# minBERT and Multitask Learning Enhancements

Stanford CS224N {Default} Project

**Naman Govil**
Stanford University
`namang7@stanford.edu`
Mentor: Andrew Lee

## Abstract

This project implements minBERT (based on the BERT model) and applies it successfully to multiple downstream Natural Language Processing tasks such as Sentiment Analysis, Paraphrase Detection and Semantic Textual Similarity. This work leverages multi-task learning to update shared model parameters and improve performance across all three tasks. Multi-task learning is a crucial technique to enable sharing of parameters across tasks to reduce computational, storage and energy overheads of task specific neural networks. In exploring improvements to the model, techniques such as implementing a combined loss function incorporating losses from each task, Gradient Surgery procedure to improve conflicting parameter updates, random data sampling to avoid over-fitting on any one single task and further finetuning by leveraging cosine similarity on the STS task are applied. The results demonstrate that a BERT-based multi-task learning architecture, combined with all the techniques listed above significantly improved model performance over the baseline of sequentially training on each task by $19\%$.

## 1 Introduction

Natural Language Processing (NLP) is making tremendous growth in recent years thanks to advancements in deep learning models like BERT (Bidirectional Encoder Representations from Transformers). Devlin et al. (2019) show that BERT can achieve impressive performance when fine-tuned on various downstream NLP tasks. Many existing methods for improving model performance on individual NLP tasks rely on separate models per task. However, for systems that execute on resource (such as compute, energy and memory) constrained devices like edge computing nodes or Mobile System on Chips (SoCs), fine-tuned networks for individual tasks would be very costly and not a scalable growth tactic. The goal of this project is to develop an effective multi-task learning architecture where transformations and parameters can be shared for multiple end output goals. Additionally, it aims to validate technical learnings from contemporary research that state training a model that learns shared encodings can help improve the performance of each individual task and also generalizes well across unseen tasks.

In this work, the three tasks considered are sentiment analysis (SST), paraphrase detection (PARA), and semantic textual similarity (STS). The key ideas behind the approach applied are to first start with a transformer-based pretrained model, then building task-specific heads on top of this model which can be trained with task-specific data. Instead of training the heads separately, the heads will all be trained together using a combined loss function along with a technique for avoiding conflicting gradients, called gradient surgery (Yu et al., 2020). Furthermore, techniques such as random data sampling to avoid overfitting as well as finetuning with cosine similarity to capture the semantic similarity between similar sentence pairs (Reimers and Gurevych, 2019) are also applied. Finally, all these approaches are combined together and model performance is reported. It achieves an average score of 0.671, with SST dev accuracy of 0.520, PARA dev accuracy of 0.742 and STS dev correlation of 0.501, thereby outperforming baseline in all three tasks, as detailed in section 4.4.

## 2 Related Work

Several recent high performing NLP models are based on the Transformer architecture which utilizes multi-head attention (Vaswani et al., 2017), including the original BERT paper by Devlin et al. where the authors additionally fine-tune independently on all the GLUE benchmark tasks (Wang et al., 2018). In the paper "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," the authors present a model architecture consisting of two BERT models with shared weights where embeddings from two input sentences are compared using cosine similarity and fed into a mean squared error loss function. A similar technique to finetune and improve the performance of the semantic similarity task head is applied in this work.

There has also been significant research in exploring multitask learning techniques. The paper "MTRec: Multi-task Learning over BERT for News Recommendation" (Bi et al., 2022) introduced a novel multi-task learning framework that incorporated downstream tasks as auxiliary tasks in the training process. This approach resulted in deeper and more representative BERT embeddings, leading to improved performance over the baseline single-task learning. The paper also successfully utilized the Gradient Surgery (Yu et al., 2020) technique to resolve conflicting gradients and further improve model performance. Both of these techniques are also applied in this work, and further, a combination of cosine similarity finetuning alongside gradient surgery is also experimented with.

## 3 Approach

In this section, the baseline model architecture, classification pipelines and enhancements applied to improve model performance are explained. The minBERT model consists of 12 transformer layers with multiheaded self-attention and outputs bidirectional word embeddings which are used to represent the sentences in high dimensional vector space. Details of the architecture can found in the default project handout and skipped here for brevity.

### 3.1 minBERT Sentiment Pipeline

After implementing the minBERT model based on instructions in the project handout, as part 1, a single-task classifier pipeline for sentiment analysis is developed. The [CLS] token embedding from the final layer of BERT is extracted and passed through a dropout and a fully connected layer to classify the sentiment of the sentence into one of five classes (negative, somewhat negative, neutral, somewhat positive, positive). The sentiment analysis task utilizes cross entropy loss to pretrain and finetune the model.

### 3.2 Multitask Pipelines

As part2, multitask classifier pipeline for the three downstream tasks is developed. The three classifier pipelines are detailed as follows.

- Sentiment Analysis: The technique described above of using the [CLS] token embedding from the final layer of BERT to represent the sentence and passing through a dropout and fully-connected layer is applied. The output of the model is a set of 5 logits (unnormalized values) for each sentence.
- Paraphrase Detection : The same pre-trained BERT model is applied on both sentences separately. Then, the [CLS] token from both sentences are concatenated. This concatenated vector is passed through a dropout and a fully connected layer to predict whether the two sentences are paraphrases or not. The output of the model is a single unnormalized logit for each pair of sentences.
- Semantic Textual Similarity : This pipeline is also constructed similar to paraphrase detection above, by concatenating the two sentence representations and deriving one logit for the pair of sentences.

### 3.3 Adam Optimizer

As suggested in the project handout, the Adam optimizer is implemented and utilized during training. Adam optimizer is a stochastic optimization algorithm that computes adaptive learning rates for different parameters by estimating the first and second moments of the gradients. This helps in adapting to the varying importance of different parameters, thereby leading to faster convergence compared to vanilla Stochastic Gradient Decent (SGD).

## 3.4 Baseline

The baseline used in this project is to create the multitask pipeline explained above and then train each task sequentially on their respective datasets. That is, in each training epoch, train through the entire dataset for each task one after the other. Cross entropy loss is applied as the loss function for SST, Binary cross entry loss with logits is applied for PARA and mean square loss is applied for STS.

## 3.5 Extention 1: Multitask Learning to update shared parameters and random sampling

To leverage multi-task learning to update BERT, the first update made is to train on all three tasks concurrently using a combined loss function which sums the losses from each task.

$$L_{Total} = L_{SST} + L_{Para} + L_{STS} \tag{1}$$

In each training epoch, a batch from each of the 3 datasets is evaluated, adding together the 3 losses and then taking an optimizer step. Since the three datasets had quite varying sizes (PARA having a significantly larger dataset compared to SST/STS), in order to avoid overfitting on one particular task, per each epoch I only iterate over the minimum number of batches across the three data sets. To make sure good coverage over the entire dataset is achieved, the shuffle parameter in the PyTorch DataLoader is used to make sure a random set of batches are picked every epoch. I also experimented by iterating over the maximum number of batches across the three datasets and using itertools to infinitely extend the smaller datasets but this significantly increased the training time (almost 5X) without very meaningful improvement in model performance.

To retain the best model, the average of the three dev metrics - SST accuracy, PARA accuracy and STS pearson coefficient is used, only overwriting the old model with the newly saved best model if it achieves better average accuracy.

## 3.6 Extention 2: Applying Gradient Surgery

Next, I experimented with using gradient surgery to update parameters instead of just a plain loss sum used in the previous approach. I used Wei-Cheng Tseng's PCGrad (Tseng, 2020) to implement this approach. PCGrad compares pairwise gradients in a round-robin manner and projects a conflicting gradient onto the normal plane of the other. When two gradients are conflicting, i.e. their dot product is negative, one gradient $g_i$ is updated as follows:

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

## 3.7 Extention 3: Applying Cosine Similarity to the Semantic Text Similarity (STS)

Next, I experimented with introducing cosine similarity into the semantic textual similarity task heads instead of using a linear layer. To implement this, the BERT embeddings for the two sentences are generated separately first, then, their cosine similarity is calculated which is scaled to the range [0, 1] with 0 implying orthogonal or unrelated sentences and 1 implying perfect overlap. The ground truth labels are also scaled to the same range [0, 1] to use mean square error loss function while training.

## 3.8 Extention 4: Combining everything together

Finally, all the extensions listed above are combined into a singular model and performance is evaluated.

## 4 Experiments

This section describes the datasets used and the evaluation metrics applied. It also explains the experiment setup details and reports the results obtained.

## 4.1 Data

The datasets used for the sentiment analysis task comes are Stanford Sentiment Treebank (SST) (Socher et al., 2013) and the CFIMDB dataset. The SST dataset has 11,855 examples, and the CFIMDB dataset has 2,434 examples. The dataset used for paraphrase detection task is a subset from Quora (Quo, 2020), and consists of 202,152 examples. Finally, the dataset used for STS task is the SemEval STS Benchmark Dataset (Agirre et al., 2013), and consists of 8,628 examples. The splits in the dataset are given in Table 1.

| Dataset | Training Set | Dev Set | Test Set |
|---|---|---|---|
| SST | 8544 | 1101 | 2210 |
| CFIMDB | 1701 | 245 | 488 |
| Quora | 141,506 | 20,215 | 40,431 |
| SemEval STS | 6041 | 864 | 1726 |

Table 1: Datasets used and split training vs dev vs test

## 4.2 Evaluation metrics

The evaluation metric for SST and PARA tasks is accuracy, because it represents the proportion of correctly classified instances among all instances in the dataset. For STS task, the Pearson correlation coefficient is used as it measures the linear association between the predicted similarity scores and the actual similarity score.

## 4.3 Experimental details

For both parts, there are two training modes: first, "pretrain", where BERT model parameters are unchanged and classifier task head (or heads in part 2) is fully trained. Second, "finetune", where all parameters including BERT model are updated. For both parts, the training and evaluation is performed on an Nvidia Tesla T4 (Turing) 16 GB GPU on Google Cloud.

For part 1, the single-task BERT implementation is trained on the sentiment analysis task alone. The hyperparameters used are number of epochs of 10, batch size of 8, learning rate of 1e-3 (pretrain) and 1e-5 (finetune), a dropout probability of 0.3, and a hidden layer size of 768. Both the SST dataset and CFIMDB dataset are relatively small, training takes about 26 second per epoch on SST and 49 seconds per epoch on CFIMDB. The model with the highest dev set accuracy out of all epochs was chosen as the best model. The results are summarized in the next section.

In part 2, with the addition of multitask learning on the three tasks and new datasets Quora and SemEval STS, I realized the size of the dataset for PARA was significant larger compared to datasets for SST and SemEval STS, so there was inequality in number of batches to be used for training. To counter that, first, I experimented with increasing batch size to reduce the total number of batches. Based on the available GPU memory on T4, I was able to increase the batch size to 16. Next, in order to avoid overfitting on PARA, I experimented with random data sampling to figure out the optimal strategy for picking the number of batches to train on, experimenting with least number of batches across the three datasets to the maximum. I noticed a large increase in training time when increasing the number of batches without significant gain in model performance. I opted for quicker training iteration time to experiment with more changes and extension techniques. I also experimented with epoch size of 10, 15 and 20 and noticed accuracy scores leveled off after 10 epochs. I experimented with varying dropout probability to 0.3, 0.5, and 0.8 and didn't notice too much variance in performance. The other hyperparameters that weren't changed were learning rate of 1e-3 (pretrain) and 1e-5 (finetune) and a hidden layer size of 768. The per epoch training time for the varying configurations are also reported in the next section.

## 4.4 Results

For part 1, the results achieved for minBERT on sentiment analysis task on the dev set are presented in Table 2.

| Training Type | SST Accuracy | CFIMDB Accuracy |
|---|---|---|
| Pretraining | 0.390 | 0.780 |
| Finetuning | 0.517 | 0.971 |

Table 2: Part 1 sentiment analysis dev results

For part 2, the results obtained for the baseline and various extensions employed as described in section 3 are presented in Table 3. The model configurations experimented with are summarized as follows:

1. **Baseline**: Sequential training per task per epoch as described in section 3.4.
2. **Multitask** : Concurrent training on the three tasks per epoch with combined loss function and random data sampling as described in section 3.5.
3. **Multitask + PCGrad** : Multitask with addition of gradient surgery as described in section 3.6.
4. **Multitask + Cosine** : Multitask with cosine similarity finetuning for STS task as described in section 3.7.
5. **Multiask + PCGrad + Cosine** : Multitask with gradient surgery and cosine similarity combined, as described in section 3.8.

| Model Configuration | SST Accuracy | PARA Accuracy | STS Corr. |
|---|---|---|---|
| Baseline | 0.391 | 0.670 | 0.262 |
| Multitask | 0.510 | 0.734 | 0.336 |
| Multitask + PCGrad | 0.514 | 0.734 | 0.346 |
| Multitask + Cosine | 0.513 | 0.741 | 0.458 |
| Multitask + PCGrad + Cosine | **0.520** | **0.742** | **0.501** |

Table 3: Part 2 Multitask Learning on SST, PARA and STS task dev results

The results on the test set with the combined model (highest performing) are presented in Table 4. The best overall result is obtained when all the optimization techniques are applied together, it improves SST accuracy by 33% over baseline, PARA accuracy by $10.75\%$ over baseline and STS correlation by $91\%$ over baseline. The overall average dev score across the three tasks jumps to **0.671** from the baseline score of **0.564**, improving by $19\%$.

Applying cosine similarity for finetuning STS task classifier worked really well, which is inline with expectations as cosine similarity should be a good measure for how closely related two sentences are. Applying gradient surgery (PCGrad) didn't give as much performance boost as I had expected. This could be because the gradients across the three tasks weren't completely orthogonal to each other. This could be expected since the three NLP tasks are similar in goal with respect to each other and rely on similar learning objectives.

| Model Configuration | Average | SST Accuracy | PARA Accuracy | STS Corr. |
|---|---|---|---|---|
| Multitask + PCGrad + Cosine | 0.669 | 0.528 | 0.743 | 0.472 |

Table 4: Part 2 Multitask Learning on SST, PARA and STS task test results

In Table 5, the training time per epoch in seconds is also reported, since this is an important aspect of training deep learning models. For baseline (sequential training per dataset) the split is SST = 89 sec, PARA = 2100 sec and STS = 90 sec. The training time is significantly higher when processing all batches of the Quora dataset every epoch, due to it's large sample size. Applying techniques like random sampling across epochs turned out to be an effective strategy to reduce iteration time and retain decent model performance.

## 5 Analysis

Several experiments were conducted through the course of this project, including hyperparameter tuning for balancing model performance with execution efficiency and iteration time. Increasing

| Model Configuration | Time (s) |
|---|---|
| Baseline | 2279 |
| Multitask | 204 |
| Multitask + PCGrad | 258 |
| Multitask + Cosine | 201 |
| Multitask + PCGrad + Cosine | 259 |

Table 5: Part 2 Multitask Learning training time per epoch (in seconds)

batch size till the working set could fit in the GPU memory helped reduce the overall latency of training. Applying random sampling across epochs was an effective way to tradeoff iteration time and model performance. Varying dropout rates didn't change model behavior much, which would indicate a small amount of dropout must have been sufficient for preventing overfitting and thus the smallest dropout rate was good enough. Including cosine similarity on the STS task improved the performance because it is a measure of the similarity between two vectors in a high-dimensional space, which is an effective way to compare the semantic similarity between two texts. Gradient surgery wasn't as effective as I had expected, hinting that parameter gradients across the three tasks weren't always orthogonal to each other.

Further qualitative analysis of the model were conducted by inspecting few model outputs on the three tasks. For the SST task, shorter sentences with clear positive or negative language turned out to be best test cases which predicted correctly with highest accuracy. For example, " A deep and meaningful film ." got labeled correctly as (4). However, sentences without clear positive or negative words, or mix of both, for example, " Good film , but very glum .", weren't accurately classified, it was labeled as (2) as opposed to (3). This signals that the model lacks the ability to adequately interpret context or balancing moods like a human would.

For paraphrase detection, many inaccurate test cases were found which although conveyed the same meaning but used different sequence (syntactics) and/or different connotations. For example, "Why are Facebook, Google, and others not allowed in China?" and "Why are Google, Facebook, YouTube and other social networking sites banned in China?" are classified as not paraphrases. This signals the model doesn't recognize synonymous words or phrases with lexical variations. Incorrect labeling was also observed on sentences that contained overlap of similar words but conveyed different meaning. For example, "How do I become great?" and "How do I become a great doctor?" are labeled as paraphrases. This may be due to the fact that the model has not been trained on enough examples of similar in sequence but not identical sentences.

For STS task, it was observed that model performance trails off when context based complex language is applied. As an instance, the model failed to capture polysemy in sentences. For example, "Work into it slowly" and "It seems to work" has the ground truth similarity of 0.0, while the model predicted this with a similarity score of 3.7. This indicates an inability to differentiate between the different meanings of the word "work".

## 6 Conclusion

During the course of this project, I delved into the inner workings of the BERT model, implemented self-attention, built pipelines for pre-training and fine-tuning, and adapted the model for the three NLP tasks. Through experimentation, I learnt hyperparameter tuning and how various aspects of batch size and dataset size are reasoned about to efficiently train models on the GPU hardware available. I implemented extensions like cosine similarity and gradient surgery and was able to demonstrate significant model performance improvement for all three tasks over the baseline by using an ensemble of all these techniques. I was especially proud of this achievement, given this work is an individual project. Possible future work include applying cosine similarity to the PARA task as well as experimenting with using cosine similarity as a feature alongside downstream linear layers to improve STS/PARA accuracy. Another set of avenues could be exploring alternative finetuning techniques for each task and improving embeddings by applying Multiple Negative Ranking Loss Learning. A key limitation of this work was not exploring training on more data samples, beyond the already provided datasets, this would have helped generalize the model more. Another limitation was varying epoch sizes didn't lead to more performance in the concurrent training phase, I would need to spend more time to understand why.

## References

2020. 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, 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, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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.

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.