Multi-task BERT Fine-Tuning with Gradient Tricks

Stanford CS224N Default Project

Henry Ang
Department of Computer Science
Stanford University
henryang@stanford.edu

Abstract

This project explores advanced techniques for improving multi-task learning in language models, focusing on mitigating gradient conflicts and leveraging dataset imbalances. We used a combination of four primary methods: PCGrad, sample-weighted PCGrad, two-step fine-tuning, and model ensemble. PCGrad, a technique that projects conflicting gradients, was used to address gradient conflicts during multi-task model fine-tuning. We introduced sample-weighted PCGrad, which extends PCGrad by proportionally scaling gradients to balance tasks with differing dataset sizes. We also introduced a two-step fine-tuning process inspired by meta-learning, where a model is first trained on multiple tasks and then fine-tuned separately for each task to specialize on individual tasks further. Finally, we utilized a model ensemble to aggregate predictions from the best-performing models, enhancing overall accuracy.

We implemented and evaluated these methods using three tasks: sentiment analysis, paraphrase detection, and textual similarity. Our experimental results show that sample-weighted PCGrad combined with two-step fine-tuning achieved significant performance improvements over the baselines. The model ensemble approach further boosted the predictive performance across all tasks. Our best model achieved an overall test score of 0.792, ranking fifth on the test leaderboard.

1 Key Information

- TA mentor: Timothy Dai
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Fine-tuning pre-trained large language models for specific language tasks is one of the most intensively researched areas in natural language processing. This process is crucial for enhancing the performance and usefulness of LLM products, making them more adaptable and effective across a variety of applications. The ability to tailor these models to handle multiple tasks simultaneously can significantly boost their utility and efficiency.

A promising approach to achieving this is multi-task learning, where a single model is trained on several related tasks. However, a major challenge in multi-task learning is the conflict between gradients of different tasks. These conflicts can lead to suboptimal updates, impairing the model's performance on individual tasks and overall learning efficiency.

To overcome this limitation, we employed the Projecting Conflicting Gradients (PCGrad) [1] technique. PCGrad modifies the gradient of each task by projecting out the conflicting components

from other tasks. Building upon PCGrad, we introduced an innovative extension: sample-weighted PCGrad. This approach assigns different weights to tasks during gradient updates, addressing the issue of varying dataset sizes. For instance, the Quora dataset in our project is significantly larger than the other datasets. By adjusting the weight of the Quora dataset, we can leverage its extensive data without letting it disproportionately influence the learning process.

Additionally, we developed a two-step fine-tuning process inspired by Meta Learning [2]. The first step fine-tunes a multi-task model on different tasks using the techiques mentioned above to develop a strong foundation. The second step further fine-tunes the model from the first step separately for each task. This two-step process ensures that the model benefits from multi-task dataset exposure and targeted fine-tuning, leading to improved performance.

Lastly, through an ensemble of our best performing models, we achieved overall performance that far exceeds the baselines.

3 Related work

BERT The cornerstone of this project is the BERT model [3]. It introduced many revolutionary and influential ideas, including using bidirectional approach for better context understanding, setting the pre-training and fine-tuning paradigm for large language models and demonstrating the superior effectiveness of transformers compared to traditional architectures. Pre-trained BERT is a good choice for solving the downstream tasks in this project with the various fine-tuning mechanisms we implement.

Multi-Task Learning Multi-task learning [4] is a paradigm in machine learning where the model simultaneously trains on multiple tasks. With shared parameters between tasks, multi-task learning generalizes on tasks better and mitigate overfitting compared to single-task training. Multi-task learning is applied to a wide range of machine learning fields, including natural language processing. BERT and PALs [5] outlines methods to enhance BERT's performance in multi-task learning while substantially reducing the number of parameters required compared to fine-tuning models individually. Our project uses a similar approach. We use different prediction heads for different tasks and share most model parameters across tasks. Since the paper demonstrates that the multi-task BERT model exceed the performance of individually trained models, we expect our model to improve over the baselines with a multi-task structure and additional training techniques.

Gradient Methods In multi-task learning models, one common pitfall is conflicting gradients and negative transfer [4]. Because of the heterogeneity of the tasks, the gradient from different tasks may be conflicting, causing the training to be unstable and the model to under-perform single-task models. Projecting Conflicting Gradients [1] is a prior work that proposes a gradient method to reduce gradient conflicts by projecting gradients onto the normal space of conflicting gradients. In addition to applying the method from the original PCGrad paper, we also propose an extension that modifies the weighting of gradients based on dataset size. We expect our addition to further improve training stability and performance. Another prior work that we drew inspiration from is Multitask Learning as a Bargaining Game [6]. This paper proposes to use the Nash Bargaining Solution to make each tasks maximize its performance without compromising the others. Our gradient weighting approach is similar to this work because tasks with more samples will try to improve its performance by training on more samples while not dominating the gradient updates.

4 Approach

4.1 Main Approaches

In this project, we iteratively improved our model by applying more techniques. The four methods that we used are PCGrad, sample-weighted PCGrad, two-step fine-tuning and model ensemble.

PCGrad One main motivation of the project is to investigate whether gradient techniques can mitigate the gradient conflicts between tasks during language model fine-tuning, and the main gradient technique chosen is "Projecting Conflicting Gradients" (PCGrad) [1]. At each gradient update step to the multi-task model, the algorithm iteratively modifies the gradient of each task with

respect to other tasks with the following formula:

$$g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} g_j$$

where g_i is the gradient being modified, and g_j is the conflicting gradient of another task. Figure 1 from the paper demonstrates that this operation eliminates the conflict between the two gradients.

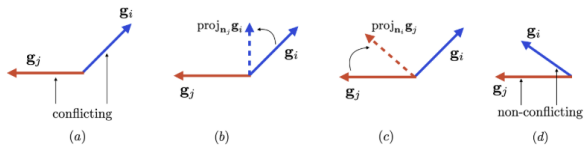

Figure 1: PCGrad

Sample-weighted PCGrad A novel technique we introduced in this project is sample-weighted PCGrad, which aims to mitigate the imbalanced dataset sizes of the different tasks. In the provided datasets, the Quota Paraphrase dataset far outsizes the other two tasks. When samples are taken from the datasets proportionally from the datasets, the Quota dataset may dominate the gradient updates with the original objective of PCGrad. Sample-weighted PCGrad mitigates this problem by proportionally scaling down the gradients of tasks with more samples, so that balance is maintained between the multiple tasks while still taking advantage of the samples in the larger dataset. The gradients are scaled with the following formula:

$$w_i = \frac{1}{n_i}$$

$$\hat{g}_i = w_i g_i$$

$$\hat{g}_i = \hat{g}_i - \frac{\hat{g}_i \cdot \hat{g}_j}{||\hat{g}_j||^2} \hat{g}_j$$

where n_i is the number of samples of a task i in a batch, w_i is the weight of task i, and $\hat{g}_i$ is the sample-weighted gradient of task i. The weighted task gradients are then updated with the PCGrad projections to resolve gradient conflicts.

Two-step Fine-tuning We took inspiration from Meta Learning methods, in which the meta learner first learns from a variety of tasks together and then adapts to individual tasks afterwards. We decided to take a similar approach to further reduce the effect of task gradient conflicts. As an analogy to the Meta Learning approach, the fine-tuned BERT layers is the meta learner. In the first step, we train a multi-task model with the gradient techniques above. In the second step, we make a copy of the weights in BERT for each task and reinitialize the parameters in the MLP layers. We then fine-tune each task individually on its own dataset to get one further fine-tuned model for each task.

Model Ensemble Lastly we explored model ensemble. The best performing models are used to compose an ensemble that predicts based on the prediction of all the models in the ensemble. For the sentiment analysis task, the ensemble takes the average of the predicted sentiment and rounds it to the nearest class. For the paraphrase binary classification task, the output is the majority vote. Lastly, for the text similarity task, the ensemble outputs the average of the models.

4.2 Baselines

Single-task Baseline Our first baseline fine-tunes the entire model for each task individually. For the SST dataset, the baseline uses the Project part 1 implementation. For the other two datasets that takes in sentence pairs as inputs, we experimented with different model architecture options. The initial approach we took was using BERT to create separate embeddings of the two sentences. The embeddings are separately transformed by a MLP layers, and the similarity is predicted using the dot product of the two sentences' embedding. Because this method did not work as well as expected, we experimented with another architecture, where the two BERT embeddings are concatenated and pass

through a MLP together to directly produce a prediction. This empirically resulted in vastly better performance than cosine similarity. Lastly, we experimented with concatenating the two sentences before the BERT layer to generate a single embedding. This single embedding is then passed through a MLP layer to generate the predictions.

Multi-task Basline Our multi-task baseline uses an architecture that makes the three tasks share the BERT parameters. Each task has a non-shared prediction head consists of MLP layers. The prediction heads used in the multi-task baseline are the same as the single-task baseline. The loss function of the multi-task model uses the native multi-task learning object, where $\mathcal{L}_i$ is the individual loss of task i.

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{N} \mathcal{L}_i$$

4.3 Originality and Implementation Statements

The originality we introduced in this project are the sample-weighting method for PCGrad and the two-step fine-tuning approach, both of which demonstrated substantial improvement in model performance.

The main implementation effort involved in this project include:

- Partial implementations of BERT as specified by Part I of the project
- Implementation of three different model architectures for sentence-pair tasks baselines
- Implementation of multi-task model baseline
- Adaptation of PCGrad implementation [7] in our multi-task model
- Implementation of sample-weighing in PCGrad and relevant custom data loaders
- Implementation of two-step fine-tuning process
- Implementation of model ensemble prediction

5 Experiments

5.1 Data

We use datasets provided by the default project. We use Stanford Sentiment Treebank dataset for sentiment analysis task, Quota data set for the paraphrase detection task, and SemEval STS Benchmark Dataset for the sentence similarity task.

5.2 Evaluation Method

We use the metrics described in the default project description. When determining the overall performance of a model, we use the same method used by the leaderboard calculation: $\frac{1}{3}(\textbf{sst\_accuracy} + \textbf{paraphrase\_acc} + \frac{1}{2}(1 + \textbf{sts\_correlation}))$.

5.3 Experimental details

This section describes the experiments conducted and provides preliminary analyses that are relevant to the designs of the experiments. More in-depth quantitative and qualitative analyses of the overall results are provided in later sections.

Overview Model training and evaluation were performed on a Nvidia L4 GPU on a GCP VM. We performed approxmiately 30 experiments in total. We first built the baseline that fine-tunes the model separately. We implemented different prediction heads, experimented, and chose the best-performing ones. We built the multi-task baseline with the vanilla objective, which made substantial improvement over the single-task baselines on the overall performance. We ran preliminary experiments with PCGrad and improved it with the two-step fine-tuning mechanism and sample-weighted gradients. Lastly, we selected the best models to create a model ensemble to make ensemble predictions.

Default Hyperparameteres Through the initial experimentation with the baselines, we tested a few hyperparameters. We found the following settings to work well and are used by default if not mentioned otherwise in all experiments:

- Learning rate of 10^{-5}
- BERT hidden size of 768
- MLP hidden sizes of 128, 64 and 64 for the prediction MLPs for the SST, Paraphrase and STS tasks respectively
- Number of samples per epoch for the Quora dataset limited to 20,000 due to computation resources constraint
- Perform Fine-tuning on all the parameters instead of only the linear layers

Single-task Baseline In the training of single-task baselines, we experimented with different model architectures mentioned in the approach section for the two pair-sentence tasks. We found the performance of both tasks increased when we used MLP instead of cosine similarity for prediction. The experiments showed another substantial increase in performance when we concatenated the sentence tokens with a SEP token between the two inputs before BERT to get a single embedding for prediction. Accordingly, we decided to use the last architecture in the following experiments.

Multi-task baseline There are a few experimental details worth mentioning for the multi-task baseline model. First, we created a custom data loader which, in each batch, takes batch_size number of random samples for each task. This means in each epoch, instead of going through the whole dataset for each task, each task is trained on the same number of samples. This setup is necessary for not skewing the naive multi-task objective, where the loss of each batch is a simple sum of losses from individual tasks. We set the default number of samples per task per epoch to 20,000, which is close to the size of the SST and STS datasets.

We studied the effect of the number of epochs in the multi-task baseline experiments. We first attempted to run the model with 10 epochs, and we found the model had shown signs of convergence around the last epochs. We increased the number of epochs and found the extra epochs helped the model achieve a slightly better performance as shown in figure 2. With this observation, we increased the default number of epochs for the following experiments to ensure that the models achieved the best performance.

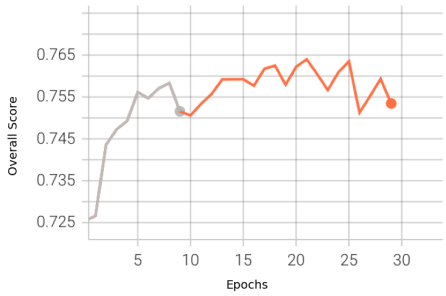

Figure 2: Overall Dev Performance of Multi-task Baseline

PCGrad For the PCGrad experiments, we used similar parameters and setup as the multi-task baseline, because the model's loss and architecture are the same as the multi-task baseline.

One issue we realized during the PCGrad experiments is that while the paraphrase and similarity tasks performance improved with training, the performance of the sentiment analysis task deteriorated over time. Figure 3 shows the dev performance of the three tasks over training epochs. To resolve this issue, we implemented the two-step fining approach next.

Two-step Fine-tuning The two-step fine-tuning approach introduced a second stage of training, thus requiring more hyper-parameter choices. For simplicity, we kept the same hyper-parameters for the first training step (multi-task training). We mainly needed to decide the number of epochs and

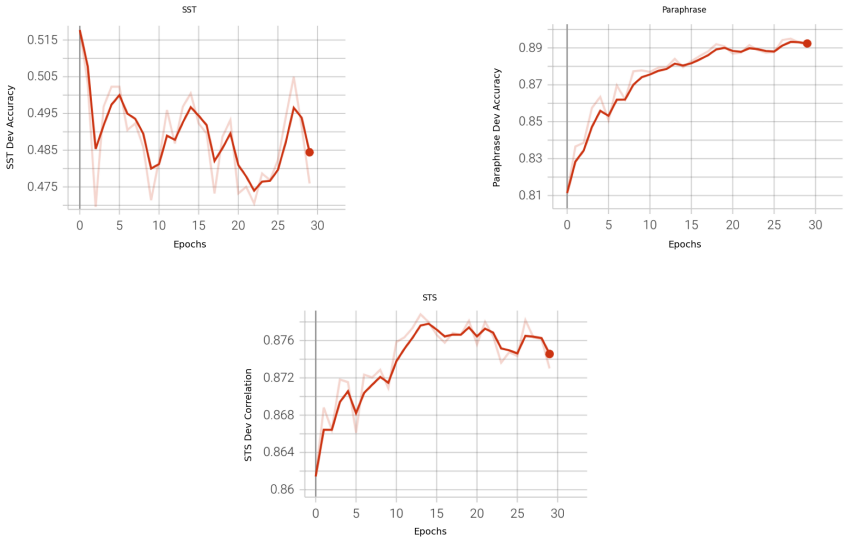

Figure 3: Individual Dev Task Performance of PCGrad model

learning rates for the second step. Empirically, we found that after around 50 epochs of training, all the tasks started to show convergence, and the learning rates of $1e-5$ or $3e-6$ both worked well. We found that the extra training did mitigate the issue with the SST task identified in the original PCGrad model. Figure 4 shows that SST dev accuracy did not decrease with more training epochs and achieved higher overall performance.

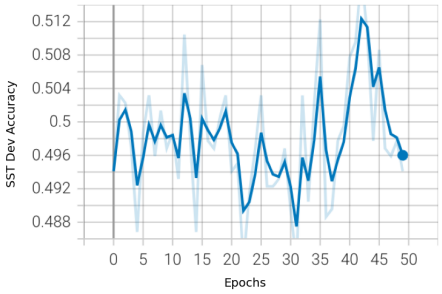

Figure 4: SST Dev Accuracy in the Second Fine-tuning Step

Sample-weighted PCGrad We implemented Sample-weighted PCGrad described in the approach section and modified the custom data loader to support a configurable ratio number of samples for each task in a batch. The sample-weighted PCGrad model is trained with the two-step fine-tuning process. The following table shows the ratio of the number of samples for different tasks in each batch and the overall performance. Note that there is not a configuration with $1:1:1$ because that is equivalent to using the original PCGrad.

SST Samples	Paraphrase Samples	STS Samples	Overall Dev Score
1	3	1	0.788
3	1	1	0.787
0	1	1	0.788
0	3	1	**0.790**

In the best-performing model, the proportion of the paraphrase samples is the largest, which matches the original motivation of taking advantage of the larger dataset size of the paraphrase task. Note that the relative ratio of SST samples in a batch is 0, which means that no SST sample was used at all in the multi-task training phase. The SST data is only introduced in the individual task fine-tuning in the second step.

Ensemble Lastly, we implemented a simple ensemble mechanism that takes the predictions from various individual models and outputs the ensemble's predictions. The model ensemble achieved better Dev and Test performance in all three tasks.

5.4 Results

The table below shows the Dev scores of the best-performing model of each category:

Model	SST Acc	Paraphrase Acc	STS Corr	Overall Dev Score
Single-task Baseline	0.511	0.787	0.699	0.729
Multi-task Baseline	0.510	0.885	0.797	0.764
Multi-task PCGrad	0.506	0.875	0.878	0.773
Two-step Fine-tuning PCGrad	0.510	0.894	0.883	0.782
Sample-weighted PCGrad	0.528	0.901	0.885	0.790
Ensemble	**0.530**	**0.904**	**0.889**	**0.793**

We only obtained one test accuracy score because of the limit on submissions attempts. Our submission currently ranks number 5 on the test leaderboard.

Model	SST Acc	Paraphrase Acc	STS Corr	Overall Test Score
Ensemble	**0.531**	**0.903**	**0.886**	**0.792**

The overall performance of the models match our expectations. All of our models out-perform the two baselines in overall score. Our best model achieved high paraphrase accuracy and STS correlation, while the SST accuracy is about 50%. This is likely because the task is framed as a classification task instead of a regression task like STS, which makes achieving high accuracy hard.

We also see clear improvements in each iteration of our models. As we implement more complex models and add more techniques, we see consistent improvement over our previous models.

6 Analysis

Regression in SST Accuracy One significant pattern we observe from the results is that the dev accuracy of the sentiment analysis task from the single-task baseline beats most multi-task models except for the sample-weighted model and the ensemble. Theoretically, the gradient methods should largely prevent negative transfer caused by multi-task learning. In practice, the overall score of the other two tasks improved substantially, resulting in better overall performance, while the SST task's accuracy decreased. We have two hypotheses for this result.

1. The paraphrase and the textual similarity tasks are very similar in nature. Despite one being a regression task and the other being binary classification, they both predict two sentences' semantic similarity. On the other hand, the sentiment analysis task evaluates the sentiment of a single sentence, which is a relatively different task. In a multi-task setting, the gradients from the paraphrase and STS tasks will likely benefit each other, but they will not necessarily improve the performance of the sentiment analysis task. This hypothesis is further corroborated by the result from sample-weighted PCGrad. The best model of this category used no sample from the SST data in the multi-task training phase. The removal of SST samples likely reduced noise for the multi-task learning for the other two tasks. In the second fine-tuning step, the SST surprisingly gained large performance increase. The samples from the other tasks likely helped provided useful knowledge to the BERT weights. As a result, we hypothesize that the heterogeneity of tasks in the multi-task learning phase is the root cause of this, but as long as the heterogeneous tasks are not trained simultaneously, they still may provide benefits.

2. Another contributing factor to the lower score may be the inherent difficulty in this task. The level of positivity in a sentence can be subjective. For example, here is one example in the dataset: " And that 's a big part of why we go to the movies . 3". It would not be strange if this is give a score of 4. The quantitative result also supports this hypothesis. While this task has an accuracy of about 50%, the other two are closer to having 90% accuracy or correlation. As a result, since the task is harder, and the multi-task learning optimizes for the overall score across 3 tasks, the gradient updates may favor the other two tasks that provide more concrete improvement on performance.

One or two BERT embeddings Another interesting finding from the project is that in the sentence-pair tasks, combining the two input sentences and producing one BERT embedding greatly outperforms generating individual embeddings and then make predictions. We hypothesize that it is the attention mechanism in BERT that produced this difference. Allowing attention mechanisms to have access to the other sentence will greatly help the model determine whether two sentences have similarity meanings or not.

7 Conclusion

In conclusion, this project demonstrated the effectiveness of advanced gradient techniques and novel fine-tuning strategies in improving multi-task learning for language models. By employing PCGrad to resolve gradient conflicts, sample-weighted PCGrad to address dataset imbalance, and a two-step fine-tuning approach inspired by meta-learning, we achieved substantial improvements in model performance. The addition of model ensemble further enhanced the predictive accuracy across tasks. Our experiments show that these methodologies not only mitigate common challenges in multi-task learning but also facilitate better generalization and efficiency. The insights gained from this study underscore the potential of integrating sophisticated gradient techniques and adaptive training phases to advance multi-task learning in NLP applications.

Possible future work for this project include: 1) Evaluate the models and techniques proposed in this paper in more NLP tasks to further investigate how well they generalize to different tasks. 2) Further hyper-parameter tuning to achieve better overall score and create a better model ensemble. 3) More investigation in possible different architecture choices for the SST task to reduce overfitting and improve task accuracy.

8 Ethical Considerations

One possible risk associated with the multi-task architecture of the model is the possible contagiousness of biases in one task dataset towards the other tasks because the language model's parameters are shared by the different tasks and are trained together. For example, if the text corpus in one task contains gender biases or discriminative content, shared parameters of the multi-task model can learn it and produce the same biases in other tasks. A possible mitigation to the issue is to add a pre-processing step for the datasets. For example, a classifier can be used to remove discriminative contexts from the dataset.

Another possible risk is the misinformation that the model could produce due to various datasets being trained together. For example, if one task uses a dataset containing science fictions and another task has scientific publications. The model may produce content using the knowledge learned from the distribution in the science fiction dataset when performing the task related to scientific publications, which would may mislead people who do not have expertise in relevant fields.A possible mitigation to the issue may be adding additional task-conditioning and attention mechanism in the model that help the model distinguish the task, which would help the model use the correct set of activations for the given dataset.

References

[1] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *NeurIPS*, abs/2001.06782, 2020.

[2] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *CoRR*, abs/2004.05439, 2020.

[3] 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.

[4] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796, 2020.

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

[6] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *CoRR*, abs/2202.01017, 2022.

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