Efficient Finetuning for Multi-tasking minBERT

Stanford CS224N Default Project

Tz-Wei Mo
Department of Electrical Engineering
Stanford University
tzwmo@stanford.edu

Annie Ho
Department of Electrical Engineering
Stanford University
annieho@stanford.edu

Abstract

Our goal for the project is to 1) maximize performance of the minBERT model for all three downstream tasks using multi-task training and 2) try to find efficient methods to speed up training time and reduce the number of trainable parameters without sacrificing the final performance of the model. To improve multi-task training performance, we came up with our own version of BERT model pre-training which includes masked language modeling (MLM) while simultaneously training on the SNLI (Williams et al., 2018) and Multi-NLI (Bowman et al., 2015) classification task. To prevent over-fitting when fine-tuning on downstream tasks, we applied an adversarial regularization from Jiang et al. (2019). We also applied gradient surgery (Yu et al., 2020) to resolve gradient conflicts and to increase training efficiency with gradient surgery, we added LoRA (Hu et al., 2022) to decrease the number of trainable parameters . Limited by the computational resources and time constraints, we were not able to conduct rigorous experiments and ablation studies for all methods applied, but we did observe an increase in performance quantitatively after applying further pre-training and adding adversarial regularization.

1 Key Information to include

- Mentor:
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

The objective of the default project is to extend the BERT-base-uncased model from Devlin et al. (2018) to generate competent word embeddings that are useful to work with for the three subsequent downstream tasks: 1) sentiment analysis (SST dataset), 2) paraphrase detection (Quora dataset), and 3) semantic textual similarity (STS dataset). Directly applying embeddings from the pre-trained BERT-base-uncased model for all three tasks would result in suboptimal outcome as BERT is trained on a general language modeling task with a larger dataset, which has a different distribution than our current tasks. This project is therefore quite challenging as we are being limited to using BERT to generate the word embeddings, but are free to design our own classification strategies, end-to-end training procedures, and incorporate additional datasets to get more robust embeddings. Here we first list an overview to the approaches we took for this project.

One approach we took is to perform further pre-training on BERT before fine-tuning on the downstream tasks. The motivation behind this is to produce better initial embeddings by performing masked language modeling (MLM) training, since in class, we have seen how an unsupervised language modeling task allows the model to learn the semantic relationships between words and phrases and enables the model to understand the context in which words are used. From Reimers and Gurevych (2019), we also learned that training on NLI data could boost performance for the STS

dataset, therefore in addition to doing MLM training on the SST, Quora, and STS datasets, we also included the SNLI and Multi-NLI datasets for an additional training objective.

While the further pre-trained model gave us better starting embeddings, we soon discovered that fine-tuning on the smaller task datasets often lead to over-fitting that hurt evaluation accuracies. To combat this we added adversarial regularization studied in Jiang et al. (2019), which adds an additional adversarial loss to enforce smoothness of the objective function under the given model parameters. We also added warm-up scheduling for the learning rate to prevent aggressive updates at the start of training.

The above methods have been reported to work well when fine-tuning on single tasks, but for multi-task learning, gradient descent is performed on all three tasks simultaneously, which could introduce conflicting gradients and lead to inefficient training process. We mitigate this effect by applying gradient surgery described in (Yu et al., 2020) where conflicting gradients are projected onto each other and altered to resolve the conflict. Limited by GPU memory, we were only able to apply gradient surgery to LoRA where the reduced number of trainable parameters allows for a more compact gradient graph.

To improve memory efficiency and training speed, we applied low rank adaptations described in Hu et al. (2022), where it is proposed that applying low rank updates on linear layers in the network may be sufficient for fine-tuning on NLP tasks. Our empirical results indicate that the utilization of LoRA led to a decline in performance, which was somewhat disappointing. Nevertheless, we deem it worthwhile to highlight the time-to-performance trade-off in our report.

By experimenting and combining these methods, we are able to get much better performances compared to directly fine-tuning the BERT-base-uncased model on the downstream tasks as can been seen in the experiment results.

3 Related Work

3.1 Language Model Pre-training

The BERT model is pre-trained for general domain usage. Further pre-training on a specific task domain can often improve the performance significantly. This is typically achieved through pre-training with language model objective and next sentence prediction. Sun et al. (2019) investigated on pre-training with various data types and domains, including 1) within-task pre-training, 2) In domain pre-training, and 3) cross-domain pre-training. Within-task pre-training involves pre-trained BERT on the target task objective using the training data. In-domain pre-training combines additional data from the same task domain and trains it with the target task. Cross-domain combines both same domain and different domain data for pre-training. Through experiments, Sun et al. (2019) found that in-domain outperformed within-task pre-training, while cross-domain pre-training did not show a significant improvement.

3.2 Smoothness-Inducing Adversarial Regularization

Due to the limited data resources in downstream tasks and the complexity of models, fine-tuning can suffer from aggressive overfitting. To address this, smoothness-inducing regularization was proposed by Jiang et al. (2019). This regularization technique measures local Lipschitz continuity and encourages smoothness in the model function by adding a regularization term to the objective. This regularization enforces the model output to remain unchanged while perturbing the input, thereby alleviating the overfitting issue and leading to a more robust model.

3.3 Gradient Surgery for multi-task learning

Optimizing multitask models has always been a challenge, and despite many studies in this field, the reasons why it is difficult are still not fully understood. Yu et al. (2020) hypothesize that one reason could be gradient conflict, which would be detrimental when it occurs with positive curvature and a large difference in gradient magnitude. To address this conflict, they proposed a gradient surgery method called PCGrad that projects one task's gradient onto the normal plane of another conflicting gradient. By repeating this process, all conflicts can be resolved. This not only alleviates challenges for optimization but also could lead to an improvement in performance for multi-task training.

3.4 Low-Rank Adaptation of Large Language Models

Due to the increasing number of parameters of language models, it becomes more expensive and impractical to fine-tune the full models. Moreover, some previous works for fine-tuning have introduced inference latency. Hu et al. (2022) proposed a solution called Low-Rank Adaptation (LoRA) to address these issues. LoRA introduces trainable rank decomposition matrices, which is called LoRA modules, into the dense layers while freezing the pre-trained weights. LoRA is designed with a very low rank, making fine-tuning both storage- and computation-efficient, and allowing the model to switch between different tasks by simply replacing the LoRA modules. Additionally, this approach introduces no inference latency since the matrices can be merged into the frozen pre-trained weight due to their linear characteristic.

4 Approach

4.1 Further Pre-training for BERT

As mentioned in previous sections, we devised our own pre-training process by adding additional datasets and an additional training objective. For masked language modeling, we included all training data from SST, Quora, STS, SNLI, and Multi-NLI datasets. We followed the same data processing procedure in Devlin et al. (2018), where 15% of the tokens were masked and 20% of the masked tokens were replaced by random tokens. In addition to MLM training, we also performed the Recognizing Textual Entailment (RTE) task of determining the inference relation between two texts using the labeled SNLI and Multi-NLI datasets consisting of the three classes: entailment, contradiction, and neutral. The loss function for the entire pre-training is then:

$$\mathcal{L}_{pretrain} = w_1 \mathcal{L}_{MLM} + w_2 \mathcal{L}_{RTE}, \tag{1}$$

where $\mathcal{L}_{MLM}$ is the cross entropy loss corresponding to all words in the dictionary and $\mathcal{L}_{RTE}$ is the cross entropy loss for predicting the three classes. We empirically set $w_1 = 0.7$, $w_2 = 0.3$ and trained for 5 epochs with a learning rate of 5×10^{-6} on the combined datasets.

4.2 Multi-task Fine-tuning for minBERT

For our project, instead of fine-tuning the BERT model on individual tasks, we used multi-task learning to train all downstream task jointly with the following loss function:

$$\mathcal{L}_{total} = \alpha_{sst} \mathcal{L}_{sst} + \alpha_{para} \mathcal{L}_{para} + \alpha_{sts} \mathcal{L}_{sts}, \tag{2}$$

where $\mathcal{L}_{sst}$, $\mathcal{L}_{para}$, and $\mathcal{L}_{sts}$ are the losses for the three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity, respectively and α_{sst}, α_{para}, α_{sts} are hyperparameters which are linear weights for each loss.

For sentiment analysis, the input to the task head is simply the pooler output from the BERT model, and the task head is simply a dropout layer followed by a linear layer.The loss for this task is the cross entropy loss between the predicted output logits from the linear layer and the true label, which is 1 of 5 classes

For paraphrase detection, we pass each sentence in a sentence pair through the pre-trained BERT model to get two pooler output embeddings. Then we concatenate the two embeddings into a single vector and pass the joint representation through a dropout layer followed by a linear layer. The loss for this task is the binary cross entropy between the output logits and the label.

For semantic textual similarity, instead of taking the pooler output, we use the entire last hidden state output of BERT and used mean pooling along with the attention mask to get a pooled representation of each sentence. We then performed cosine similarity on the pair of embeddings to get a final output score. Both the output score and labels are then linearly scaled to the range 0 to 1 and a mean squared error (MSE) loss is calculated for the scaled score and label.

4.3 Adversarial Regularization

To allow better generalization to unseen data during evaluation, we add a regularization loss to the loss function of a task, such that given a model $f(\cdot; \theta)$ with parameters θ , n data points $\{(x_i, y_i)\}_{i=1}^{n}$,

and a loss function $\ell(f(x_i;\theta), y_i)$, we instead optimize the following equation:

$$\text{minimize}_\theta \mathcal{F}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i) + \mu\mathcal{R}(\theta) \tag{3}$$

where μ is a positive scalar, and $\mathcal{R}$ is the regularization loss. The regularization loss is defined as:

$$\mathcal{R}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_\infty \leq \epsilon} \ell_s(f(\tilde{x}_i;\theta), f(x_i;\theta)), \tag{4}$$

where $\epsilon > 0$ is a tuning parameter and ℓ_s is chosen as the symmetrized KL-divergence for classification tasks and squared loss for regression tasks. This essentially means that the model $f(\cdot;\theta)$ should output similar values for nearby points within a neighborhood of the infinity-norm ball, forcing the model to produce smoother decision boundaries. Since finding the maximum value difference within the norm ball requires exhaustive search, instead we approximate the regularization loss in equation (4) by first adding random normal noise $\mathcal{N}(0, \sigma^2)$ to each x_i, then perform one gradient ascent step and project back onto the infinity-norm ball to get an approximate $\tilde{x}_i$.

4.4 Gradient Surgery

Since we are training parameters with three tasks, we may expect to find gradient conflicting while training. Therefore, we tried adopting PCgrad for gradient surgery to modify the conflicting gradients. If gradient g_i and gradient g_j conflict each other, we will replace g_i by projecting it to the normal plane of g_j:

$$g_i = g_i - \frac{g_i \cdot g_j}{\| g_j \|_2} g_j \tag{5}$$

4.5 Low Rank Adaption for BERT

To allow memory efficient finetuning with gradient surgery, we applied Low Rank Adaption (LoRA) updates to BERT. For a general weight matrix $W \in \mathbb{R}^{d\times k}$ in a linear layer of the network, we want to represent the parameter update ΔW_0 with a low rank decomposition A and B such as:

$$W + \Delta W_0 = W + AB, \tag{6}$$

where $\Delta W_0 \in \mathbb{R}^{d\times k}$, $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$, and $r << min(d, k)$. This way, the number of parameters in matrix A and B would be much smaller than ΔW_0. With this approach we can freeze the pre-trained BERT weights and only train on the LoRA parameters, which allows us to combine this method with gradient surgery as computing gradient graphs for the entire BERT model can be too expensive.

5 Experiments

5.1 Data

Default Dataset We use provided datasets for fine-tuning. For the sentiment analysis task, we use Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013). For the paraphrase detection (para) tast, we use the Quora question pairs dataset (Quo), and finally, for the semantic textual similarity (sts) task, we use the SemEval STS Benchmark dataset (Agirre et al., 2013).

Additional Dataset To further improve on the performance, we conducted a in-domain pre-training prior to fine-tuning on the target tasks. We use the Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015) and the Multi-Genre Natural Language Inference (Multi-NLI) corpus (Williams et al., 2018).

5.2 Evaluation method

We model sentiment analysis and paraphrase detection tasks as classification problems, and evaluate their performance using the default accuracy metric as specified in the handout. For the semantic textual similarity task, we calculate the similarity between two sentences and evaluate it using the Pearson correlation coefficient.

5.3 Experimental details

For the final submission training, we first pre-train the BERT model as a separate multi-task model. The first task involves training the model on a language model objective using both provided datasets (SST, Quora, STS) and our additional datasets (SNLI, multi-NLI). For the second task, we train the model on a recognizing textual entailment task using the SNLI and multi-NLI datasets. We set task weights $w_1 = 0.7, w_2 = 0.3$ and learning rate 5×10^{-6}. The training time on a NVIDIA GPU-based Amazon EC2 G5 instance (g5.2xlarge) is around 26.5 hours for 5 epochs.

Then, we use the pre-trained BERT weight to fine-tune on our target tasks. The task weights for three tasks are $\alpha_{sst} = 1, \alpha_{para} = 1, \alpha_{sts} = 1$, and the scalar for regularization loss is set to $\mu_{sts} = 10$. The learning rate is set separately for BERT and the task heads, while $lr_{BERT} = 1 \times 10^{-7}$ and $lr_{heads} = 1 \times 10^{-5}$. The batch size is set to 16, and $p_{dropout} = 0.3$ The training time on our instance is about 32 hours.

In table 1, we list some other experiments we have done:

- **Finetune** We directly fine-tune the model with all the tasks without any pre-training.
- **Finetune-LM-pretrain** In this experiment, we use masked language modeling training on the given three datasets (SST, Quora, STS).
- **Finetune-NLI-LM-pretrain** Besides the masked language modeling training on SST, Quora, and STS, we also include SNLI and Multi-NLI. Additionally, we pre-train the SNLI and Multi-NLI datasets with the semantic textual similarity task, which is a three-class classification task.
- **Finetune-NLI-LM-pretrain (Adv + LoRA + PCGrad)**In this experiment, we pre-train BERT as same as in the above experiment. Then, we freeze BERT, add trainable 2-rank LoRA modules to all linear layers, and fine-tune both LoRA modules and task heads. Also, to prevent gradient conflict, we use PCGrad for the fine-tuning. Note that, though fine-tuning on LoRA can speed up the training process by reducing the size fo the gradient graph, using PCGrad to perform calculation on gradients is time-consuming, which costs about 1 hour for 1 epoch totally. We add the adversarial loss only for STS tasks, since we observed that this task is most prone to overfitting in our setup.
- **Finetune-NLI-LM-pretrain (Adv)** We use the same pre-trained BERT from the above experiment and fine-tune on both BERT and task heads. This is the setting for our final submission.

Experiments	SST	Para	STS	Avg.	Training Time
Finetune	48.14	78.31	27.88	51.43	32 mins
Finetune-LM-pretrain	47.77	78.76	62.40	62.96	32 mins
Finetune-NLI-LM-pretrain	51.68	72.93	75.48	66.66	32 mins
Finetune-NLI-LM-pretrain (Adv + LoRA + PCGrad)	50.59	71.05	75.55	65.73	61 mins
Finetune-NLI-LM-pretrain (Adv)	**52.86**	**73.79**	**78.17**	**68.27**	**67 mins**

Table 1: This table shows the accuracy and correlation scores on dev datasets with different experiment settings. See 5.3 for the details.

5.4 Results

The below table 2 shows the scores we obtained from the dec/test set leaderboard. We find the results from dev set to be very similar to the test set. We believe that adding regularization helped with generalization, assuming that the dev and test sets have data distributions similar to the training set.

6 Analysis

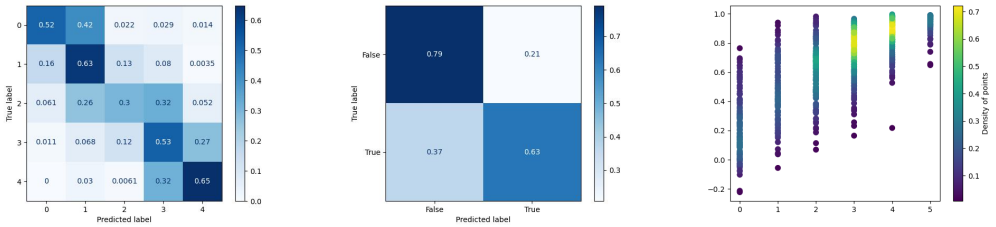
- **Sentiment Analysis**
  In Figure 1a we printed out the confusion matrix for the dev set for all 5 classes for SST. When the true label is class 0, our model almost has a 50-50 chance of guessing either class 0 or class 1, which means it is learning to detect negative sentiments, but is having a hard

Scores	Dev	Test
SST Accuracy	0.529	0.543
Paraphrase Accuracy	0.738	0.737
STS Correlation	0.782	0.758
Overall score	0.683	0.679

Table 2: Scores from leaderboard

time distinguishing between the worst comments from the slightly worse ones. Our model also performs relatively better for class 1 and class 4, and for class 2, our model again has a harder time distinguishing the classes of neutral comments and guesses evenly between classes 1 through 3. To see a possible reason for our model's behavior, we counted the number of labels for 5 classes as shown in Figure 2a and found that there are less data for class 0 which might explain why our model has a harder time learning to predict the worst comments. There are also less labels for class 2 and more labels for classes 1 and 3 which may also explain why our model is having difficulty to get good predictions for class 2.

- **Paraphrase Detection** For paraphrase detection, we printed out the confusion matrix in Figure 1b to analyze our model's performance. Here we see that our model tends to get more false negatives where the true label is true, but our model predicts false. Again after looking at the number of labels in the training dataset shown in Figure 2b, we suspect that having more negative true labels in the dataset may have impacted our model to predict false more often.
- **Semantic Textual Similarity**: We drew a scatter plot to visualize the predictions and labels for the semantic textual similarity task, as shown in Figure 1c. The y-axis represents the cosine similarity of the embeddings, and x-axis represents the ground truth label. We did not normalize the cosine similarity score, but in the figure, we can see that most of the outputs fall in the range $[-0.2, 1]$, with only a few negative scores, which is only half the range of $[-1, 1]$. We suspect that this is caused by the final layer normalization before outputting the embeddings in BERT. When calculating the regression loss, we mapped the output similarity from the range $[-1, 1]$ to the label range $[0, 1]$. However, since our similarity scores do not vary that widely , this can lead to a worse regression result. Therefore, we expect that we might need some post-processing before feeding the embeddings to the similarity loss function in order to enhance the ability to distinguish the similarity of sentence pairs .

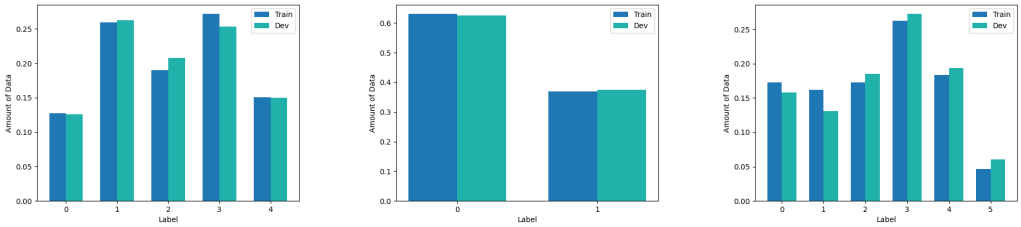


(a) Confusion Matrix for SST (b) Confusion Matrix for Quora (c) Scatter Plot for STS

Figure 1: Evaluation on three tasks

7 Conclusion

In our project, we learned that doing pre-training with another related task (RTE) helped with boosting the performance the most and adversarial regularization helped generalize the model better to unseen data. The primary limitation to our project was insufficient memory to perform PCGrad when fine-tuning the whole BERT model and not enough computation resources to try out and tune hyperparameters as we discovered learning rate and weights for different tasks impacted the model's performance greatly. We also analyzed our model's performance and hypothesized that the different number of labels in the training set may have impacted our model's behavior.



(a) SST (b) Quora (c) STS

Figure 2: Label Distribution of three datasets

References

First quora dataset release: Question pairs. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

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.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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.

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 Y 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.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.