Multitask Finetuning for MinBERT

Stanford CS224N Default Project

Ethan Boneh
Department of Computer Science
Stanford University
`ethanbc@stanford.edu`

Abstract

We investigate the efficacy of several techniques to fine-tune BERT for downstream tasks. By experimenting with methods including cross-attention, bi-encoded cosine similarity, annealed sampling, gradient surgery, and contrastive learning, we improve our base BERT model on sentiment analysis, paraphrase detection, and semantic text similarity. We find that utilizing cross-attention, gradient surgery, and contrastive learning is most effective for improving BERT's performance on downstream tasks. Our approach yields a 0.787 score on the test set, demonstrating an improvement above the baseline. These results highlight the potential of advanced fine-tuning methods to significantly enhance the capabilities of pre-trained language models.

1 Key Information to include

- Mentor: Rashon
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

Rather than training large models from scratch, which is computationally expensive and data intensive, NLP research today embraces transfer learning. In this paradigm, pretrained language models are fine-tuned on downstream NLP tasks to achieve State-of-the-Art performance.

A landmark contribution to this trend is BERT (Devlin et al. (2019)), or Bidirectional Encoder Representations from Transformers. BERT is pretrained with Masked Language Modeling and Next Sentence Prediction tasks on the Wikipedia and BookCorpus datasets. However, while BERT embeddings can encode rich contextual information and perform well on downstream tasks with standard full-fine tuning methods, several steps can be taken to improve upon these baselines and achieve better performance for multitask learning.

We evaluate several of these improvement techniques, broadly across four categories: encoding methods, data sampling methods, gradient methods, and contrastive learning methods. With these improvements, we hope to develop a model capable of performing well in multitask learning. Specifically, we evaluate our model's performance on sentiment analysis using the Stanford Sentiment Treebank (SST), its ability to detect paraphrasing using the Quora Question Pairs dataset (QQP), and its ability to detect semantic textual similarity via the SemEval STS dataset (STS). Our strongest model, B2 (referenced below), achieves a score of 0.502 on SST, 0.905 on QQP, and 0.877 on STS dev datasets.

3 Related Work

Methods of fine-tuning BERT for multitask learning have been explored thoroughly. Our work is inspired by the following papers:

Bi-encoding: While the original BERT paper uses cross-encoding for inputs with multiple sentences, Reimers and Gurevych (2019) experiment with bi-encoding, where inputs are fed separately into the models and then concatenated and passed through either an MLP or cosine similarity. They show that this approach boosts performance on similarity tasks.

Gradient surgery: Naive multitask fine-tuning involves simply accumulating the gradients after each batch and running the optimizer on the result. Yu et al. (2020) propose an alternative approach where conflicting gradients get projected onto normal planes, thus minimizing destructive interference. The team shows that this improves multi-task learning.

Contrastive learning: Gao et al. (2022) demonstrate that BERT can be further fine-tuned using contrastive learning to improve performance on similarity tasks. We explain further details in section 4.

4 Approach

We develop on a standard MinBERT architecture with 12 layers, each consisting of multi-head self attention, an add-norm, an MLP, and another add-norm. Since we are doing sentence classification, the input for every task is the [CLS] token appended to the start of every input sentence in BERT, which is transformed through a pooling layer. We optimize with an AdamW Optimizer.

4.1 Baselines:

For our baseline, we leverage the pooled embedding for the [CLS] token. For sentiment analysis, we feed the output through a linear layer, and train with cross-entropy loss. For paraphrase detection, we employ bi-encoding—that is, we feed sentences through a Siamese network, concatenate the outputs, and feed the result through an MLP with one hidden layer. We train this with binary cross-entropy loss. For semantic text similarity, we employ a similar scheme as for paraphrase detection but train the result of the MLP with MSE Loss (since it is a regression task).

We sample data in a round-robin fashion, so that with each iteration we see one batch from each dataset. Every epoch runs for as many batches as the longest dataset (in this case QQP), and the other datasets start cycling once we get to their ends. We sum the loss functions for our backward phase.

4.2 Encoding Methods:

Rather than using an MLP to transform pooled outputs into a similarity ranking, for STS we experiment with using cosine similarity to compute the similarity between the two sentences' [CLS] tokens, given by $\frac{h_i \cdot h_j}{\|h_i\| \|h_j\|}$. Specifically, we use cosine similarity between the two sentences' [CLS] tokens to compute similarity ranking, and minimize our error with MSELoss. This may improve performance because cosine similarity directly relates direction to semantic meaning, which is what BERT sentence embeddings have been shown to encode (Reimers and Gurevych (2019)).

We also experiment with cross-encoding, where we first concatenate input sentences and then feed the result into the BERT model, before transforming the resulting [CLS] token into a similarity score. We reason that BERT was trained with the ability to process multiple sentences in this way, and therefore by encoding independently we lose crucial inter-sentence contextual information that the model can leverage to determine similarity (see Figures 1, 2, and 3 for graphical depictions of our 3 methods).

4.3 Data Sampling Methods:

Our baseline round-robin sampling strategy iterates through every dataset's data on every epoch, but we reason that since the QQP dataset is almost two orders of magnitude larger than the other two datasets, we see the same data from the other two datasets too frequently and are thus overfitting. To combat this issue we explore several fixes.

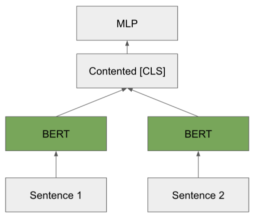

Figure 1: Bi-encoding with MLP

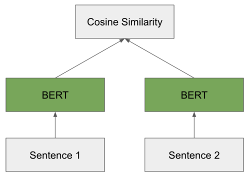

Figure 2: Bi-encoding with Cosine Similarity

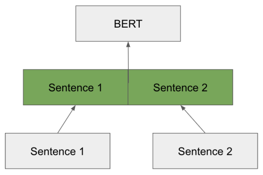

Figure 3: Cross-encoding

The first is to vary each dataset's batch size with the size of the dataset. Specifically, we assume we want each epoch to iterate through 500 batches, so task i's batch size is $\frac{\text{dataset i's size}}{500}$.

After seeing that this leads the model to overfit on the QQP data and ignore the other two tasks, we pivot to testing Stickland and Murray (2019)'s annealed sampling strategy. In annealed sampling, we randomly sample a batch from task i with probability

$$\begin{aligned} p_i &\propto N_i^{\alpha} \\ \alpha &= 1 - 0.8\frac{e-1}{E-1} \end{aligned} \tag{1}$$

where N_i is the number of training examples for task i, e is the current epoch we're on, and E is the total number of epochs. The intuition here is that we want to make use of all training data without looping on the smaller datasets, and thus sample roughly proportionally to the size of each dataset. However, Stickland and Murray find it is better to reduce vast discrepancies between datasets and start sampling with roughly equal probabilities toward the end of training, thus introducing the alpha term which bring sampling probabilities closer together as time goes on. Unfortunately, despite this reformed approach we still noticed that the model was underperforming on SST and STS, likely due to how we were neglecting them. The paraphrase detection gradients also probably interfered with any optimizer steps optimizing for SST or STS, hurting our model's ability to perform well on those tasks.

Having experienced difficulties attempting to leverage the entire quora dataset, we turned back to round-robin sampling with equal batch sizes, and capped the number of iterations per epoch (somewhat arbitrarily) at 500. This significantly reduced overfit, while not affecting performance on the Quora dataset. However, in this approach we still did not see the entire QQP dataset.

As such, for our last technique we introduce staggered round-robin sampling, inspired by Jack Le and Febie Lin's CS224n final project report[1], where we run our model across the entire Quora dataset for two epochs, then go back to training on all datasets with round robin sampling for 500 iterations. By doing so, our model has the chance to develop strong embeddings for paraphrase detection and then continue improving for SST and STS.

4.4 Gradient Methods:

Our method of simply summing the losses for each task and performing gradient descent on the result leaves room for gradient interference, where gradients point in opposite directions and cancel each other out. To fix this issue, we use gradient surgery, introduced by Yu et al. (2020); specifically, we employ PCGrad (Project Conflicting Gradients). Given two gradients g_i and g_j for tasks i and j, if the two gradients conflict with one another, meaning their cosine similarity is negative, we project one of the gradients onto the normal plane of the other by subtracting out the projection onto the other gradient (see Figure 4 for a geometric intuition):

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

[1]https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1244/final-projects/FebieJaneLinJackPLe.pdf

We randomly chose the gradient we're projecting and the gradient being projected onto in order to make sure we don't bias the model in favor of one task. Our implementation is uses Antoine Nzeyimana's PyTorch 1.11 adaptation of the original implementation[2].

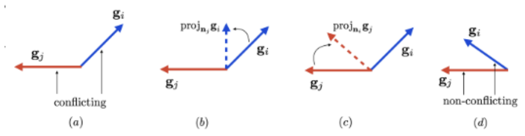

Figure 4: Geometric Description of PCGrad, taken from Yu et al. (2020). We project interfering gradients onto normal planes.

4.5 Contrastive Learning:

Lastly, we implement Unsupervised SimCSE, introduced by Gao et al. (2022), for additional fine-tuning. SimCSE is a contrastive learning framework that seeks to bring semantically similar sentences' representations closer together while pushing away semantically different sentences' embeddings. To do so, we use the Wikipedia dataset and, for each batch, feed a single input sentence into the model twice. Since the model randomly applies dropout, this results in two different [CLS] tokens, which we treat as "positive pairs." We then feed in the rest of the sentences in the batch, which we treat as negatives. Because we want our model to output similar embeddings for the same given input, we try to maximize the cosine similarity of positive pairs and minimize the similarity of negative pairs through the objective function

$$l_i = -\log \frac{e^{sim(h_i^{z_i}, h_i^{z'_i})/\tau}}{\sum_{j=1}^{N} e^{sim(h_i^{z_i}, h_j^{z'_j})/\tau}} \tag{3}$$

Where h_i^z is the output of the BERT model with dropout mask z, N is the number of sentences in the batch, sim represents the cosine similarity between two embeddings, and τ is a temperature hyperparameter. We train this for one epoch before proceeding to the other fine-tuning techniques. Further motivations for SimCSE are explained in section 5.4. We write the Unsupervised SimCSE implementation ourselves, but draw inspiration from Bhuvana Kundumani's Medium post[3].

5 Experiments

5.1 Data

As described in section one, we leverage the Sanford Sentiment Treebank (SST), Quora Question Pairs (QQP), and SemEval STS Benchmark (STS) datasets. SST contains single sentence movie reviews annotated with a score from 1 to 5, with 5 being most positive. QQP consists of sentence pairs which are labeled either as paraphrases of each other or not. STS consists of sentence pairs with similarity scores ranging from 0 (unrelated) to 5 (most related). Training splits are provided in the project handout, and are listed below:

- SST: 8,544 training examples, 1,101 dev examples, 2,210 test examples
- QQP: 283,010 training examples, 40,429 dev examples, 80,859 test examples
- STS: 6,040 train examples, 863 dev examples, 1,725 test examples

We also use the Wikipedia dataset for Unsupervised SimCSE, which contains 1,000,000 English sentences.

[2]https://github.com/anzeyimana/Pytorch-PCGrad-GradVac-AMP-GradAccum

[3]https://bhuvana-kundumani.medium.com/implementation-of-simcse-for-unsupervised-approach-in-pytorch-a3f8da756839

5.2 Evaluation method

Since SST and QQP are categorical tasks, we use accuracy as our evaluation metric. For STS, we use Pearson correlation between predictions and truth.

5.3 Experimental details

We employ the following hyperparameters for our experiments: a learning rate of 1e-5, a batch size of 64, 10 epochs, a dropout probability of 0.3, and a hidden size of 768. Unless otherwise specified, each epoch sees 500 batches of each task.

Additionally, for Unsupervised SimCSE we refer to the original paper's hyperparameters: a learning rate of 3e-5, a temperature of 0.05, and 1 epoch of training.

All training is done on an NVIDIA A100 through Modal Labs. We were unfortunately unable to do a full hyperparameter sweep due to compute and time constraints, but found that batch size 64 was better than 32 or 128, and our learning rate 1e-5 was better than 1e-3.

5.4 Results

Results for our encoding method experiments are given in Table 1.

Model	**SST Accuracy**	**QQP Accuracy**	**STS Correlation**
Baseline	0.486	0.773	0.31
Baseline + Cosine Similarity	0.49	0.753	0.354
Baseline + Cross-Encoding (B1)	**0.495**	**0.848**	**0.861**

Table 1: Dev Performance of different encoding architectures

As we can see, cross-encoding yields the best results of the three methods we tried. This is likely because it takes advantage of BERT's pretraining format, which accepts 2-sentence inputs if they are concatenated together. Concatenation allows the model to learn nuanced interactions between input sentences, necessary for paraphrase detection and STS. Since the tasks are related, this newfound ability transfers into boosted performance for SST as well.

Cosine similarity probably worked better than feeding the independent outputs through an MLP because it directly captures the sentence embeddings' closeness, thus being simpler to learn than an additional classification network.

Having figured out our model's architecture, which we label B1, we now evaluate our additional fine-tuning techniques, demonstrated in Table 2:

Model	**SST Accuracy**	**QQP Accuracy**	**STS Correlation**
B1	0.495	0.848	0.861
B1 + PCGrad	0.505	0.867	0.86
B1 + PCGrad + Staggered RR Sampling (B2)	**0.516**	**0.905**	**0.877**
B1 + PCGrad + SimCSE	0.500	0.848	0.853
B1 + PCGrad + Staggered RR Sampling + SimCSE (B3)	**0.502**	**0.900**	**0.870**

Table 2: Dev Performance of various fine-tuning techniques

The first result of note is that PCGrad outperforms B1, likely due to the theoretical reasons Yu et al. (2020) lay out: by minimizing interference between gradients, the model's improvement in one task does not harm its performance on others as much.

As far as our sampling methods go, staggered round-robin sampling greatly outperformed all other methods. We do not include our data from variable batch size or annealed sampling because training was stopped three-quarters of the way through, after consistently seeing performance on SST and STS that was 2 orders of magnitude below B1. Staggered Round-Robin sampling probably worked best because it exposed the model to the entire QQP dataset, which enhanced its performance on QQP and probably transfered over to SST and STS embedding performance.

Lastly, we find no improvement on the dev datasets by employing Unsupervised SimCSE. However, not only does B3 beat every other approach except for B2, it actually beats B2 on the test set:

Model	SST Accuracy	QQP Accuracy	STS Correlation	Score
B2	0.510	0.906	0.881	0.785
B3	0.527	0.898	0.874	0.787

Table 3: Dev Performance of various fine-tuning techniques

As such, it is unclear whether using Unsupervised SimCSE actually improved results, or had any impact at all. We reason that perhaps our model's dropout probability was too low for the positive sentence pairs examples to be pushed close to each other in a meaningful way. Gao et al. (2022) notes that low dropout probabilities lead to degraded performance when applied to SimCSE (though our dropout probability is higher than the ones they tried). It is also interesting that in the test set sentiment analysis benefited the most, given that SimCSE was originally evaluated and developed to improve on STS.

A potential explanation for why paired tasks may suffer is that Unsupervised SimCSE tries to increase the uniformity of embeddings without degrading alignment. Intuitively, alignment can be viewed as how well the embeddings of semantically similar sentences are grouped together, while uniformity refers to how evenly spread out the embeddings are across the entire embedding space. Gao et al. (2019) provides empirical evidence that pretrained word embeddings tend to occupy a tiny cone of the overall embedding space, which can lead to the clustering of semantically different concepts and thus prevent the model from distinguishing between them. By increasing the uniformity of embeddings through contrastive learning, SimCSE is supposed to prevent the model from confusing semantically distinct concepts, and thus improve its performance on downstream tasks.

However, Unsupervised SimCSE pushes down on all sentences different than the input sentence, regardless of how semantically similar they are. If dropout is too low and the two vectors in our positive pair are too similar, we end up not actually pushing the vectors toward anything and instead simply push other vectors away, which may weaken semantic relationships built up by our pretrained model.

Thus, the model is better able to perform sentiment analysis because it confuses different concepts less often (improved uniformity), but is worse at dealing with sentence pairs (QQP and STS) because it is unable to detect encode inter-sentence similarities and differences as well (degraded alignment).

Ultimately, we find that B2 performs best on the dev set and B3 performs best on the test set, demonstrating that gradient surgery, staggered round-robin sampling, and cross-attention are effective methods of improving on a baseline BERT model full fine-tuned on sentiment analysis, paraphrase detection, and STS.

6 Analysis

We qualitatively analyze our model's outputs for SST, QQP, and STS tasks. To gain a deeper understanding of the model's performance, we plot the confusion matrix for SST in Figure 5.

Note that most errors which occur are 1 sentiment class away from the true sentiment. As such, we believe this is attributable to variations in sentiment data labeling. Sentiment labels are extremely subjective, and humans can only approximate what they believe a statement's sentiment is. As such, we argue that our model's ability to predict overall sentiment is strong.

There are some outliers, however, where the model may incorrectly attribute the sentiment of specific words to the sentiment of the sentence overall. Consider the sentence "watching the spectacle of

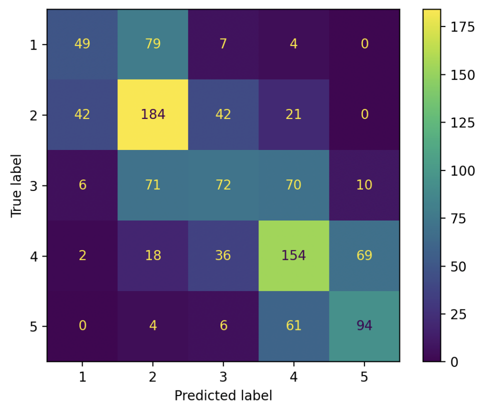

Figure 5: Confusion Matrix for SST

a promising young lad treading desperately in a nasty sea" caused the viewer to "shed an errant tear." The words desperate, tear, and nasty suggest negative sentiment, causing the model to predict a sentiment of 1. However, the true sentiment was 3 because the viewer was talking about the effect of the movie, which is actually a good thing.

For paraphrase detection, we notice that the cases our model fails tend to be when the subjects of the sentences are similar, but small details or single-word changes change the meaning of a sentence. Consider, for example, the two sentences "How did Donald Trump win the 2016 Presidential Election?" and "How is Donald Trump winning?" These two sentences are mislabeled as being paraphrases because they deal with the same topic, but ask slightly different questions.

Lastly, our model's semantic textual similarity issues are similar to its paraphrase detection ones—similar sentences differing by one rare word cause the model to mislabel the sentences' similarity scores.

7 Conclusion

In this study, we explored several advanced fine-tuning techniques to enhance the performance of MinBERT on multiple downstream NLP tasks, specifically sentiment analysis, paraphrase detection, and semantic textual similarity. Our approach incorporated cross-attention, gradient surgery, and contrastive learning methods, yielding significant improvements over the baseline model. We reached a final score of 0.787 on the testing set.

Our results demonstrated that cross-attention was particularly effective in leveraging BERT's pre-training format, resulting in better performance on tasks requiring nuanced understanding of sentence pairs. Gradient surgery (PCGrad) further enhanced the model by minimizing gradient interference, thus improving multitask learning without sacrificing performance on individual tasks. Staggered round-robin sampling proved beneficial in balancing the training data from different datasets, leading to robust embeddings that generalize well across tasks.

Interestingly, while Unsupervised SimCSE did not significantly improve performance on the development set, it yielded the highest test set score in combination with other techniques, suggesting its potential under specific conditions. This highlights the importance of dropout probability and the need for further investigation into optimal hyperparameter settings for contrastive learning.

Our best-performing model (B3) achieved notable improvements on the test set, demonstrating the effectiveness of our fine-tuning strategies. These findings contribute valuable insights into the development of more robust and versatile NLP models, capable of handling a variety of language tasks with improved accuracy.

Future work could focus on a more extensive hyperparameter search, exploring additional augmentation techniques, and investigating the impact of different dropout probabilities on contrastive learning performance. Furthermore, SimCSE could be improved by exploring a mechanism in which semantically similar sentences are pushed down on less than semantically different sentences, so we can better obtain the dual goals of alignment and uniformity.

8 Ethics Statement

While our paper demonstrates improved results due to improved fine-tuning techniques, there are several potential ethical risks posed by our work. For starters, some of our methods are quite data-intensive. Unsupervised SimCSE is trained on 1,000,000 sentences; more exploration into similar methods might incentivize researchers to completely disregard copyright and train models on copyrighted data. To migitate this issue, we need comprehensive AI policy that dictates fair use of training data. Policy must establish basic guidelines for what data on the internet companies can use for training language models, and what data is labeled "protected." We can also work toward researching and prioritizing more data-efficient fine-tuning methods, setting things like data caps for our training techniques.

Furthermore, our work involves leveraging large computational resources, which raises environmental concerns due to the significant energy consumption associated with training large language models. Our total training energy costs come out to about 10 kWh, which about a third of the daily energy costs of the average US household. As AI continues to spread and grow, global compute costs will continue to scale and hurt the environment. To address this, more research needs to be done on computer hardware optimization for deep learning. Novel computing paradigms should be explored to potentially scale down energy costs on a massive level. The AI community should also be more cautious about the amount of energy it uses. Adopting a global standard of a certain amount of compute used for research per year might encourage the research community as a whole to reduce compute costs and only run high quality, data-efficient experiments.

References

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

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

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

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