Building Robust Adaptation for Multi-Task Learning over minBERT

Stanford CS224N Default Project

Guhui Zhang
Department of Statistics
Stanford University
ghzhang@stanford.edu

Jialing Pan
Department of Statistics
Stanford University
jlpan@stanford.edu

Xinyun Tao
Department of Statistics
Stanford University
ttao23@stanford.edu

Abstract

Our motivation is to address the limitation of Projected Attention Layers (PALs). Our goal is to improve the performance of PALs with minBERT as the baseline model by experimenting with training methods, adding regularization and hyperparameter optimization. During on our experiments, we saw an increase in accuracy and correlation with both PALs and low rank layers. We found that using weight decay and annealed sampling could ameliorate overfitting induced by increased model complexity, but more approaches could be explored to further improve model performance.

1 Key Information to include

- TA Mentor: Drew Kaul

2 Introduction

Improving the performance of sentence-level tasks is an essential part of Natural Language Processing. Such tasks, such as sentiment analysis, are widely deployed in the industry due to their applicability in various scenarios. For example, companies would like to predict the users' sentiment based on their reviews so that they can learn from the reviews and produce better products. They may also wish to analyze the semantic similarity between reviews to detect bots. Therefore, we need a robust model to perform task predictions accurately and in a timely manner.

Our approach to performing sentence-level tasks simultaneously is to use multi-task learning, which involves the development of training models based on a pre-trained model named Bidirectional Encoder Representations from Transformers (BERT). Previous works on multi-task learning include adaption parameters, fine-tuning approaches, and adapting self-attention.

In this project, we want to improve the multi-task learning performance by drawing inspiration on previous works and building a robust adaptation for multi-task learning over minBERT on 3 NLU (Natural Language Understanding) tasks. Specifically, we introduce the Project Attention Layer (PAL) with parameter sharing which learns task-specific parameters; we deploy different training methods and use regularizations to reduce overfitting.

3 Related Work

Stickland and Murray [4] present a more efficient and effective approach to multi-task learning by sharing most parameters and incorporating a smaller number of task-specific parameters. Their contributions include task-specific functions $TS(\cdot)$ and scheduling training. In the BERT model, a task-specific function is added in each BERT layer. The task-specific function could either be a "Low-rank Layer", containing an identity function resulting in a low-rank linear transformation, or a "Projected Attention Layer" (PAL), containing a multi-head attention with shared parameters across layers. In addition, the paper introduces a new method for schedule training using "annealed sampling".

In order to simultaneously perform three natural language tasks with the BERT embeddings, we need to find methods that can build more robust embeddings that generalize well to different language tasks. Among many valid options, our team is interested in exploring how to modify the model architecture of minBERT to adapt to multi-task learning. Specifically, we are aware of the potential increase in model parameters if we fine-tune a separate model for each task, and want to explore methods that can balance sharing some parameters across the tasks and adding task-specific units.

Therefore, we chose to base our research on this paper since it proposed the “Projected Attention Layer” (PAL) that could fulfill our goal. By adding a task-specific function in parallel with each self-attention layer, this method only requires 1.13x original parameters to match the state-of-the-art performance for different GLUE tasks. Also, since the paper compares different choices of the function $g(\cdot)$ within the adaptation layer, we believe that following the paper’s approach would grant us the freedom to design and experiment with new ways of transformations. In addition, the model has its limitation: the model does not use various training methods or any methods to limit the interference from training on separate tasks, so we do not know the performance after adapting different training methods on PAL and BERT. Then, we would explore and evaluate the model performance by adapting different training methods on PAL and BERT.

4 Approach

Our approach to building this adaptation for multi-task learning over minBERT is that we first use minBERT as the baseline model and experiment with two extensions, PAL and Lowrank. Then, we utilized the pretrained minBERT model weights and fed the outputted embeddings to three prediction heads to perform sentiment analysis, paraphrase detection, and semantic textual similarity on the Stanford Sentiment Treebank dataset (SST), Quora dataset, and SemEval Benchmark Dataset (STS). To perform these tasks, we trained the model on all datasets and further develop three training methods as well as regularization.

4.1 Baseline: minBERT

We implemented minBERT [2] as our baseline model, a classifier that utilizes the output of minBERT on Sentiment Analysis with Adam Optimizer as the Stochastic Optimization method, and a module that performs multitask predictions.

To tokenize input sentences, we used a ‘WordPiece‘ tokenizer, which split sentences into word pieces, and then we converted the word pieces into ids. Following tokenization, we applied a trainable embedding layer to each token. The minBERT model then utilized 12 Encoder Transformer layers, consisting of multi-head attention, followed by an additive and normalization layer with a residual connection, a feed-forward layer, and a final additive and normalization layer with a residual connection. The minBERT model outputted the BERT encoder as well as the [CLS] token embedding.

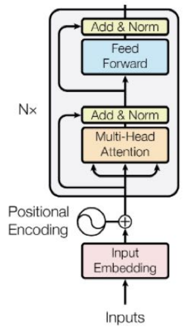

Figure 1: Encoder Layer of Transformer used in BERT. Figure from [3]

4.2 Prediction heads and loss functions

We passed the output embeddings of minBERT to different prediction heads for each task. Based on the prediction heads and evaluation metrics, we chose three different loss functions

For sentiment classification, the embeddings went through a dropout layer and a linear layer. The linear layer has an output size equal to the number of classes (0 to 4). The class with the maximum output value (logit) is predicted to be the sentiment class. Since this is a classic classification setting, we used cross entropy (CE) as the loss.

For paraphrase detection, the embeddings went through a dropout layer and a linear layer. Then we computed the dot products between each pair of sentences and applied a sigmoid function to produce the probability that the sentence pair is a paraphrase of each other. Under such a binary classification scenario, we chose binary cross entropy (BCE) with logits as the loss.

For semantic similarity prediction, we used the same treatment for the embeddings as for paraphrase detection: a dropout layer, a linear layer and the computation of dot products for each sentence pair. No sigmoid function is applied as the prediction is not binary. Considering that the evaluation metric is Pearson's correlation, we selected mean squared error (MSE) as the loss.

4.3 Task-specific Parameters: Low-rank Layers and Projected Attention Layers

We implemented a task-specific transformation of the hidden states on the minBERT architecture by adding a task-specific function $TS(\cdot)$ in parallel to the minBERT layers. In each layer l, the hidden vector for a particular sequence element $\mathbf{h}$ is updated as $\mathbf{h}^{(l+1)} = LN(\mathbf{h}^{(l)} + SA(\mathbf{h}^{(l)} + TS(\mathbf{h}^{(l)}))$, where $LN(\cdot)$ is layer normalization, $SA(\cdot)$ is self-attention and $TS(\cdot)$ is a task-specific function of the form $TS(\mathbf{h}) = V^D g(V^E \mathbf{h})$ with V^E and V^D as encoder and decoder matrices. Taking $g(\cdot)$ as the identity function gives **Low-rank Layers**; taking $g(\cdot)$ as multi-head attention with shared V^E and V^D across layers gives **PAL**s. We implemented and integrated Low-rank and PAL layers to each layer in the minBERT model.

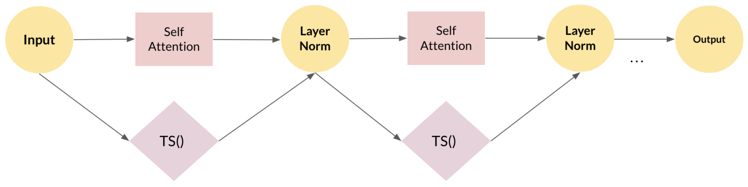

Figure 2: Schematic diagram with task-specific function

4.4 Training Methods

We experimented with the following different training methods to effectively adapt the model parameters to multi-task learning.

- **Sequential Training** For every epoch, we trained tasks sequentially. To be specific, we go through the batches of one dataset before proceeding to another.
- **Additive Loss** The purpose of using additive loss [1] is similar to sequential training. Specifically, we interweave the batches of three datasets and sum their losses using $L_{total} = L_{task1} + L_{task2} + L_{task3}$.
- **Gradient Surgery** Gradient directions of different tasks might result in conflicting gradient. Gradient Surgery [5] can solve this issue by projecting the tasks's gradient of the i-th task g_i onto the normal plane of any other task that has a conflicting gradient for each task. This process can be written as: $g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} \cdot g_j$.
- **Annealed Sampling** The BERT&PALs paper [4] introduces a new method for scheduling training using "annealed sampling" to deal with differences in dataset sizes for multi-task training. Data examples are sampled with probability p_i proportional to their training set

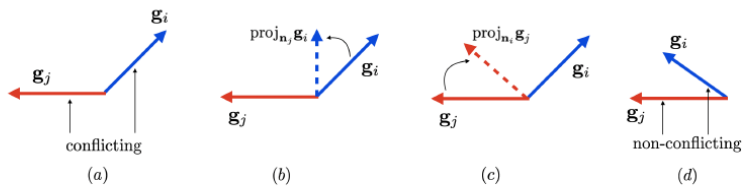

Figure 3: Gradient surgery

size N_i at first, and the influence of N_i is de-emphasized as training proceeds. This is formulated as $p_i \propto N_i^\alpha$ and $\alpha = 1 - 0.8\frac{e-1}{E-1}$, where E is the total number of epochs. The method trains on tasks more equally towards the end of the training phase and guards against interference. We implemented sequential training, additive loss, and gradient surgery as three separate training methods.

4.5 Optimizer: AdamW + regularization

We implemented AdamW as the optimizer using weight decay as a form of regularization. Weight decay is a technique used to prevent overfitting in the model by incorporating a weight decay update.

5 Experiments

5.1 Data

We will use the following provided datasets to test our model on three natural language tasks.

1. **Stanford Sentiment Treebank (SST) dataset**: This dataset consists of 11,855 single sentences from movie reviews extracted from movie reviews and includes a total of 215,154 unique phrases. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive.
2. **CFIMDB dataset**: This dataset consists of 2,434 highly polar movie reviews. In this project, we will predict each movie review to be negative or positive.
3. **Quora Dataset**: This dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. We will be given a subset of this dataset and use it to perform the task of paraphrase detection.
4. **SemEval STS Benchmark Dataset**: This dataset consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). We will use this dataset to perform the task of semantic textual similarity.

Since all datasets are included in the provided CS224N guideline, we believe that we can have easy access to the resources.

5.2 Evaluation method

To evaluate the model's multi-task performance on sentiment analysis, paraphrase detection, and semantic textual similarity, we fine-tuned the embeddings and evaluated the model's performance on three datasets.

During training, we reported the average training loss of three tasks for every epoch. According to the nature of tasks, we chose to use cross entropy loss for sentiment analysis, binary cross entropy loss for paraphrase detection, and mean squared error (MSE) loss for semantic textual similarity as it is a regression problem in nature.

To evaluate the model's performance, we reported the sentiment classification accuracy on the SST development dataset. For paraphrase detection, we predicted binary labels and reported the classification accuracy of labels on the Quora development dataset. For semantic textual similarity, we calculated the Pearson correlation of the true similarity values against the predicted similarity values across the SemEval STS development dataset.

5.3 Experimental details and analysis

Our baseline is a minBERT model trained on the task of sentiment classification using the SST and the CFIMDB dataset. Without the finetuning, the model achieved a classification accuracy of 40.3% on the SST development dataset and a classification accuracy of 76.3% on the CFIMDB development dataset. With fine-tuning, the classification accuracy increased to 51.1% on the SST development dataset, and increased to 97.1% on the CFIMDB development dataset.

To establish a robust adaptation of minBERT to multitask learning, we adopted the architecture of adding task-specific parameters to each shared BERT layer. As described in the previous section, we experimented with two choices of task-specific functions $TS(\cdot)$: low-rank layers with the identity function and projected attention layers (PALs). We also experimented with different training methods by sequentially training three tasks, adding the losses of three tasks, gradient surgery, and annealed sampling method. Finally, we implemented weight decay on the AdamW optimizer as a regularization method. The experimentation details are described below.

5.3.1 Experiment 1: Sequential and Additive Loss

Description. We trained the baseline minBERT model with both additive loss and sequential training. For both experiments, we used the default parameters of a learning rate of 1e-5, a batch size of 8, and a hidden dropout probability of 0.3.

Results. Sequential training achieved a development classification accuracy of 46.7% on the SST dataset, a development accuracy of 76.0% on the Quora dataset, and a development Pearson correlation coefficient of 0.519 on the STS dataset. Training with additive loss achieved a development accuracy of 49.5% on the SST dataset, a development classification accuracy of 72.6% on the Quora dataset, and a development Pearson correlation coefficient of 0.423 on the STS dataset.

Analysis. Despite the training efficiency of additive loss, sequential training achieved a better performance on two of the three tasks, as well as a higher average development accuracy. We hypothesize that when we add the losses from three tasks together in a training step, the losses are not on the same scale and their gradients could interfere with each other when pointing in conflicting directions. This could lead to degraded performance in the multi-task learning setting. We then experimented with gradient surgery to alleviate the issue of conflicting gradients when simultaneously training three tasks in one step.

5.3.2 Experiment 2: PALs with Sequential Training

Description. We implemented the projected attention layer (PALs) to add a task-specific layer to each BERT layer. The projection matrices in the task-specific layer are shared across layers for each task. The $g(\cdot)$ function is chosen to be the multi-head self-attention layer. We used the default parameters of a learning rate of 1e-5, a batch size of 8, and a hidden dropout probability of 0.3.

Results. After implementing PALs and training the model with the sequential training method, we achieved a development classification accuracy of 48.9% on the SST dataset, a development accuracy of 79.0% on the Quora dataset, and a development Pearson correlation coefficient of 0.499 on the STS dataset.

Analysis. We hypothesized that PALs should improve the performance of multi-task learning by adapting task-specific parameters in each BERT layer. Our results mostly aligned with our hypothesis that PALs with sequential training improved the classification accuracy of both sentiment analysis and paraphrase detection, compared to the baseline model. However, the effect is not significant for the semantic textual similarity task. We further observed that this configuration led to significant overfitting. The training loss steadily decreased and the training performances improved significantly. In particular, the model achieved > 0.9 on the training metrics for all three tasks (Table 1), but the dev performance reached a plateau after the first few epochs (Figure 4).

5.3.3 Experiment 3: Low Rank with Sequential Training

Description. We implemented the low rank layers, which simply use the identify function as the $g(\cdot)$ function in the task-specific layer. The projection matrices in the task-specific layer are not shared

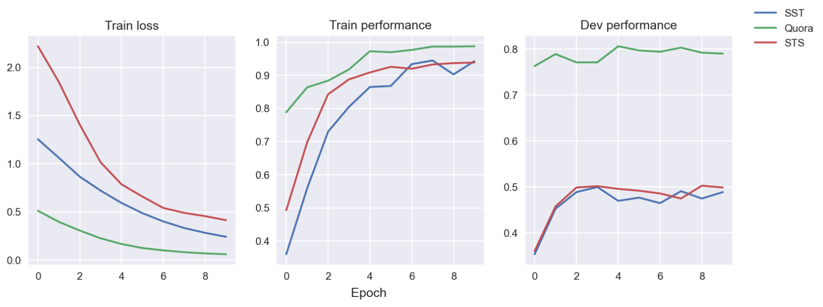

Figure 4: PALs + Sequential model performance

across layers for each task. We used the default parameters of a learning rate of 1e-5, a batch size of 8, and a hidden dropout probability of 0.3.

Results. After implementing low rank layers and training the model with the sequential training method, we achieved a development classification accuracy of 50.6% on the SST dataset, a development accuracy of 80.1% on the Quora dataset, and a development Pearson correlation coefficient of 0.481 on the STS dataset.

Analysis. Although the $g(\cdot)$ function is chosen to be the identity function, the projection matrices are not shared in this setting, allowing the model to learn more robust adaptation to specific tasks. Therefore, we hypothesized that low rank layers should achieve similar performance to PALs. Our results mostly aligned with our hypothesis that low rank layers with sequential training furthered improved the classification accuracy of both sentiment analysis and paraphrase detection, compared to the baseline model. However, the effect is worse for the semantic textual similarity task than for PALs. In addition, the model faced a similar overfitting problem as PALs.

5.3.4 Experiment 4: PALs with Gradient Surgery

Description. We implemented gradient surgery, which alters the gradients of tasks in a way that allows for positive interactions between the task gradients. For the model architecture, we used minBERT with PALs as it has shown promising performance in previous experiments. We used the default parameters for the learning rate, batch size, and hidden dropout probability.

Results. After implementing PALs with gradient surgery, we achieved a development classification accuracy of 49.6% on the SST dataset, a development accuracy of 75.3% on the Quora dataset, and a development Pearson correlation coefficient of 0.409 on the STS dataset.

Analysis. At first, we hypothesized that adding gradient surgery should improve the performance of multi-task learning by resolving conflicting gradients.[5] Our results show that gradient surgery with PALs did outperform the baseline model with additive loss. However, gradient surgery failed to perform better than sequential training for our datasets. One explanation is that since gradient surgery adds gradients of three tasks in one step, the interference from other tasks still remains (despite not conflicting). Secondly, the sizes of our datasets are very imbalanced: the Quora dataset has 141498 training data, while the STS dataset has only 6040 training samples. Since we had to limit the batch size to at most 16 due to memory constraints, we could not iterate over the entire Quora training dataset when training with additive loss. By comparison, sequential training trains every dataset sequentially during each epoch, and therefore sees more training samples for the Quora dataset. This provides a valid explanation of why the performance of gradient surgery on our paraphrase detection task is worse than sequential training.

5.3.5 Experiment 5: Low Rank with Weight Decay

Description. Since both the PALs and low rank layers showed overfitting, we experimented with different values of weight decay λ for the AdamW optimizer. We used the default parameters of a learning rate of 1e-5, a batch size of 8, and a hidden dropout probability of 0.3. We trained the model with $\lambda = 0.01$ and $\lambda = 0.001$

Results. Using weight decay $\lambda = 0.01$, we achieved a development classification accuracy of 51.1% on the SST dataset, a development accuracy of 81.6% on the Quora dataset, and a development Pearson correlation coefficient of 0.409 on the STS dataset. A weight decay of $\lambda = 0.001$ gave almost the same results as $\lambda = 0.0$.

Analysis. We hypothesized that weight decay as a form of regularization should reduce overfitting. Our results mostly aligned with our hypothesis that $\lambda = 0.01$ improved the classification accuracy of both sentiment analysis and paraphrase detection, compared to the model without regularization. However, the performance on the semantic textual similarity task became worse, causing the overall score to be lower than the baseline sequential model. We proposed that PALs and low rank layers added considerable complexity to the model and thus were the main source of overfitting. Although adjusting weight decay forced the model to learn "less" during training (Table 1), it did not reduce model complexity, so overfitting persisted.

5.3.6 Experiment 6: PALs with Annealed Sampling

Description. We hypothesized that the difference in dataset sizes caused the model to fit the large Quora dataset much more than the smaller STS and SST datasets. Our previous experiments often obtained an accuracy of around 80% on the Quora dataset, compared to an accuracy of around 50% on the SST dataset and a correlation of around 0.5 on the STS dataset. Therefore, we implemented annealed sampling in combination with PALs. We used the default parameters of a learning rate of 1e-5, a batch size of 8, and a hidden dropout probability of 0.3.

Results. After training the PALs model with annealed sampling, we achieved a development classification accuracy of 51.2% on the SST dataset, a development accuracy of 76.1% on the Quora dataset, and a development Pearson correlation coefficient of 0.407 on the STS dataset.

Analysis. We hypothesized that annealed sampling was able to balance between datasets and guar against interference between tasks. Our results did not align with our hypothesis as the model did not outperform the baseline sequential model. We proposed that the reason might be our selected number of training steps in each epoch is computed as the size of the STS dataset divided by the batch size and times 3, which is small for the SST and the Quora dataset. One potential improvement would be to allow for sampling with replacement and increase the step size, so that the model is able to fit more portions of the datasets.

5.4 Overall Results

Report the quantitative results that you have found so far. Use a table or plot to compare results and compare against baselines.

	Train SST	**Train Quora**	**Train STS**	**Dev SST**	**Dev Quora**	**Dev STS**	**Dev Avg Score**
Baseline+sequential	73.4%	94.2%	.918	46.7%	76.0%	.519	.582
Baseline+additive loss	64.9%	73.4%	.911	49.5%	72.6%	.423	.548
PAL+sequential	94.3%	98.8%	.939	48.9%	79.0%	.499	.593
PAL+grad surgery	76.7%	76.8%	.922	49.6%	75.3%	.409	.553
Lowrank+sequential	86.7%	98.5%	.912	50.6%	80.1%	.481	<u>.596</u>
Lowrank+sequential+λ=0.01	92.4%	99.1%	.906	51.1%	81.6%	.409	.579
PAL+annealed	63.1%	78.1%	.666	51.2%	76.1%	.407	.560

Table 1: Model performances on the development set (fine-tuned)

	Test SST	**Test Quora**	**Test STS**	**Test Avg Score**
PAL+sequential	48.3%	79.4%	.510	.596
Lowrank+sequential	51.4%	80.6%	.476	<u>.599</u>

Table 2: Model performances on the test set

The test results were what we expected based on the performances on the development set. Nevertheless, the performances of PALs or low rank layers were different from our expectations. The original paper claimed that the PALs + annealed sampling configuration was able to achieve performances close to finetuning the BERT model on each individual task. However, we found that the additional layers in the BERT layers turned out to be adding much complexity to the model and caused the model to overfit greatly during the training stage. We hypothesized that overfitting, along with the difference in dataset sizes, were the main obstacles in our experiments. Our additional experiments with weight decay and annealed sampling reduced overfitting but did not improve model performance. We suggest that further approaches should be explored.

6 Conclusion

After experimenting with different model structures and training methods, we arrived at the following conclusions.

1. Compared to the baseline model, adding **task-specific parameters** to each BERT layer improved the model performance on multi-task learning. With sequential training, our implementation of **Low Rank Layers** achieved the best performance on the tasks of sentiment analysis and paraphrase detection, and has the highest development average score.
2. We experimented with different training methods, e.g. gradient surgery and annealed sampling to improve the performance. However, their effects are not obvious as in the original paper. We believe that the effects of training techniques would vary for different model structures and tasks.
3. Despite the lower performance, we noticed that training with sampling greatly **enhanced the training speed** compared to training all tasks in sequence. In practice, we need to evaluate the trade-off between training efficiency and model performance when designing large language models.

Potential future work could include experimenting different sampling methods to address the vast difference in dataset sizes. For example, we could tweak annealed sampling to allow for sampling with replacement and increase the step size in each epoch, so that more portions of the larger datasets could be fit by the model. We could also experiment with uniform sampling with replacement or square root sampling where $p_i \propto N_i^\alpha$ and $\alpha = 0.5$. Given more time, we could further train the models with different hyperparameters such as the number of epochs and learning rates and explore other forms of regularization.

References

[1] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, 2022.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Bernhard Schölkopf, John Platt, and Thomas Hofmann. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference (Bradford Books)*. The MIT Press, 2007.

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

[5] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.