Extension-BERT: Multitask Learning with BERT

Stanford CS224N Default Project

Jingwen Wu
Department of Computer Science
Stanford University
jingwenw@stanford.edu

Abstract

In this project, we extend the BERT Transformer model by performing multitask learning on three NLP tasks: sentiment classification, paraphrase accuracy, and semantic textual similarity. To improve model performance on these three tasks, we investigate four main methods: cosine similarity finetuning, additional Masked Language Modeling pretraining, additional dataset finetuning, and multitask learning optimization through gradient surgery. We find that gradient surgery optimization achieves the best results, with a Pearson correlation of 0.8517 on semantic textual similarity, 51.40% accuracy on sentiment classification, 85.17% accuracy on paraphrase identification, and an average score of 0.7392 on the Test leaderboard.

1 Key Information to include

- Mentor: Manasi Sharma
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

In Multitask Learning (MTL), an NLP model is simultaneously trained on multiple tasks using shared embedding representations. MTL is similar to human learning, where knowledge in one domain can be transferred to improve performance in related domains (Liu et al., 2019a). This paradigm has gained research attention due to its potential to increase data efficiency and model learning speeds, but is challenging because an improvement in the performance of one task could downgrade the performance of another task (Crawshaw, 2020).

The baseline model for this project is BERT, a bidirectional Transformer model with language representations pretrained on two tasks: Next Sentence Prediction (NSP) and Masked Language Modeling (MLM) (Devlin et al., 2018). The goal of this project is to extend the pretrained BERT model by simultaneously finetuning on three NLP tasks: 1) Sentiment Classification of five sentiment labels on the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), 2) Semantic Textual Similarity, a measure of the similarity in meaning of two sentences, on the SemEval STS dataset (Agirre et al., 2013), and 3) Paraphrase Identification on the Quora Question Pairs (QQP) dataset (Iyer et al., 2017). Sentiment classification has practical applications to areas such as customer product research, while semantic textual similarity can be applied to information retrieval and paraphrase identification can be useful for plagiarism detection. To finetune BERT on these three tasks, we adapt the following extensions: 1) cosine similarity finetuning (Reimers and Gurevych, 2019), 2) additional pretraining on the MLM task (Devlin et al., 2018), 3) additional finetuning on the SNLI dataset (Bowman et al., 2015), 4) gradient surgery for multitask learning (Yu et al., 2020).

We find that although the traditional multitask learning method already improves upon the baseline score by over 0.2 points, adding the PCGrad method yields the highest average model score. Since the PCGrad model performs best overall but not in any task-specific subcategory, we hypothesize

that the PCGrad procedure allows the model to step in the optimal direction for all three tasks while leading to tradeoffs in individual task performance, providing an improvement in average task scores.

3 Related Work

The Transformer architecture outlined by Vaswani et al. (2017) originally applied the self-attention mechanism and fully-connected feed-forward networks to machine translation. Devlin et al. (2018)'s BERT model, based on this Transformer architecture, advanced the state-of-the-art on 11 individual NLP tasks, including QQP, SST-2 (binary sentiment classification), and STS-B, but literature on multitask learning for our three datasets is limited. Liu et al. (2019b)'s RoBERTa model modified BERT through additional training methods, removal of the Next Sentence Prediction task, and adjustment of the masking strategy, thus advancing the state-of-the-art on STS-B and three other General Language Understanding Evaluation (GLUE) (Wang et al., 2018) tasks. Lan et al. (2019)'s ALBERT model significantly reduced the number of parameters in BERT, establishing a new state-of-the-art on the GLUE benchmark. Jiang et al. (2019)'s smoothness-inducing adversarial regularization and proximal point optimization methods, aimed at reducing the overfitting caused by aggressive finetuning, further advanced the state-of-the-art on SST-2, QQP, and STS-B.

Multitask learning using BERT through a formulation of the loss function as a sum of individual loss functions has been previously applied to domains such as News Recommendation (Bi et al., 2022). The MT-DNN model presented by Liu et al. (2019a) applies multitask learning to deep neural networks by incorporating BERT into its shared layers. An optimization method for multitask learning is the gradient surgery procedure, in which individual task gradients are deconflicted through projection of one gradient onto the normal plane of another (Yu et al., 2020). We build off of these methods to extend BERT for multitask learning on SST, STS, and QQP.

4 Approach

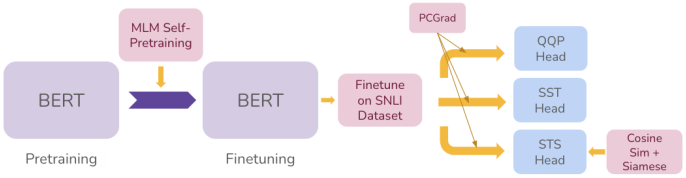

Figure 1: Overview of extensions added to the BERT model.

4.1 BERT Architecture

For this project, we use the pretrained sentence embeddings of BERT, whose full architecture is explained in-depth in Devlin et al. (2018).

4.2 Baseline

We create a baseline model which uses the pretrained BERT weights, adding dropout and one linear layer for each of the three tasks to predict outputs. We train the SST and QQP heads using Cross Entropy Loss, and the STS head using simple regression and Mean Squared Error Loss.

4.3 Multitask Learning

We build upon the baseline by adapting the approach outlined in Bi et al. (2022) of multitask learning by optimizing the following multitask loss function:

$$\mathcal{L} = \mathcal{L}_{\text{STS}} + \mathcal{L}_{\text{SST}} + \mathcal{L}_{\text{QQP}} \tag{1}$$

We use Cross Entropy Loss for QQP and SST, and Mean Squared Error Loss for STS:

$$\mathcal{L}_{\text{STS}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

$$\mathcal{L}_{\text{QQP}} = -\sum_{i=1}^{m_Q} y_i \log(\hat{y}_i) \tag{3}$$

$$\mathcal{L}_{\text{SST}} = -\sum_{i=1}^{m_S} y_i \log(\hat{y}_i) \tag{4}$$

where y is the true label, $\hat{y}$ is the predicted label, N is the number of data points, m_Q is the number of classes for QQP, and m_S is the number of classes for SST. We set $m_Q = 2$ and $m_S = 5$ for our model. For each task, we use dropout and then a linear prediction layer.

4.4 Gradient Surgery

Yu et al. (2020) hypothesized that one issue with multitask learning optimization is that gradients from different tasks often conflict with one another in ways that harm overall model performance. As depicted in Figure 2, conflicting task gradients, large differences in gradient magnitudes, and high multitask curvature can contribute to optimization difficulty (Yu et al., 2020).

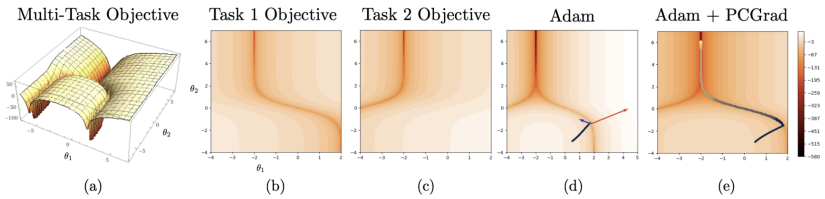

Figure 2: Visualization of PCGrad method. Source: (Yu et al., 2020)

The PCGrad gradient surgery method, as shown in Yu et al. (2020), alters pairs of conflicting task gradients by projecting each gradient onto the normal plane of the other. Their procedure decreases the destructive interference of task gradients with other batch gradients. Denoting the gradient of task $\mathcal{T}_i$ as $\mathbf{g}_i$ and the gradient of task $\mathcal{T}_j$ as $\mathbf{g}_j$, if $\mathbf{g}_i \cdot \mathbf{g}_j < 0$, they compute the following projection:

$$\mathbf{g}_i \leftarrow \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2} \mathbf{g}_j \tag{5}$$

We implement the PCGrad algorithm for multitask learning with the AdamW optimizer following the pseudocode outlined in Yu et al. (2020), with minimal reference to the code implementation presented by Tseng (2020) for debugging. We perform the PCGrad procedure after training on one batch for each task, and then perform a single optimization step on the shared task gradient parameters.

4.5 Cosine Similarity Finetuning

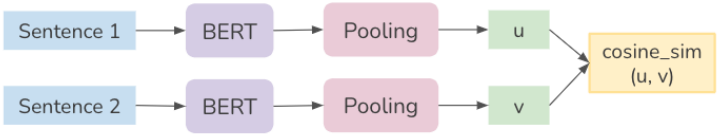

Figure 3: The cosine similarity SBERT architecture for training on the STS task. Adapted from the original visualization in Reimers and Gurevych (2019).

The SBERT model, explained in Reimers and Gurevych (2019), finetunes BERT on the STS task using the regression objective function and a Siamese network structure, as pictured in Figure 3. For a given sentence pair, SBERT passes each individual sentence embedding through the network, obtains each pooler output, and computes the cosine similarity between the two outputs as a prediction. SBERT uses the MEAN pooling strategy, which computes the mean of all BERT output vectors.

We implement the above Siamese network structure with cosine similarity for finetuning on the STS task by following the description of SBERT in Reimers and Gurevych (2019).

4.6 Additional MLM Pretraining

Devlin et al. (2018)'s BERT model is pretrained on the Masked Language Modeling (MLM) task, which contributes to the bidirectionality of BERT embedding representations. In their MLM task, a set percentage (15%) of input tokens are selected for prediction. However, replacing all of these selected tokens with the [MASK] token creates issues during finetuning when the [MASK] token is not present. To mitigate this issue, Devlin et al. (2018) replace 80% of the tokens selected for prediction with a [MASK] token, replace 10% with a random token, and keep 10% as the unchanged token. Each selected token is predicted using its final hidden vector and Cross Entropy Loss.

Gururangan et al. (2020) showed that both in-domain pretraining and additional pretraining on the task's unlabeled data can improve model performance. Krishna et al. (2022) further demonstrated that pretraining directly on the downstream corpora, known as self-pretraining, can improve results.

Accordingly, we implement additional MLM pretraining on the SST, STS, and QQP train datasets, following the procedure outlined in Devlin et al. (2018). We train and optimize each dataset separately within each epoch. We apply dropout after obtaining the pooler outputs for each token, and then use a linear prediction layer.

4.7 Finetuning on SNLI dataset

Bowman et al. (2015)'s Stanford Natural Language Inference (SNLI) dataset contains 570,000 sentence pairs labeled *contradiction*, *neutral*, or *entailment*. Reimers and Gurevych (2019) train SBERT on SNLI using Cross Entropy Loss and a three-way softmax-classifier objective function:

$$o = \text{softmax}(W_t(u, v, |u - v|)) \tag{6}$$

where $W_t \in \mathbb{R}^{3n \times k}$ is a trainable weight for k output labels, u and v are the sentence embeddings in a given sentence pair, and $|u - v|$ refers to the element-wise difference between u and v.

Before finetuning BERT on the downstream tasks, we implement the above procedure to first train on a subset of SNLI using the softmax objective function and Cross Entropy Loss. We perform dropout on the concatenated embeddings and then use a linear layer with 3 output logits for prediction.

5 Experiments

5.1 Data

We use the following datasets:

- CFIMDB Dataset, consisting of 2,434 highly polar movie reviews with binary labels indicating whether the sentiment is positive (Teaching Team, 2023). This dataset is only used in Part 1 of the project to test BERT performance on sentiment analysis.
- Stanford Sentiment Treebank (SST) Dataset, consisting of 11,855 sentences from movie reviews with 5 possible sentiment labels: *negative*, *somewhat negative*, *neutral*, *somewhat positive*, and *positive* (Socher et al., 2013). Sentiment classification is one of the three tasks that the multitask BERT model is expected to perform.
- Quora Question Pairs (QQP) Dataset, consisting of 400,000 question pairs with binary labels indicating whether the sentences are paraphrases of each other (Iyer et al., 2017). This paraphrase task is one of three that the multitask BERT model is expected to perform.
- SemEval STS Dataset, consisting of 8,628 sentence pairs with continuous labels of similarity from 0 (least similar) to 5 (most similar) (Agirre et al., 2013). This similarity evaluation task is one of three that the multitask BERT model is expected to perform.
- Stanford Natural Language Inference (SNLI) Dataset, consisting of 570,000 sentence pairs labeled *contradiction*, *neutral*, and *entailment* (Bowman et al., 2015). This dataset is used to finetune BERT before downstream multitask training.

5.2 Evaluation method

Paraphrase detection on QQP and sentiment classification on SST are both evaluated on **accuracy**, defined as $\frac{\text{\# of correct predictions}}{\text{\# of total predictions}}$. Given the continuity of STS labels, semantic textual similarity is evaluated on **Pearson Correlation**, a value between -1 and 1 which measures the linear correlation between predicted and true labels. To evaluate multitask performance, we average these three scores.

Denoting TP as true positive, FP as false positive, and FN as false negative, secondary metrics that we use to evaluate QQP are **precision** = $\frac{\text{TP}}{\text{TP+FP}}$, **recall** = $\frac{\text{TP}}{\text{TP+FN}}$, and **F1 score** = $2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$.

5.3 Experimental details

For all models, to balance the downstream datasets due to large differences in size, we finetune on the same fixed number of samples n for each train dataset during each epoch and automatically reset a dataset to the first sample when all samples have been trained on. We use the output of the [CLS] token. Unless otherwise specified, these are the default model configurations and hyperparameters:

- **Multitask Finetuning**: 5 epochs, batch size = 8, dropout = 0.3, learning rate (LR) = 1e-5, and $n = 50000$.
- **MLM Pretraining**: 20 epochs, batch size = 8, dropout = 0.3, LR = 1e-5, and $n = 4000$.
- **SNLI Finetuning**: 3 epochs, batch size = 8, dropout = 0.3, LR = 1e-5, and $n = 40000$.
- **Default GPU**: NVIDIA A10G.

Baseline We run the baseline using the pretrained BERT embeddings with a learning rate of 1e-3 and all other default hyperparameters. We freeze all model weights except those in the last prediction layer for each task. The baseline trained for 1 hour 4 minutes on a Tesla P100 GPU.

Multitask Learning We finetune weights using the default configuration for 1 hour 22 minutes.

PCGrad The final PCGrad model was finetuned using default hyperparameters. We experiment with different dropout, learning rates, pooling strategies, and sizes of n (Appendix A.2-A.4), but find that the default hyperparameters perform best. This model trained for 2 hours 41 minutes.

MLM We first train the BERT embeddings on the MLM task using the train sets of QQP, SST, and STS with default MLM hyperparameters. Using these trained embeddings, we then run the MTL procedure as above with the default hyperparameters. The runtime of MLM training is 1 hour 3 minutes and the finetuning runtime is 1 hour 21 minutes.

SNLI We first finetune embeddings on the SNLI dataset for 5 epochs and all other SNLI hyperparameters. Then, we perform the MTL procedure as above with the default hyperparameters. The runtime of SNLI training is 36 minutes and the runtime of MTL training is 1 hour 22 minutes.

Cos Sim + PCGrad We finetune weights for 10 epochs and use all other default hyperparameters, but pass embeddings separately through a Siamese network setup for STS and obtain similarity predictions by computing the cosine similarity between the two sentence outputs. We use the MEAN pooling strategy outlined in Reimers and Gurevych (2019). We use the PCGrad procedure in conjunction. This model ran for 4 hours 40 minutes.

SNLI + PCGrad We train on SNLI, then perform 10 epochs of finetuning with PCGrad and use all other default hyperparameters. The total runtime is 5 hours 32 minutes.

MLM + PCGrad We use MLM, then finetune with PCGrad for 10 epochs with all other default hyperparameters. The runtime is 1 hour 34 minutes for MLM and 5 hours 7 minutes for finetuning.

MLM + SNLI + PCGrad We use all three methods with default hyperparameters. The runtime is 1 hour 4 minutes for MLM, 23 minutes for SNLI, and 2 hours 54 minutes for finetuning.

5.4 Results

The results are presented in Tables 1 and 2. The vanilla MTL model without extensions significantly outperforms the Baseline model, suggesting that multitask finetuning of BERT weights is effective. All models perform relatively similar to each other, although the Cos Sim + PCGrad model attains an average score at least 0.01 points lower than other extension models.

Model	STS Corr.	SST Acc.	QQP Acc.	**Dev** Avg.
Baseline	0.4809	0.4005	0.6811	0.5208
MTL	0.8435	0.5114	0.8564	0.7371
PCGrad	0.852	0.515	0.8538	**0.7403**
MLM	0.8483	0.5114	0.8545	0.7381
SNLI	0.8447	0.4977	0.8582	0.7335
Cos Sim + PCGrad	0.8078	0.5141	0.8502	0.724
SNLI + PCGrad	0.8466	0.5041	**0.8588**	0.7365
MLM + PCGrad	**0.8572**	**0.5204**	0.838	0.7385
MLM + SNLI + PCGrad	0.8493	0.5159	0.855	0.7401

Table 1: Results of models on dev sets.

Model	STS Corr.	SST Acc.	QQP Acc.	**Test** Avg.
PCGrad	0.8517	0.5140	0.8517	0.7392

Table 2: Results of best model on test set.

The PCGrad model attains the highest average score although not on any individual task, followed closely by the MLM + SNLI + PCGrad model. This result is expected since the PCGrad procedure deconflicts task gradients through projection, allowing the model to step in the optimal direction for all three tasks with tradeoffs in individual task performance. In contrast, the Cos Sim + PCGrad model's comparatively lower performance could be because we do not use a linear layer between the BERT embeddings and the cosine similarity computation, making it difficult to finetune the embeddings to produce accurate cosine similarity values while maintaining high multitask performance.

Surprisingly, the MLM task does not significantly improve overall model performance. As seen in Appendix A.1, we were only able to attain accuracies between 23% to 41% on the MLM tasks for the three datasets. Furthermore, increasing the number of epochs for MLM training did not increase the model's average performance, despite MLM accuracies increasing. There are three potential explanations: 1) MLM training on target-domain data does not produce significant performance improvements unless the model achieves significantly higher accuracies on the MLM task. 2) Since the MLM + PCGrad model attains the highest performance on the STS and SST tasks but performs poorly on QQP, MLM may not be as helpful for paraphrase identification. 3) As hypothesized by Zhu et al. (2021), the shallow domain knowledge encoded by additional pretraining only has an effect when not enough data for finetuning is available. That is, the model can better learn task-specific knowledge through direct finetuning instead of additional pretraining.

Finetuning on SNLI also does not improve model performance, despite our model attaining 71.75% accuracy on the SNLI dev dataset. We hypothesize that this is due to a similar reason as with MLM, in that additional finetuning on other datasets is not as useful as encoding knowledge through finetuning on the downstream datasets themselves.

6 Analysis

Sentiment Prediction (SST) We compare PCGrad to the MLM + PCGrad model, which performed best on SST. From Figure 4 and Table 3, we observe that both models have the most trouble with and err away from definitively classifying a sentence as *positive*, *negative*, or *neutral*. One potential reason, when observing the third sentence in Table 3, is that while the model focuses its attention on both positive sentiment words such as "sweet" and "satisfying", this sentiment is negated by words such as "wickedly", leading to a *somewhat positive* prediction. Contrastingly, the MLM + PCGrad model correctly predicts this sentence as *positive*, which may be attributed to the bidirectional contextualization that the MLM task provides. One potential improvement is to further perform the MLM task on additional sentiment datasets to give the model even more context.

Paraphrase Prediction (QQP) From Table 4, we observe that the PCGrad model has a significantly higher recall score than precision score, suggesting that it errs on the side of predicting two sentences as paraphrases. The examples in Table 5 demonstrate that the model tends to classify sentences that contain similar tokens with subject-specific meanings as paraphrases (e.g. "LLB", "BA" in sentence pair 1 or "variable" in sentence pair 2), while classifying sentence pairs with minimal rephrasing as non-paraphrases (e.g. "earlobe" vs "ear" in sentence pair four). The model struggles to understand

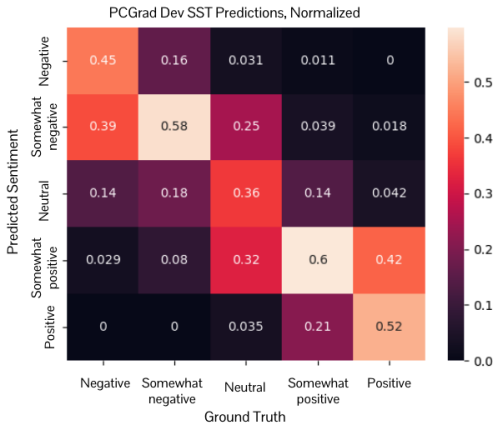

Figure 4: Normalized confusion matrix of PCGrad predictions on the SST Dev set.

Sentence	Predicted Sentiment	Ground Truth
Ruzowitzky has taken this mothball-y stuff and made a rather sturdy, old-fashioned entertainment out of it.	Somewhat positive	Neutral
Stultifyingly, dumbfoundingly, mind-numbingly bad.	Somewhat negative	Negative
Manages to be sweet and wickedly satisfying at the same time.	Somewhat positive	Positive

Table 3: Erroneous predictions of PCGrad model on SST Dev set.

the meaning of domain-specific phrases such as "variable stars" and "BA LLB". Given the extensive pretraining of BERT on massive corpuses, we hypothesize that a lack of domain knowledge is not so much an issue as the forgetting of pretrained knowledge due to aggressive finetuning, and that Jiang et al. (2019)'s smoothness-inducing adversarial regularization and Bregman proximal point optimization methods may combat this overfitting.

Model	Dev QQP Metrics						
	TP	TN	FP	FN	Precision	Recall	F1 Score
PCGrad	6532	10726	1901	1053	0.7746	**0.8612**	**0.8156**
SNLI + PCGrad	6262	11096	1531	1323	**0.8035**	0.8255	0.8144

Table 4: Dev QQP confusion matrix and other evaluation metrics.

Semantic Textual Similarity (STS)

Although we compute the cosine similarity between the two sentence outputs as in SBERT, the Cos Sim + PCGrad model attains lower correlation values than both SBERT and PCGrad. As seen in Figure 5, the Cos Sim + PCGrad model seems to predict negative labels more frequently than PCGrad. This could be attributed to negative cosine similarity being associated with negative correlation between two sentences, and could be remedied by adding a ReLU layer after the cosine similarity computation. Additionally, because we do not add a linear layer between the BERT pooler outputs and the cosine similarity computation, there are no trainable STS prediction weights for the model to finetune outside of the BERT embeddings; in contrast, we use a linear projection layer in the PCGrad model and other extension models. Adding this additional layer to the Cos Sim + PCGrad model could improve STS performance.

In Table 6, we examine the three most erroneous predictions made by the PCGrad model compared to true STS labels, noting the difference in scale. We measure prediction error as the residual, or the vertical distance between the line of best fit and the true y-value. From these examples, we find that the model has trouble with sentence pairs marked as very similar or very dissimilar. Sentence pairs 2 and 3 both feature more complex rephrasings: while "point to lower start" and "signal early losses" have similar meanings in the context of the stock market, the model has difficulty recognizing this contextual meaning. In contrast, the MLM + PCGrad model assigns example 3 a score of 0.4950

Sentence 1	Sentence 2	Prediction	Ground Truth
Can one do an MA in international relations after BA LLB?	Can I do LLB after completing BA?	Paraphrase	Not paraphrase
What is the difference between an engagement ring and a wedding ring?	What is the difference between getting an engagement ring and a wedding ring?	Not paraphrase	Paraphrase
What are variable stars?	What is variable in C?	Paraphrase	Not paraphrase
How do I get rid of a zit on my ear?	How can I get rid of zits on my earlobe?	Not paraphrase	Paraphrase

Table 5: Erroneous predictions of PCGrad model on Dev QQP set.

Sentence 1	Sentence 2	Prediction	Ground Truth
You should do it.	You should prime it first.	0.5551	0.0
It's also a matter of taste.	It's definitely just a matter of preference.	0.3725	5.0
Stock index futures point to lower start	Stock index futures signal early losses	0.2381	4.4

Table 6: Erroneous predictions of PCGrad model on Dev STS set.

with a residual of 1.82, which is lower than PCGrad's residual of 2.75. This may be attributed to the additional contextualization that the MLM model provides on the STS dataset.

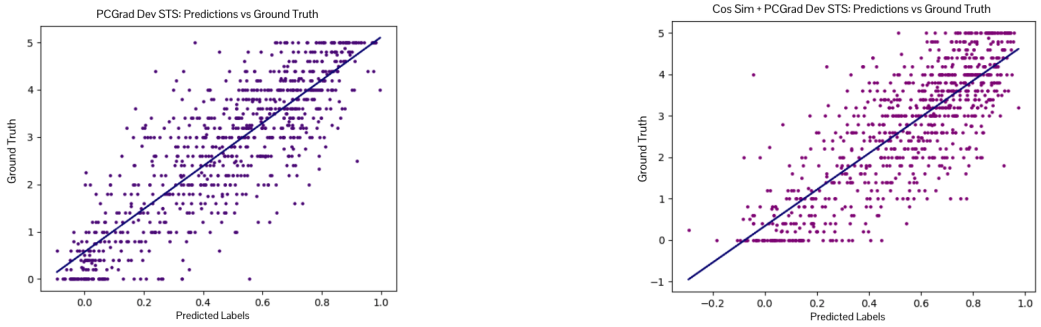

Figure 5: Left - PCGrad STS dev predictions; Right - Cos Sim + PCGrad STS dev predictions.

7 Conclusion

In this project, we finetune a BERT model by using multitask learning on sentiment classification (SST), paraphrase accuracy (QQP), and semantic textual similarity (STS). Although a simple multi-task learning approach already improves the baseline score by over 0.2 points, the gradient surgery optimization procedure yields the most improvement and performs best overall. Masked LM pre-training improves SST and STS scores, but comparatively decreases QQP accuracy. SNLI finetuning does not significantly improve model performance. Given the difficulty of attaining high accuracies on the MLM task with our current approach, further avenues of exploration include performing PCGrad optimization on the MLM task, only performing MLM on the SST and STS datasets, and exploring additional methods to remedy aggressive multitask finetuning. Furthermore, since our MTL finetuning method of balancing datasets leads to long runtimes per epoch, exploration of more efficient sampling methods could improve model efficiency.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *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, Atlanta, Georgia, USA. Association for Computational Linguistics.

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 of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326.

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

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

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *CoRR*, abs/2004.10964.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437.

Kundan Krishna, Saurabh Garg, Jeffrey P. Bigham, and Zachary C. Lipton. 2022. Downstream datasets make surprisingly good pretraining corpora.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. 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, Seattle, Washington, USA. Association for Computational Linguistics.

CS 224N Teaching Team. 2023. Default final project bert handout.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

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

Qi Zhu, Yuxian Gu, Lingxiao Luo, Bing Li, Cheng Li, Wei Peng, Minlie Huang, and Xiaoyan Zhu. 2021. When does further pre-training MLM help? an empirical study on task-oriented dialog pre-training. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 54–61, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Appendix

A.1 MLM Task

For the MLM task, we loop over each dataset separately and optimize each dataset separately within each epoch using Cross Entropy Loss.

Model	STS MLM Acc.	SST MLM Acc.	QQP MLM Acc.	Avg Dev MTL Acc.
MLM, 20 epochs + MTL, 5 epochs	0.2364	0.3074	0.3091	0.7381
MLM, 40 epochs + MTL, 10 epochs	0.3272	0.4062	0.3863	0.7374

A.2 Hyperparameter Search for PCGrad Model

The default hyperparameters are: LR = 1e-5, epochs = 5, dropout = 0.3, and batch size = 8. We change one hyperparameter at a time.

Model	STS Corr.	SST Acc.	QQP Acc.	Avg Dev Acc.
PCGrad (default)	0.852	**0.515**	0.8538	**0.7403**
PCGrad, LR=2e-5	**0.8557**	0.4959	**0.8596**	0.7371
PCGrad, LR=3e-5	0.8504	0.4977	0.8549	0.7343
PCGrad, dropout=0.2	0.8499	0.5032	0.8561	0.7364
PCGrad, dropout=0.5	0.8505	0.5032	0.859	0.7376

A.3 Pooling Strategy

Model	STS Corr.	SST Acc.	QQP Acc.	Avg Dev Acc.
PCGrad, CLS (default)	**0.852**	0.515	**0.8538**	**0.7403**
PCGrad, MEAN	0.8485	**0.5295**	0.8423	0.7401

A.4 Number of Samples per Epoch

n denotes the number of samples that we train on for each dataset in multitask learning. To balance datasets, we reset a dataset back to its first sample if all samples have been trained on. We repeat this process until n samples in the dataset have been trained.

Model	STS Corr.	SST Acc.	QQP Acc.	Avg Dev Acc.
PCGrad, n=30000	0.8441	0.5141	0.8501	0.7361
PCGrad, n=40000	0.8508	0.5123	0.8492	0.7374
PCGrad, n=50000 (default)	**0.852**	**0.515**	**0.8538**	**0.7403**

A.5 STS Spearman Correlation

For completeness, we use **Spearman Correlation**, another measure of correlation mainly used when the data being compared has a non-linear relationship, to evaluate STS, as this metric was used to evaluate SBERT (Reimers and Gurevych, 2019).

Model	Dev STS Spearman Correlation
PCGrad	83.37
Cos Sim, PCGrad	78.45
SBERT*	**84.67**