Optimizing minBERT for Downstream Tasks using Multitask Fine-Tuning

Stanford CS224N Default Project

Carlos Ayala Bellido
Institute of Computational and Mathematical Engineering
Stanford University
cayalabe@stanford.edu

Abstract

In 2018, developers at Google created one of the most impactful LLMs, the Bidirectional Encoder Representations from Transformers, more commonly referred to as BERT. As an LLM which handled multiple tasks, BERT helped shape our understanding and approach to development of LLMs by using transformer architecture, paving the way for the advanced LLMs being implemented today. For this default project, I was tasked with implementing a minimalist version of BERT, referred to as minBERT, as well as providing an approach to improve performance on the downstream tasks of sentiment analysis, paraphrase detection, and semantic textual similarity (STS). For my project, I implement multitask fine-tuning using the 2019 research paper "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning" by Asa Cooper Stickland and Iain Murray as a basis. My work, outlined in the paper below, seeks to understand whether implementing multitask fine-tuning in minBERT results in significant improvement on the performance of the previously stated downstream tasks, as well as determining whether the increase in resource usage is worth it.

1 Key Information to include

- Mentor: Olivia Y. Lee
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

When it comes to implementing large language models like BERT that can handle multiple tasks, the primary objective is finding out how to balance training and testing to make sure each task is performing sufficiently well. Often, these downstream tasks are not sufficiently related so as necessarily benefit from singular task training in an ideal environment. In language models like BERT, this holds true, as even the minimalist implementation minBERT handles three distinct tasks with different sizes and domains for their outputs.
As such, developing fine-tuning techniques which maximize success across all tasks which the LLM is designed to handle is a priority in these models, especially if the fine-tuning can be done with minimal additional resource usage. For fine-tuning, there are a few approaches which have been implemented to handle this issue. The most common, and naive, approach is to determine which task provides the best average performance among downstream tasks and fine-tune on that task. While the model will likely improve performance on that task, there are no such guarantees for the other tasks across epochs of fine-tuning. Another approach, which is explained in further detail in the “Related Work" section below as it pertains to the Stickland and Cooper research paper, sees the introduction of new transformer designs and model architecture to aggregate the results of fine-tuning on a set of

tasks. This aggregate is then used to help train the model to best approach all of the tasks at hand. As explained in the “Approach" section below, my approach takes inspiration from the latter while recognizing the significant difference in resources that I have at my disposal. Given that there are three downstream tasks which minBERT should develop predictions for, my plan for this project was to determine which combination of tasks to fine-tune the minBERT model on, based on the accuracy and correlation scores obtained from these tasks. Various different combinations of tasks and training epoch quantities should allow for greater understanding as to how multitask fine-tuning improves performance.

3 Related Work

To provide research context for the work described in this paper, we look to the 2019 research paper "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning" by Asa Cooper Stickland and Iain Murray. This paper served as the basis for the extension of my project, specifically when considering how to approach multitask fine-tuning. For this paper, Stickland and Murray consider the effects of multi-task learning, specifically when used to consolidate the learning process of different tasks with different input and output spaces. They recognize, as I mentioned earlier, that previous implementations show individual task fine-tuning would be best; however, they work to show that they can provide better results than the baseline model while implementing multitask learning with modifications to model architecture. They evaluate their performance using the GLUE benchmark, which involves a series of datasets handling "question answering, sentiment analysis, and textual entailment" (Stickland and Murray, 2019), among other tasks. They explore this style of learning using adapters with shared layers on top of existing layers in order to implement hard-parameter sharing while considering approaches which allow them to reduce the parameter size increase compared to other fine-tuned models and implementations previously attempted.
There are four main contributions which this paper provides. First, this paper provides context as to how implementing multitask fine-tuning can increase performance in reference to the GLUE benchmark. However, not all of the tasks are improved, something which is individual fine-tuning permits. Additionally, this paper introduces the transformation known as "Projected Attention Layers", or PALs for short, which allow for higher amounts of parameters to be considered using simple low-rank transformations. Rather than implementing modifications to existing layers, PALs adds an extra set of layers to combine the results of the BERT tasks. The third contribution is providing a method for implementing schedule training. The model can now "sample tasks proportional to their training set size", using this as a starting point, and decrease importance with sampling size as the training continues (Stickland and Murray, 2019). Finally, this paper provides additional context to other models on top of their own, as they compare other model architectures which implement self-attention in terms of the alternative adaptation modules used.

4 Approach

When it comes to the multitask iteration of minBERT, there are a few key sections of the model to consider. When extracting embeddings, whether in the single task or multitask classifier versions of minBERT, there are three steps which this neural network is designed to take when performing the forward step. First, it calls upon the embeddings and the pooler to produce a token transformation of the embedding layers. Having done this, it then passes the embeddings through a dropout layer, which uses a hidden dropout probability parameter to randomly drop values. Finally, the outputs of the dropout layer are passed through a linear layer, which will return back a set of outputs corresponding to the number of logits needed for the task.
This is relatively straight forward for sentiment analysis, given that both the single and multitask implementations of minBERT do not need any further processing beyond this point. However, for the paraphrase detection and semantic textual similarity downstream tasks, there is a bit more to the process. Specifically, since each of them has two embedding inputs (as we are comparing strings), then there will be two sets of outputs from the dropout layer. As such, we add together the results of these two layers to form one layer, combining them before passing this aggregate into the linear layer specified earlier. This is a naive approach, as there are likely better methods for combining the two dropout outputs in a way which preserves information and context at a higher level.
As for the multitask fine-tuning, this is implemented in series with similar tendencies as the single task fine-tuning already implemented in the starter code. We modify the training function to now

include options to train based on paraphrase detection and semantic textual similarity, where the loss on each batch of training will use binary cross entropy and mean squared error respectively. There are also weightings applied to each set of training task: the value will determine if and how many times the model will be trained on a task within an epoch. For example, if we assign a weight of 0 to the paraphrase detection task, then the model will not be trained on that task. Finally, the model will detect improvements and save better models by using the average of the dev scores on each of the tasks and determining whether the current epoch is getting a better average score.

5 Experiments

5.1 Data

In this project, I use the following datasets:

- Stanford Sentiment Treebank (SST)
- CFIMDB
- Quora Question-Pair (QQP) Dataset
- SemEval STS Benchmark Dataset

As is clear from the list above, I'm only using the datasets which were provided in the starter code for the default project. Given that the research task which I have selected does not require any additional preprocessing on the data (since it involves improving on the evaluation metrics already available in the initial implementation of minBERT), the datasets remain unmodified between the initial implementation training and development on the fine-tuning extension.

5.2 Evaluation method

The evaluation of this task will rely on the performance of the fine-tuned version of the miniBERT model on each of the downstream task. As outlined by the Strickland and Murray paper referenced above and standard GLUE score practice, I'll be calculating this performance using accuracy for sentiment analysis and paraphrase detection, while STS performance will be measured using Pearson correlation coefficients. The scores I get from my fine-tuned model will be compared both against the baseline minBERT model, which is pretrained using the sentiment analysis task as done in the initial implementation. This should provide significant context against whether different versions of multitask fine-tuning are providing improvements in task performance.
Additionally, I will compare these results against the scores obtained from the referenced paper. While I'm not completely certain the datasets are the same as those which are provided, the Quora Question-Pair and STS datasets listed in this paper seem to match with the names of the datasets provided in the starter code. Therefore, I am only confident in my ability to compare the new iterations of fine-tuning to PALS model for performance on the Paraphrase Detection and STS downstream tasks. With these comparisons, we should be able to determine if the new fine-tuned minBERT model is performing better when the model outperforms both the baseline and research paper results for each of the individual tasks, with a focus on the average score across the downstream tasks.

5.3 Experimental details

This project was run on through a Deep Learning VM instance on Google Cloud, using a NVIDIA T4 GPU. Pre-training and fine-tuning used learning rates of 1e-3 and 1e-5 respectively, as is done with the single task initial minBERT implementation. While the intention was to run each of these tasks for ten epochs in training, resource issues began to arise as a single epoch of training on the paraphrase detection task takes at least one hour. This would mean performing training for ten epochs would require, at minimum, 10 hours of time on the GPU, disregarding training and evaluation time for any other tasks involved in the multitask fine-tuning. This, combined with early fine-tuning attempts with other tasks that showed minimal improvement over a large set of epochs, meant a new approach was considered. First, for all combinations tasks, produce results after fine-tuning for a single epoch. Second, for tasks which only involved training on either the sentiment analysis or semantic similarity tasks, perform up to ten epochs of fine-tuning as well, given time constraints. Finally, perform fine-tuning over one, three and five epochs when fine-tuning on all tasks simultaneously. This would allows us to not only see which tasks gave higher levels of performance based on initial epochs of fine-tuning, but also whether multiple epochs of fine-tuning are needed in a such way that would provide significantly better results.

5.4 Results

As outlined above, there were multiple combinations which were tested in the multitask fine-tuning with regards to which tasks performed best. The results of these tasks performance can be seen in the following table:

Model Version	# Epochs	SA Acc.	PD Acc.	STS Corr.	Score
Baseline - SST Pretraining	10	0.405	0.552	-0.067	0.4745
Baseline - SST Finetuning	10	**0.53**	0.554	-0.094	0.51233
Research Paper - PALS Model	N/A	N/A	0.715	**0.858**	N/A
SA Finetuning	1	0.49	0.593	-0.1	0.511
STS Finetuning	1	0.21	0.625	0.34	0.50167
STS Finetuning	10	0.24	0.625	0.39	0.52
PD Finetuning	1	0.198	0.77	-0.042	0.48233
SA + PD Finetuning	1	0.336	0.771	0.052	0.54433
STS + PD Finetuning	1	0.213	0.744	0.347	0.5435
SA + STS Finetuning	1	0.47	0.611	0.344	0.58433
SA + STS Finetuning	3	0.511	0.51	0.385	0.571167
SA + STS Finetuning	5	0.511	0.51	0.385	0.571167
SA + STS Finetuning	10	0.511	0.51	0.385	0.571167
All Tasks Finetuning	1	0.286	0.763	0.346	0.574
All Tasks Finetuning	3	0.376	**0.783**	0.383	0.616833
All Tasks Finetuning	5	0.45	0.766	0.36	**0.632**

Table 1: *Dev Dataset Metrics for Downstream Tasks on baseline and alternative fine-tune models*

In the table above, each of the values represents results the model obtained on that specific downstream task using the dev set. Note that bold indicates best value for metric. The parameters of the table are defined as follows:

- **Model Version**: This refers to the way the model was defined in terms of input parameters. Models are defined by the number of training epochs, whether the model was running on the "pretrain" or "finetune" option, and which tasks were used during model training.
- **SA Acc.**: This refers to the accuracy score on the Sentiment Analysis downstream task.
- **PD Acc.**: This refers to the accuracy score on the Paraphrase Detection downstream task.
- **STS Corr.**: This refers to the Pearson correlation coefficient score on the Semantic Textual Similarity (STS) downstream task.
- **Score**: This is modeled after the leaderboard scoring method, where each of the evaluation metrics values obtained for the downstream task is adjusted to a value in the range of [0,1] over the space of that metric's domain, after which all metrics are averaged. The formula for this is as follows:

$$\text{Score} = \frac{\text{SA Acc.} + \text{PD Acc.} + \frac{1}{2}(1 + \text{STS Acc.})}{3}$$

As we can see above, the performance of a majority of the fine-tuned model iterations provides a higher level of performance compared to the baseline models when comparing the scores. In fact, the only models which underperform are the three models which perform single-epoch, singular task fine-tuning. The fine-tuned baseline model still performs best at Sentiment Analysis accuracy, which is expected given the focus of training, and the PALS model from the Stickland and Cooper paper performs best on STS correlation. However, the multi-epoch versions of the models fine-tuned using all tasks perform best when it comes to Paraphrase Detection accuracy and overall score. When considering that, due to time limitations, this set of fine-tuning was improving on performance nearly every single epoch, then it is quite simple to consider that this trend should continue, given more

epochs of training time. As such, these are positive indications that multitask fine-tuning can be significantly impactful. Given the performance of the five epoch, all task finetuning model, this was the iteration which was submitted to the leaderboard, giving a leaderboard score of 0.621.
Beyond looking at individual results, another focus of this project was to see how much implementing multitask fine-tuning improved training and evaluation. As such, we can look to the following graphs to see how performance across the board for the minBERT model, training for 10 epochs, differed by training on STS by itself compared to training on both STS and SST.

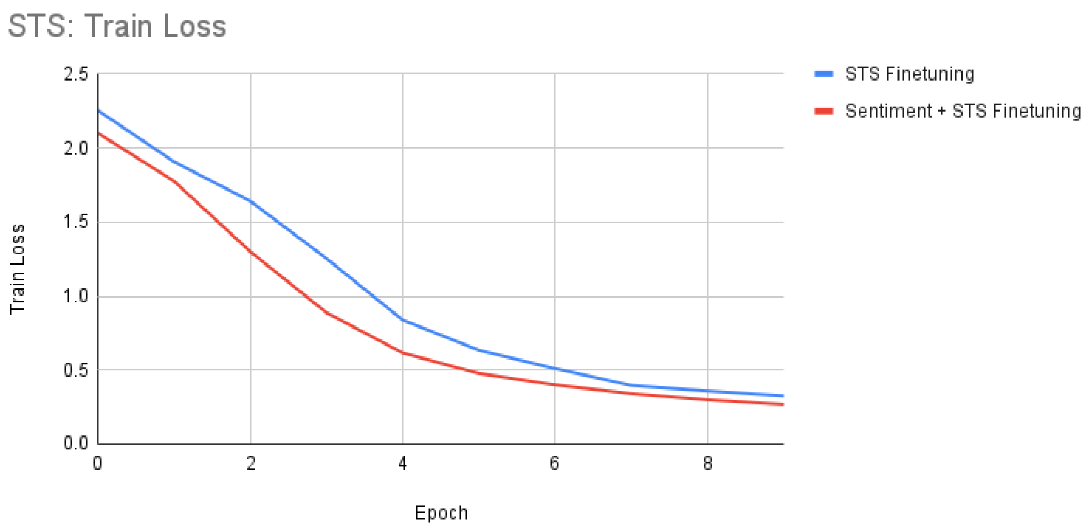

Figure 1: *Training Loss across Epochs for Semantic Textual Similarity (STS)*

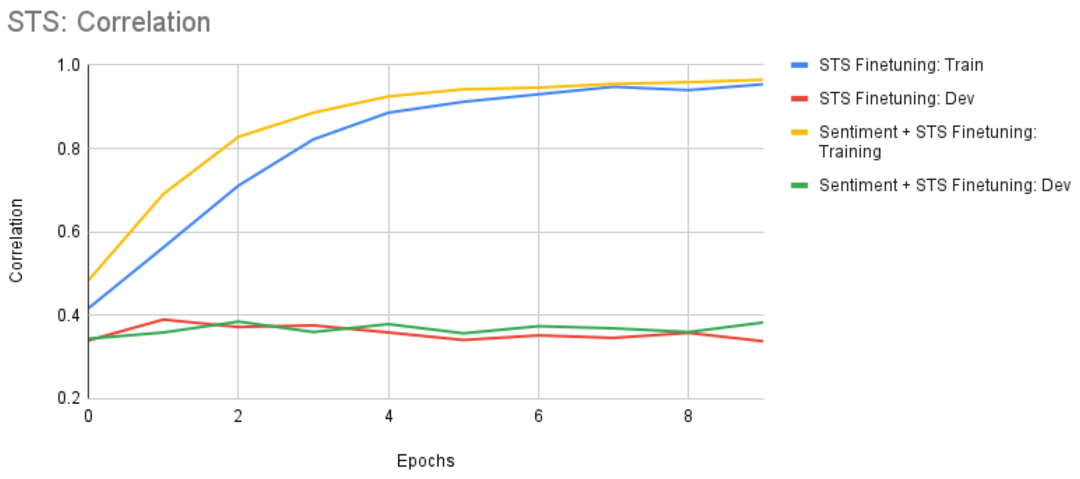

Figure 2: *Pearson Correlation Coefficient across Epochs for Semantic Textual Similarity (STS)*

The two models perform differently based on the tasks which is being evaluated. With the STS downstream task, we can see that the model which involves both STS and SST in finetune training experienced less training loss, as well as performing better on training and dev Pearson correlation coefficients for the majority of the epochs, though not by much.

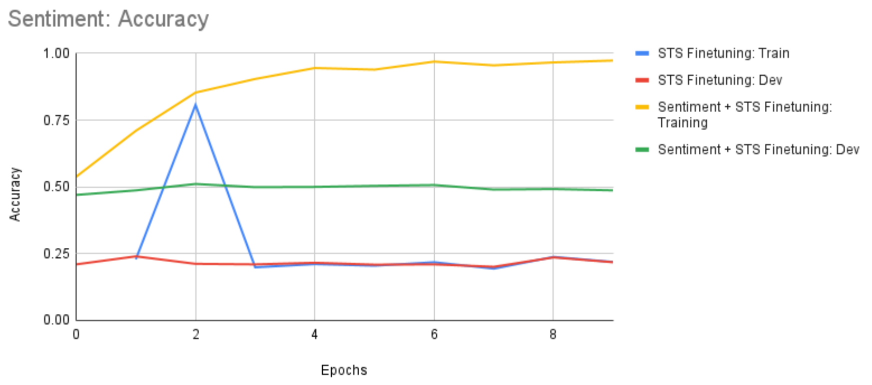

Figure 3: *Accuracy across Epochs for Sentiment Analysis*

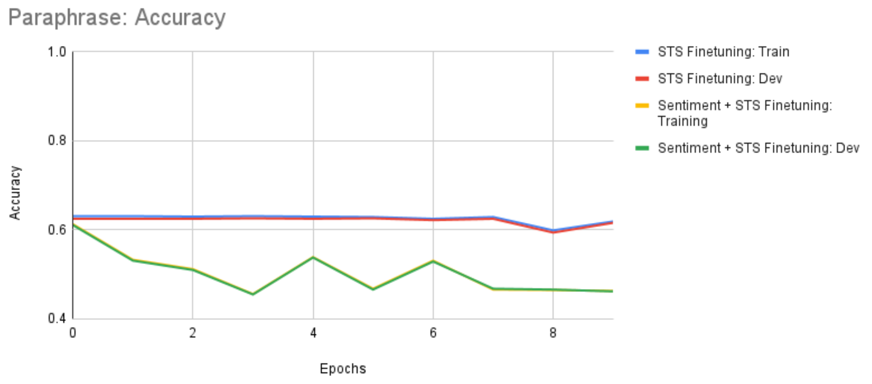

Figure 4: *Accuracy across Epochs for Paraphrase Detection*

The same relationship holds for the SST training and dev accuracies across epochs, thought not nearly as close as before. However, we observe an interesting relationship when it comes to the Paraphrase Detection downstream task. When it comes to accuracy, the performance on training and dev datasets are similar within each model; however, the minBERT model finetuned solely on STS provides a significantly higher accuracy on this downstream task.

Finally, given the success of the all-task finetuning, it would be helpful to consider how metric performance improved across epochs for this set of fine-tuning.

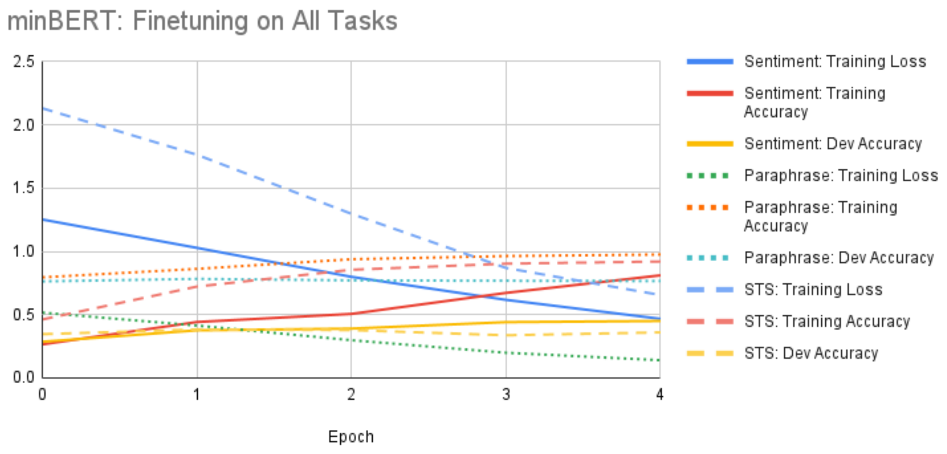

Figure 5: *Metric Performance across Epochs for finetuning using all downstream tasks*

As we can observe from Figure 5, there are significant drop offs in loss for all three tasks as we progress through the epochs, especially when it comes to STS. As for task performance, it does seem that Paraphrase Detection is the downstream task which improves to the highest level in training and dev accuracy, with STS and Sentiment Analysis following closely behind.
As shown in the graphs above, training more tasks does not always result in uniform improvements in performance across tasks. In the case from above, fine-tuning the minBERT model for certain sets of tasks actually cut down the performance the model had on tasks which were not included in fine-tuning, compared to an iteration of fine-tuning which only trained on a singular class. On the other hand, the work here shows that a model which is fine-tuned on multiple tasks doesn't necessarily sacrifice performance across the tasks used in the training process; rather, there are chances where the model could maintain or improve performance, even given the same quantity of training time. As evidenced by the all task fine-tuning iterations of the minBERT model, applying fine-tuning with all tasks can improve performance signficantly across epochs. Thus, there is success in the way that this model has been implemented to use dropout and linear layers in combination with training in series.

6 Analysis

When analyzing the work above, the key focus here seems to be that multitask fine-tuning does have potential improvements on a baseline single task fine-tuned model; however, it's important to understand where these improvements or limitations come from.
Some of the limitations to the claims which can be made mainly lie in the naivety of the model. Considering the size of the neural network generated to produce these results, there are very few layers used to process data from input to final output. Given that these are linear layers as well, the amount of information maintained between layers is likely not kept at a maximum. Even comparing the cases for the paraphrase detection and semantic textual similarity tasks, the way which the outputs from the dropout layers are combined is not using any major calculations to detect whether information from previous layers is being properly preserved. As such, understanding the failures of the model as currently implemented lies in determining whether implementing different types of layers or new formulas to combine layers provides better results.
Additionally, looking into how the multitask fine-tuning is approached reveals that there are significant improvements which can be made to developing the training process. At the moment, tasks are trained in series: first the semantic classification, then paraphrase detection, and finally, semantic textual similarity. While model states are not stored until the end of this chain of training, it also means that each set of task training acts independently. To get a stronger insight into how multitask fine-tuning could improve, intertwining batches from each set of a task's training dataset could allow for the learning and improvement of the model to more accurately intertwine the parameter updates from learning through each individual task. Finally, a look through the code shows that there are some design choices made early in the process which could be changed and have a sizeable impact on the performance. An example of this is with how this implementation of multitask fine-tuning training decides when to update the model. As part of an earlier decision with determining average score, the training method only considers the tasks which were being used in the fine-tuning training to denote the performance of the model, rather than the performance of the model on every task. This, in theory, means that some models throughout this process have been discarded unfairly, in that the improvement in performance from tasks not being used in the fine-tune training significantly outweight the decrease in performance from those which were being used.

7 Conclusion

With this work, the path for multitask fine-tuning does not seem as bleak as previously indicated. Going into this project, I expected to implement the extension and see negligible improvements on the baseline model, if that. The literature referenced earlier supported this claim as well, as their findings found that having to implement a new architecture was the way they were able to get improvements on the results, yet multitask fine-tuning on its own would not provide significant improvements. However, the work I outlined over these pages shows that, even with limited resources, there is a distinct possibility that multitask fine-tuning can provide significantly improved results on top of what is already possible. As for future steps, resource allocation and availability plays a major role. With many of the fine-tuning combinations I wanted to test, there were limitations based on the GPUs and time I had available to run this project. Testing out how more resources and streamlined algorithms could speed up the training and evaluation process could prove important to getting better results than what is already shown. Additionally, since it seems that multitask fine-tuning does provide improved accuracy and correlation metrics, it would be interesting to determine at what point the amount of tasks involved cuts off the benefits of this method of fine-tuning. LLMs can implement larger amounts of tasks, so to see how this method plays with the limits of larger LLMs could develop new sets of questions as to how to design subsets of tasks to perform multitask fine-tuning.

References

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