namely, LoRA and prefix tuning. We researched how these PEFT methods work through reading the literature that explains their mechanisms. In this first iteration of the experiment, a LoRA class was implemented to the self-attention layer (query and value matrices), and then to the feed-forward layers of the transformer. Then, we implemented prefix tuning that was added to the embedding layers. In the end, we evaluated the performance of these methods across the different variations on these downstream tasks over the baseline performances of our PLM (Pre-trained Language Model).

RESULTS:

3 Related Work

BERT[1] revolutionized NLP by introducing bidirectional training of Transformer models. Unlike previous models that processed text in a left-to-right or right-to-left manner, BERT reads the text in both directions through "jointly conditioning on both left and right context in all layers", allowing the model to understand the context of a word based on all surrounding words in a sentence. This bidirectional approach enables BERT to achieve superior performance on various NLP tasks. We worked on a smaller version of BERT, minBERT, to be able to train and make inferences in an accessible way.

In the systematic review of PEFT methods for PLMs titled "Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment", various PEFT methods that can be used on pretrained language models are explained. The review catogizes PEFT methods into the following cateogies: additive fine tuning, unified fine tuning, reparameterized fine-tuning, hybrid fine-tuning, and partial fine-tuning. Amongst the reparameterized Fine-tuning of the PEFT methods, we focused on LoRA, listed under low-rank decomposition.

LoRA [2] was proposed by researchers at Microsoft through the paper "LoRA: Low-Rank Adaptation of Large Language Models". The paper discusses the method of freezing "the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture". The insight is that certain tasks have intrinsic "low dimensionality" and thus we only need low rank matrices to capture task specific features. The paper notes that this approach greatly reduces "the number of trainable parameters for downstream tasks". The paper discusses the method of adapting large pre-trained language models by learning low-rank updates to their weights, significantly reducing the number of trainable parameters required for fine-tuning. We used this paper as a baseline in forming our understanding of how LoRA works and the best practices to implement it. For example, we followed the recommendations of the ideal rank parameter to use and to which transformer layers LoRA should be applied to.

Amongst the additive fine-tuning of the PEFT methods, we focused on prefix-tuning, listed under soft prompt-based fine-tuning. Reading the systemic review, we got inspired by prefix-tuning [3], which proposes to prepend soft prompts to the "hidden states of the multi-head attention layer, differing from prompt-tuning which adds soft prompts to the input". Prefix tuning was developed by researchers from the University of Washington and Microsoft Research, introduced in the paper titled "Prefix-Tuning: Optimizing Continuous Prompts for Generation". The paper defines prefix tuning as "a lightweight alternative to fine-tuning for natural language generation tasks, which keeps language model parameters frozen, but optimizes a small continuous task-specific vector"[3]. This paper was helpful for us in understanding how prefix-tuning works and the best practices for its implementation.

Our work is a promising analysis and exploration of these two PEFT methods applied to minBERT for downstream tasks such as sentiment classifications, paraphrase detection and semantic text similarity. The papers discussed above compare the BLEU scores of both methods to fine-tuning and found that both lead to similar or better performance than full fine-tuning. We want to replicate their results on a smaller model as well as validate whether the best practices described also lead to better performance on minBERT.

4 Approach

The first project implementation is the multi-head self-attention and transformer layers of minBERT, a simplified BERT model. The coding of this model is done on top of the minBERT- Default-Final-Project Repository, integrating them into the baseline minBERT model provided. Finally, we ensured they function as intended through testing and sanity checks.
After implementing a minimalist version of BERT, we implemented LoRA and prefix tuning as described below.

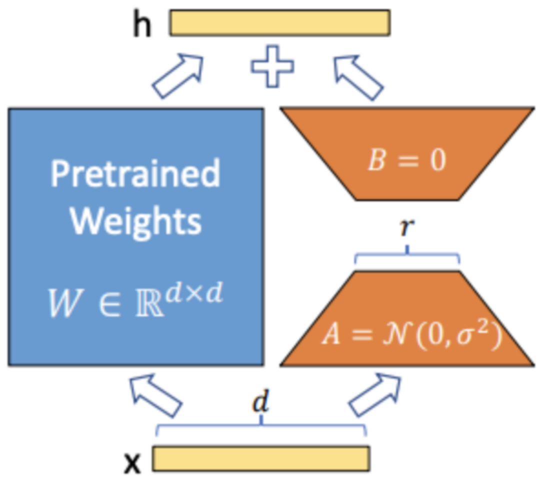

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 1: $\Delta W = BA$[2]

LoRA:
LoRA uses two trainable low-rank matrices for weight update. In LoRA, a down-projection matrix and an up- projection matrix are utilized in parallel with the query (Q) and value (V) matrices in the attention layer of the transformer as well as the feed-forward layers. These matrices represent the full-rank pre-trained weight matrix.

For a pretrained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, the low rank decomposition can be represented as:
$W_0 + \Delta W = W_0 + BA, \quad \text{where} \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times k}$ and ΔW is the adaptation matrix, with $r \ll \min(d, k)$ indicating the rank and BA being a low-rank product.
This decomposition reduces the number of trainable parameters even though the total number of parameters in the model increases, as A and B have smaller dimensions than the original weight matrix which is freezed during training.

To implement this into our minBERT model, we first initialized the necessary LoRA parameters, the low rank matrices. The up-projection matrix B is initilized with zeros and the down projection matrix A with a uniform distribution. Then when calculating the projection, we integrated the LoRA low rank matrices into this calculation, when the linear layer is applied to the input tensor x, it projects it into a new space. This is then multiplied by the low rank LoRA matrices, and added to the original projection (Figure 1).

During training, only the LoRA parameters A and B (the low-rank matrices of LoRA), and layer norm, biases and task-specific heads are fine-tuned, and all other pre-trained parameters are frozen. This allows for more efficiency in tasks by reducing the number of parameters to train (Figure 2). Selective training allows the model to focus on the task at hand without compromising the model's overall capabilities. It also allows us to train and use different LoRA instances by appending them to the original matrices at inference time depending on the task at hand.

Prefix Tuning:
In prefix tuning, we add a set of trainable parameters that are meant to increase the performance of specific tasks, without having to retrain the entire model. In our implementation, we prefixed the input with a set of trainable embeddings. In the model's embedding layer, we included a set of embeddings that are learned during training. When training this data using the classifier and multitask classifier, we froze all parameters excluding any prefix parameters, layer norms, biases and task-specific model heads. This gave the model less parameters to train in order to enhance performance on different tasks. We configured the prefix length in the initialization of the BERT model and experimented with different prefix lengths depending on the task we wanted to improve. For more complex tasks, we would experiment with longer prefix lengths, and shorter prefix lengths for simpler tasks. We found the most optimal prefix length to be 10. In the embed function, the aforementioned prefix embeddings were added to the standard embeddings of the input tokens, this contained the original information and the additional information from the trainable prefix parameters.

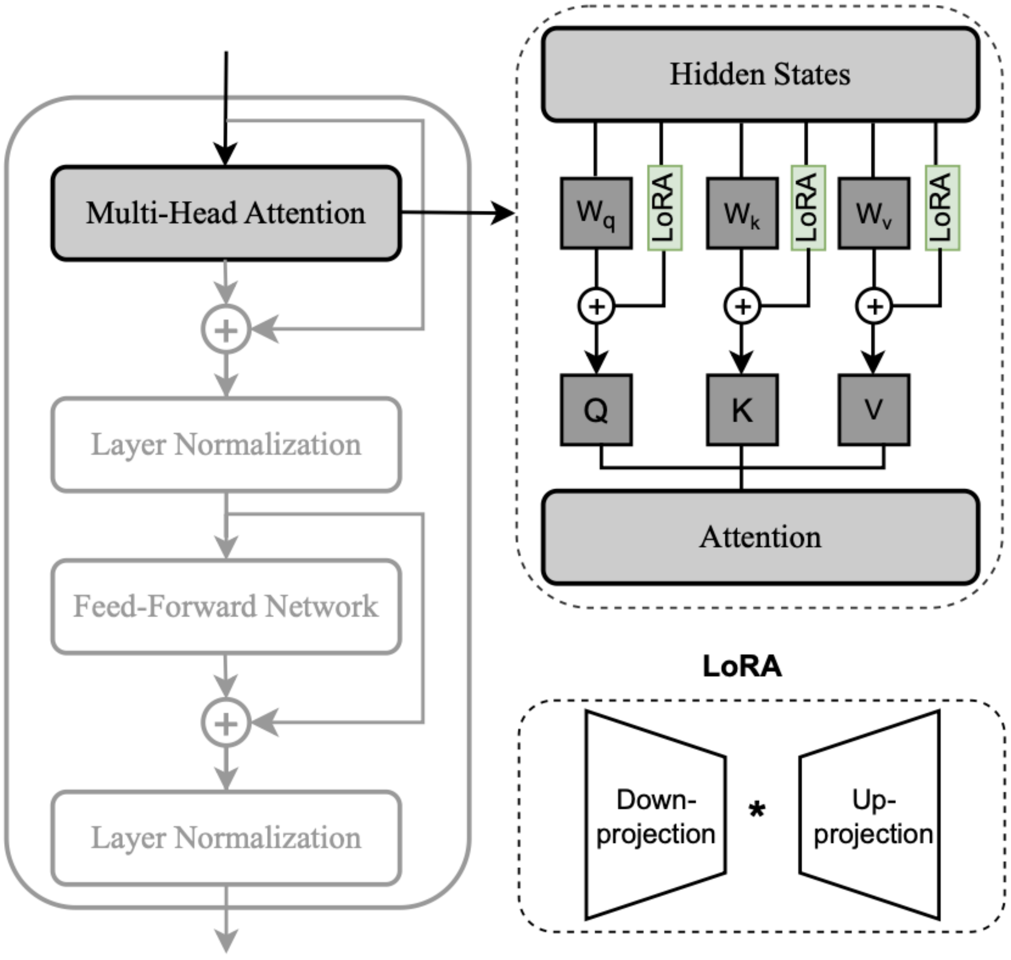


Figure 2: LoRA [1] (LoRA not applied to key matrix in our case)

5 Experiments

5.1 Data

1. **Stanford Sentiment Treebank (SST) [4]**: Consists of 11,855 sentences from movie reviews, parsed and annotated into 215,154 unique phrases. The dataset is divided into training (8,544 examples), development (1,101 examples), and testing (2,210 examples) splits. Each phrase is labeled as negative, somewhat negative, neutral, somewhat positive, or positive.
2. **CFIMDB dataset**: Includes 2,434 polar movie reviews with a binary sentiment label (negative or positive). The splits are training (1,701 examples), development (245 examples), and testing (488 examples).
3. **Quora Dataset [5]**: Comprises 404,298 question pairs labeled to indicate whether they are paraphrases of one another. The data splits include training (283,010 examples), development (40,429 examples), and testing (80,859 examples).
4. **SemEval STS Benchmark Dataset [6]**: Contains 8,628 sentence pairs with similarities rated on a scale from 0 (unrelated) to 5 (equivalent meaning). This dataset is divided into training (6,040 examples), development (863 examples), and testing (1,725 examples).

5.2 Evaluation method

Our study employs well-defined, quantitative metrics to evaluate the performance of the PEFT-enhanced minBERT model across different tasks.

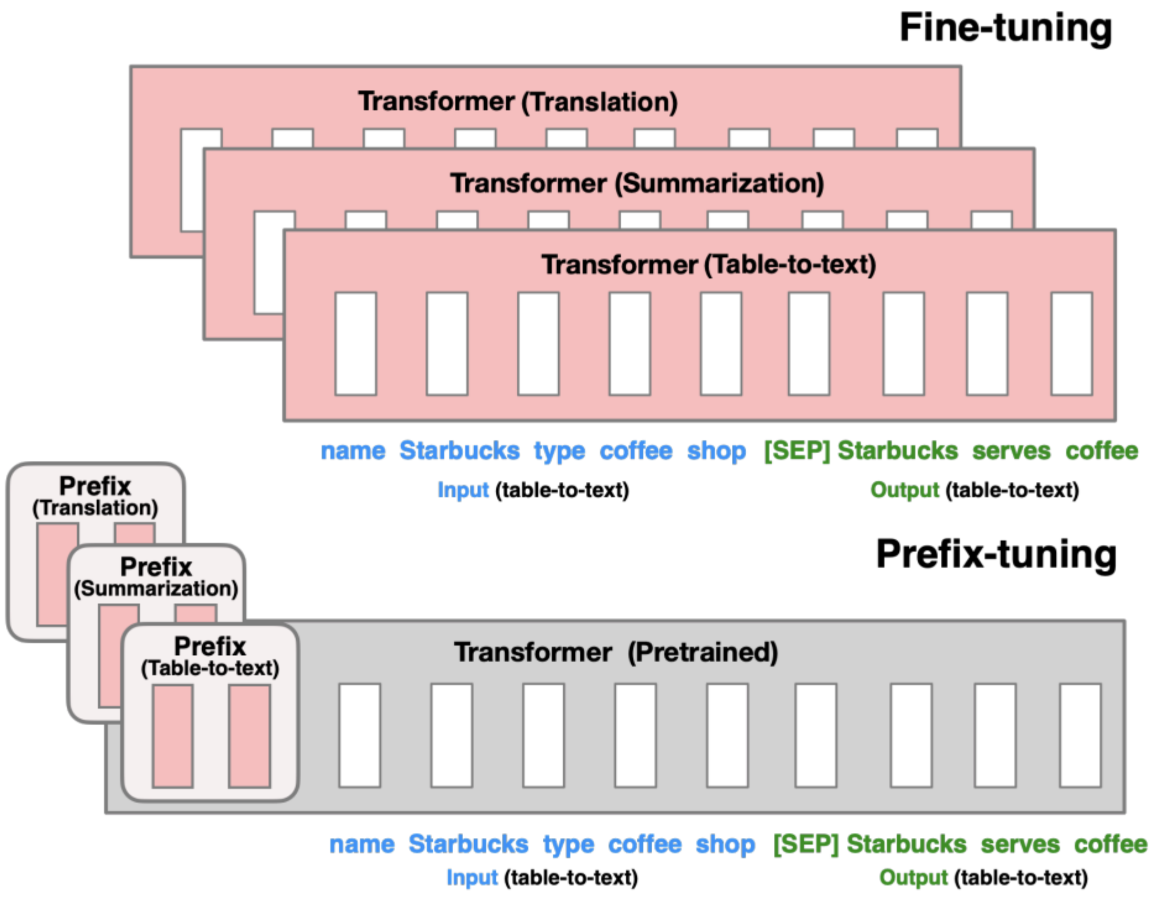

Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix-tuning
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)

Figure 3: Prefix Tuning [3]

Accuracy: For sentiment analysis on the Stanford Sentiment Treebank (SST) and the CFIMDB dataset, as well as for paraphrase detection on the Quora dataset, accuracy will serve as the primary metric.
Trainable Parameter Reduction: The Trainable Parameter Reduction is the percentage decrease of parameters that are trained (not frozen) in the model during fine-tuning compared to full fine-tuning or to other fine-tuning methods. This metric will be used for comparisons across all experiments.
Pearson Correlation: For the task of semantic textual similarity, as assessed using the SemEval STS Benchmark dataset, we will utilize the Pearson correlation between the true similarity values and the predicted similarity scores.
Loss: For sentiment classification and paraphrase detection, we used cross-entropy loss to measure the difference between the predicted probabilities and the actual class labels. For semantic textual similarity, we employed mean squared error (MSE) loss to quantify the difference between the predicted and true similarity scores.
GPU usage: We tracked GPU usage during training thanks to the pynvml library. We report average and maximum GPU usage in Megabytes during training.

5.3 Experimental details

Report how you ran your experiments (e.g., model configurations, learning rate, training time, etc.)

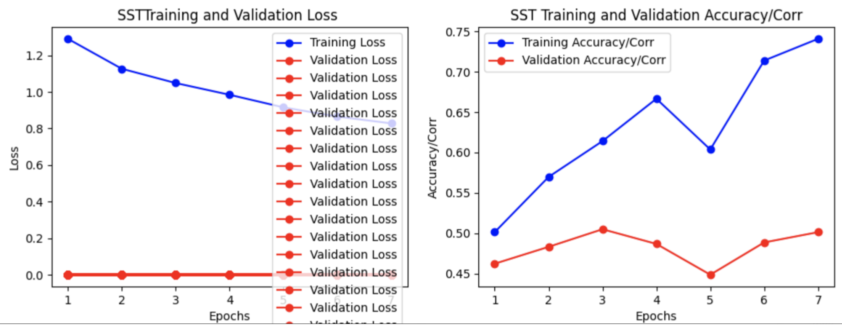
In our experiments with Low-Rank Adaptation (LoRA), we tested various configurations to evaluate their performance on the task. Specifically, we utilized a learning rate of $1e-4$ and experimented with different ranks of 4, 8, and 16 for the low-rank matrices. LoRA was applied both exclusively to the self-attention layers and to a combination of self-attention and feed-forward network (FFN) layers within the model. Each model was trained for a maximum of 10 epochs. However, we kept the checkpoint weights at 7 epochs due to signs of overfitting or a reduction in training accuracy observed beyond the fifth epoch. We also incorporated an alpha scaling factor of 32 to appropriately scale the low-rank updates. The experimental setup allowed us to identify the optimal configuration while preventing overfitting, ensuring robust evaluation of LoRA's impact on model performance.

For our experiments with prefix tuning, we focused on exploring the impact of different prefix lengths on model performance. We tested a range of prefix lengths to determine the most effective configuration. The models were trained with a consistent learning rate of 1e-4, ensuring a controlled comparison of prefix length effects. Each experiment was conducted over 10 epochs to observe the training dynamics and performance across a sufficient number of iterations. The training was consistently stopped at X epochs to maintain uniformity in the evaluation process.

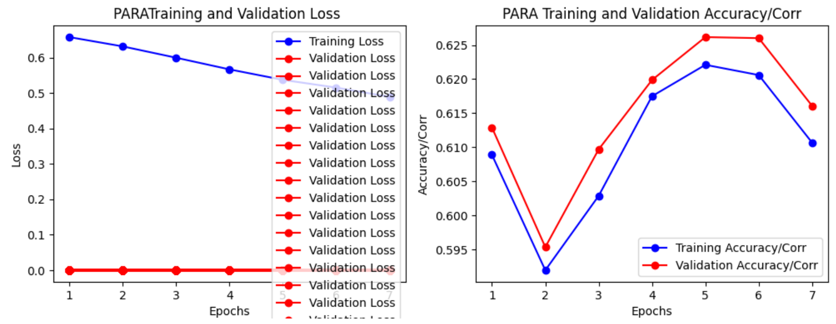
5.4 Results

- **Pre-Trained MinBERT Baseline Results:**
  - **SST Dataset:**
    - Full model Dev Accuracy: 0.516
    - Last Linear Layer Dev Accuracy: 0.409
  - **CFIMDB Dataset:**
    - Full model Dev Accuracy: 0.967
    - Last Linear Layer Dev Accuracy: 0.787

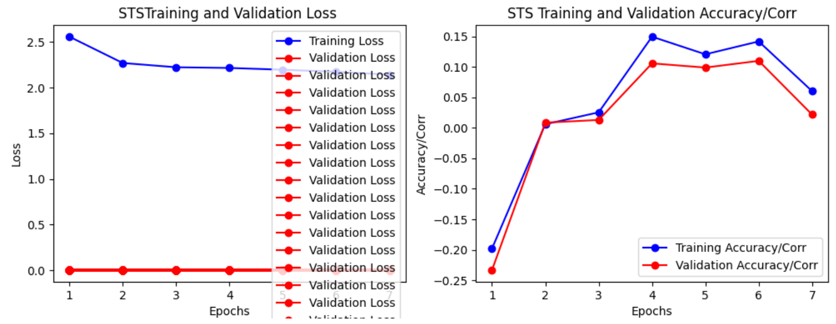
LoRA Training:



(a) Training curves SST



(b) Training Curves PARA



(c) Training Curves STS

Figure 4: Analysis Graphs

Trainable Parameter Reduction:

Table 1: LoRA Results

Dataset	Configuration	Rank	Dev Accuracy/Correlation
4*SST	Self-attention only	4	0.712
	Self-attention only	8	**0.743**
	Self-attention only	16	0.732
	Self-attention and FFN	4	0.705
PARA	Self-attention only	8	0.627
STS	Self-attention only	8	0.15
Test Leaderboard Results			
SST	Self-attention only	8	**0.471**
PARA	Self-attention only	8	**0.372**
STS	Self-attention only	8	**-0.037**

Number of frozen parameters: 109360128
Number of total parameters: 109629696
Parameter Reduction: 99.75%

GPU usage:

Average GPU usage: 1222.76 MB
Maximum GPU usage: 1224.94 MB

We achieved a LoRA implementation with 99.75% fewer trainable parameters and with the Dev Accuracy higher than for the Full Model and Last Linear Layer fine-tuning baseline model in sentiment classification. This increase in performance was expected, and it can even be said more than expected as LoRA usually only brings similar or slightly better performance than full fine-tuning. In our LoRA variations exploration, we achieved the best results with rank 8 and when the LoRA matrices were only applied to the self-attention layer. Despite good performance on the dev accuracies (SST/PARA datasets), the correaltion for the STS dataset seems rather low, being close to 0 and therefore signaling no correlation at all. Overall, despite the tarining loss steadily declining, we saw over-fitting for the SST dataset and reduction in training accuracy/correlation on the PARA and STS datasets after 4 or 5 epochs. Finally for the test leaderboard, we did not achieve promising results for all three tasks compared to the performance of other apporaches in the leaderboard, thus showing that despite LoRA bringing increases in performance, it is not a suitable approach for small models like minBERT on these tasks.
Prefix Tuning:
Trainable Parameter Reduction:

Table 2: Prefix Tuning Results

Dataset	Prefix Length	Dev Accuracy/Correlation	Test Leaderboard
SST	10	0.381	0.230
PARA	10	0.677	0.532
STS	10	0.130	0.019

Number of frozen parameters: 109482240
Number of total parameters: 109474560
Parameter Reduction: 99.99%

GPU usage:

Average GPU usage: 1215.34 MB
Maximum GPU usage: 1216.48 MB

The prefix tuning results indicate varied performance across different datasets. For the SST dataset, a prefix length of 10 resulted in a development accuracy of 0.381 and a test leaderboard accuracy of 0.230. These were lower than baseline and LoRA models. In the paraphrase detection task (PARA), the same prefix length yielded a higher than LoRA dev accuracy of 0.677 and a test leaderboard accuracy of 0.532, demonstrating the effectiveness of prefix tuning in this context. The semantic textual similarity (STS) task showed a lower performance than LoRA, with a development correlation of 0.130 and a test leaderboard correlation of 0.019. These results highlight the differential impact of prefix tuning across various NLP tasks, with notable success in paraphrase detection compared to sentiment classification and semantic similarity where it performed poorly.

6 Analysis

The prefix tuning results provide valuable insights into the qualitative performance of the PEFT-enhanced minBERT model across different tasks. The observed performance variations across datasets suggest several potential causes and areas for improvement.

For the SST dataset, the prefix tuning approach resulted in lower development and test accuracies compared to both the LoRA implementation and the baseline models. This could be attributed to the inherent complexity of sentiment classification tasks, which might require more nuanced understanding and representation capabilities that prefix tuning, with its fixed-length prefixes, may struggle to capture. The lower performance indicates that the fixed prefix length might not provide enough flexibility for the model to adapt to the diverse sentiment expressions in the data. To address this, experimenting with dynamic or variable prefix lengths could help the model better capture context and nuances. Additionally, combining prefix tuning with other fine-tuning methods, such as LoRA, might leverage the strengths of both approaches and improve performance. Enhancing the training data with augmented examples could also help the model generalize better to varied sentiment expressions. In the paraphrase detection task, prefix tuning demonstrated significant effectiveness, surpassing the LoRA approach in development accuracy and achieving a respectable test leaderboard accuracy. This success suggests that prefix tuning can effectively capture the relationships between sentence pairs, which is crucial for paraphrase detection. The fixed-length prefixes likely provided a consistent context for evaluating sentence similarity, contributing to the higher performance.

For the STS dataset, prefix tuning showed lower performance compared to the LoRA implementation. The low development correlation and test leaderboard correlation suggest that the fixed-length prefixes were not effective in capturing the subtleties of semantic similarity between sentences. The STS task requires a nuanced understanding of context and meaning, which fixed-length prefixes might fail to provide. Implementing adaptive prefixes that can change length or content based on the input might better capture the semantic nuances required for the STS task. Integrating external semantic resources or knowledge graphs during fine-tuning could also provide additional context and improve the model's understanding of semantic similarity.

The varied performance across different tasks highlights the strengths and limitations of prefix tuning. While it shows promise in tasks like paraphrase detection, it struggles with sentiment classification and semantic textual similarity. This indicates that prefix tuning may be more suited to tasks where the relationship between inputs is relatively straightforward and less reliant on nuanced contextual understanding. In general, it might be a valuable next step to add more layers to the BERT model and then apply prefix tuning and LoRA to these.

7 Conclusion

The main findings were that the PEFT strategies implemented did improve performance compared to the full fine tuning model but did not achieve good results in the leaderboard. They did, however, improve the parameter efficiency of the model by reducing the number of parameters being used significantly, as demonstrated by the results above and the GPU usage. In future work, we would apply these methods to a larger model to see how it would improve performance when there are more parameters. We would also run the model with the extensions both implemented to see if the performance is improved when there are both LoRA and prefix tuning implemented into the model so the model is able to handle more complex tasks.

8 Ethics Statement

Implication 1: Bias Amplification One significant concern is the potential for these models to amplify existing biases present in the training data. For instance, in sentiment classification, if the training data contains biased sentiments toward specific demographic groups, the model may perpetuate and even amplify these biases, leading to unfair and discriminatory outcomes in real-world applications such as automated hiring systems or customer service bots.

Mitigation Strategy: To mitigate this, it is essential to perform thorough bias audits on the training data and the model outputs. Techniques such as fairness-aware learning, which adjusts the training process to minimize bias, can be employed. Additionally, integrating diverse and representative datasets during training can help in reducing bias. Regular monitoring and updating of the models with new data that reflects diverse perspectives is also crucial to ensure fairness over time.

Implication 2: False Positive Paraphrase Detection In the context of paraphrase detection, a significant real-world implication is the risk of false positives—incorrectly identifying two non-paraphrase sentences as paraphrases. This can have serious consequences in applications such as academic integrity systems, where wrongly flagged paraphrases might unjustly accuse students of plagiarism, or in content moderation, where different but related content might be erroneously treated as duplicates and removed.

Mitigation Strategy: To address this issue, implementing a robust validation and verification process is crucial. One potential strategy is to use ensemble methods, where multiple models with different architectures and training regimes are combined to reduce the likelihood of false positives. Additionally, human review processes can be integrated for cases where the model's confidence is low, ensuring that critical decisions are verified by humans. Regularly updating the model with new examples of non-paraphrases and edge cases can also improve its ability to distinguish between paraphrases and non-paraphrases accurately.

References

[1] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. 2023.

[2] 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 *Proceedings of ICLR*, 2022.

[3] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, January 2021.

[4] 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.

[5] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52, 2008.

[6] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, 2013.

A Appendix (optional)

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc. that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.