Triple-Batch vs Proportional Sampling: Investigating Multitask Learning Architectures on minBERT

Stanford CS224N Default Project

Ethan Tiao
Department of Computer Science
Department of Economics
Stanford University
etiao02@stanford.edu

Rikhil Vagadia
Department of Computer Science
Department of Economics
Stanford University
rvagadia@stanford.edu

Abstract

The development of versatile language models adept at handling a wide range of tasks is crucial to the progress of NLP. In this paper, we seek to build upon minBERT's baseline sentiment analysis capabilities and extend the model so that its embeddings are capable of handling additional NLP tasks including paraphrase detection and semantic textual analysis. We find that leveraging an augmented triple-batch multitask training model performs better than an augmented proportional sampling model. Although the baseline proportional sampling model outperforms its triple batch counterpart, we observe that extending the triple batch model with cosine similarity fine tuning, gradient surgery, and separator tokens for comparison tasks leads to more balanced scores than the proportional sampling model extended with annealed sampling and separator tokens. Our results speak to the importance of guarding against over-fitting when there are disparities between dataset sizes and task difficulties.

1 Key Information to include

- Mentor: Annabelle Wang, External Collaborators: N/A, Sharing project: N/A
- Distribution of work: Ethan and Rikhil shared project work evenly. Rikhil focused primarily on extensions while Ethan focused on the training loops. Despite the primary responsibilities, both partners worked on all aspects of the project. Report writing was split evenly.

2 Introduction

Despite minBERT's sentiment analysis capabilities, the model's architecture is unfit for handling multiple NLP tasks. We explore two different ways of re-engineering minBERT's training architecture to handle multiple tasks at once. Our objective is to train minBERT to develop robust embeddings that generalize to multiple tasks including sentiment analysis, paraphrase detection, and semantic textual analysis. We then extend each of these training architectures by leveraging fine-tuning techniques that serve purposes ranging from reducing model over-fitting to preventing conflicting gradients during learning.

Our first training approach, proportional sampling, samples a batch from the three training datasets during each training step with probabilities proportional to the relative size of each task training dataset. Our second training approach, triple-batch, truncates each training dataset to the length of the smallest dataset and cycles through three batches per training step - one for each task.

We extend our two multitask training approaches with several fine-tuning methods. For our proportional sampling approach, we use annealed sampling and introduce a separator token between concatenated embeddings in the two textual comparison tasks. Broadly, these extensions serve to balance the training loop so that it doesn't over-index on tasks with larger datasets and allow the attention mechanism to distinguish input embeddings when dealing with concatenated tensors. For our triple-batch approach, we implemented a cosine similarity layer for the semantic textual analysis task, gradient surgery (Yu et. al, 2020), and a separator token for the paraphrase detection task. These extensions aim to optimize the correctness of input embedding proximity in embedding space and prevent the learning of one task from interfering with that of another.

We find that the triple batch approach in its final variation outperforms the proportional sampling counterpart. This is likely attributed to the extensions focused on managing over-fitting and conflicting gradients that were not feasible to implement in the proportional sampling method due to its training architecture.

3 Related Work

We utilized the methods outlined in "Gradient Surgery for Multi-Task Learning" (Yu et al., 2020) paper to extend our triple batch training model. This paper introduces a novel algorithm to address optimization challenges in multi-task learning relating to the adverse impact of conflicting gradients on multitask learning. Their PCGrad algorithm performs "gradient surgery" on conflicting gradients by projecting each gradient onto the normal plane of the other gradient. This allows the backpropagation step to traverse the loss surface smoothly without conflict.

We drew inspiration from the "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" (Reimers et al., 2019) paper when developing our cosine similarity + MSE loss method aimed at improving our semantic textual analysis task performance. The authors describe an adaptation of the BERT model that is enhanced with Siamese and triplet network architectures for generating semantically meaningful sentence embeddings. The authors compute the cosine similarity between sentence embeddings with a MSE loss function to efficiently identify similar sentences, allowing BERT to excel on tasks such as semantic similarity assessments and clustering.

4 Approach

Baseline BERT The architecture of the original BERT model is described in the paper "Attention is all you need" (Vaswani et al., 2018) and the pre-training paradigm is described in the original BERT paper (Devlin et al., 2018).

New Model Architecture Building on our minBERT model, we implement two distinct multitask training architectures to enable multitask capabilities: (1) proportional sampling and (2) triple batch. We extend both models using methods described below.

4.1 Proportional Sampling Training

As a baseline, we train this multitask model by sampling a batch from the three task datasets with probabilities proportional to the relative size of each dataset:

$$p_i \propto N_i$$

where p_i is the probability of sampling a task and N_i is the size of that task's dataset. A more naive round-robin approach falls victim to over-fitting smaller datasets due to example repetition. Proportional sampling circumvents this by increasing the probability that batches from tasks associated with larger datasets are sampled.

4.1.1 Annealed Sampling

In an effort to protect against over-fitting the model on tasks with small datasets, proportional sampling becomes vulnerable to over-performing on larger datasets. The Quora dataset for paraphrase detection is significantly larger than the datasets for the other tasks, leading to excellent performance on the paraphrase task at the expense of sentiment and semantic textual analysis.

Annealed sampling strikes a balance between over-fitting on tasks with small and large datasets by dynamically adjusting the sampling probabilities over time according to α:

$$p_i \propto N_i^{\alpha}$$

where α is a parameter that keeps the sampling probabilities close to their task dataset proportions towards the beginning of the training cycle and draws sampling probabilities towards a uniform distribution towards the end of the cycle. We define α as:

$$\alpha = 1 - 0.8\frac{e-1}{E-1}$$

where e is the current epoch and E is the total number of epochs in the training loop. As the number of epochs increases, the importance of the number of examples decreases and each task probability converges towards one another so that towards the end of the training cycle we stray away from oversampling batches from larger datasets. This approach was heavily inspired by the "BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning" paper (Stickland et al., 2019).

4.2 Truncated Triple-batch Training

As a baseline for our second training approach, we truncate the length of each dataset to the length of the smallest one - the SemEval dataset. In a given training step, we then iterate through a length-three tuple of batches from each task dataset, label them, and sequentially train the model on each batch. In the baseline model, we calculate loss for each task, backpropogate, and take an optimizer step. Although we sacrifice training examples, this approach allows us to bypass the issue of dataset size disparity which could lead to model over-performing on tasks with more data.

4.2.1 Cosine Similarity

We built a class to optimize embeddings and weights using cosine similarity. We hypothesized introducing cosine similarity between embeddings would help our model better learn the correct proximity of comparison texts in embedding space. Our CosineSimilarityLayer class calculates the cosine similarity of two embeddings, passing the result through a linear layer, and then scaling the logit output to a 0-5 range that corresponds to the output scores.

4.2.2 Gradient Surgery

We implemented gradient surgery as another extension for our truncated triple batch training approach, a technique used in multitask settings when dealing with conflicting gradients from different tasks. Conflicting gradients are defined as:

Let $\phi_{i,j}$ be the angle between gradients g_i and g_j. Gradients are conflicting when $cos(\phi_{i,j}) < 0$.

Gradient surgery modifies the gradients from each task in a way that reduces or eliminates their conflict, allowing the model to learn more effectively from all tasks simultaneously. It does this by projecting gradients onto the normal plane of the other task's gradients which ensures that the updated gradient is orthogonal to the conflicting gradient direction - removing the component of the gradient that interferes with the other task's learning process. For our specific implementation, we leveraged existing public code [1] that followed the specifications as outlined in "Gradient Surgery for Multi-Task Learning" (Yu et al., 2020).

4.3 Separator Token for Concatenated Embeddings

In our proportional sampling model's paraphrase detection and semantic textual analysis tasks, we inserted a separator token between embeddings prior to the concatenation of each sentence pair embedding and the linear layer processing. In our triple batch model, this was only done in the paraphrase task to preserve the cosine similarity extension. The separator token allows the model to explicitly demarcate distinct input segments, capturing the nuanced relationships and context within and across these segments. This led to improved performance on tasks that require an understanding of the relationship between separate text segments.

[1]https://github.com/WeiChengTseng/Pytorch-PCGrad.git

5 Experiments

5.1 Data

We pre-train and fine-tune our minBERT model to complete three distinct tasks: (1) sentiment analysis; (2) paraphrase detection; and (3) semantic textual similarity. The first task uses the Stanford Sentiment Treebank (SST) dataset consisting of $11,855$ single sentence movie reviews and their sentiment labels and the CFIMDB dataset consisting of $2,434$ movie reviews with postive and negative binary labels. The second task uses the Quora dataset consisting of $400,000$ question pairs with binary labels indicating whether particular instances are paraphrases of one another. The third task uses the SemEval STS Benchmark dataset consisting of $8,268$ sentence pairs labelled by their similarity on a scale from 0 to 5.

5.2 Evaluation method

The sentiment analysis and paraphrase detection task are evaluated using accuracy – the proportion of correctly classified examples out of the total number of examples. The semantic textual analysis task is evaluated using the Pearson correlation coefficient - a metric that quantifies the linear correlation between predicted and actual similarity scores.

5.3 Experimental details

For our original minBERT model trained for sentiment analysis, we pre-trained with a learning rate of $1 * 10^{-3}$ and fine-tuned with a learning rate of $1 * 10^{-5}$. The batch size was 64 for the SST dataset and 8 for the CFIMDB dataset. We ran 10 epochs with a dropout probability of 0.3. In both multitask training architectures, we used the same learning rates as the original BERT model. Our batch size was 8 for all three tasks and the dropout probability was 0.3. Both models used cross entropy loss for the semantic analysis, binary cross entropy loss with logits for the paraphrase detection, and MSE loss for the semantic textual analysis task.

For each variation of our proportional sampling model, we ran 20 epochs with 1650 training steps (the total number of batches across the three datasets divided by 12). For each variation of our triple batch model, we ran 10 epochs with 755 training steps (the number of batches in the smallest training dataset). The proportional sampling model and triple batch model took roughly 4 hours and 2 hours, respectively, to train on an NVIDIA T4 GPU using these hyper-parameters.

5.4 Results

Table 1 shows the results for the original BERT model trained on the SST and CFIMDB datasets. The accuracy aligns with the baseline accuracy provided in the handout.

Table 1: Original BERT Sentiment Analysis

Model	SST score	CFIMDB score
minBERT baseline	0.404	0.808
minBERT finetune	0.530	0.963

Table 2 displays the test scores achieved by our highest performing proportional sampling and triple batch models.

Table 2: Best Multitask Test Scores

Model	SST score	PARA score	STS score
Prop. Sampling	0.523	0.767	0.352
Triple Batch	0.518	0.715	0.679

Surprisingly, the final variation of our triple batch model outperformed the proportional sampling model counterpart. Given the lost data stemming from our truncation technique, we suspected that the reverse would occur. However, gradient surgery seemed to be the differentiating factor allowing the STS score of the triple batch model to exceed that of the proportional sampling model.

Figure 1 below displays the dev set performance of three variations of the triple batch model on the three subject NLP tasks.

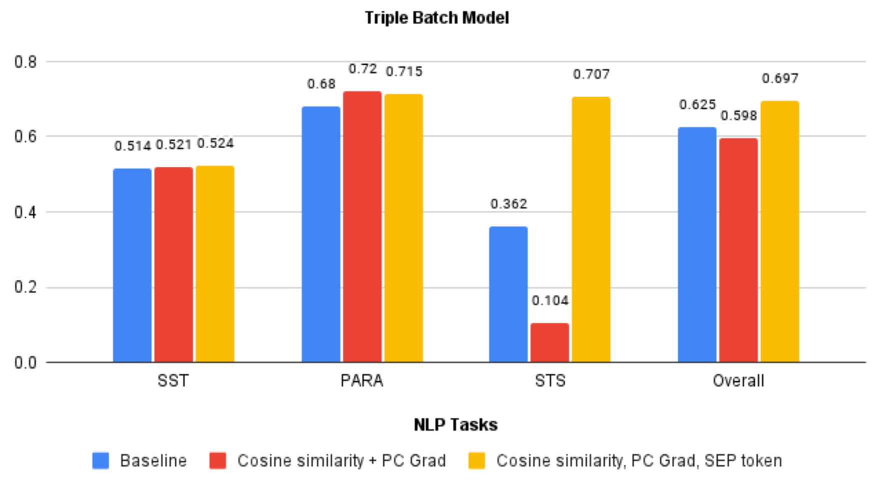

Figure 1: We observe notable improvement in triple batch model overall performance from the baseline to the final variation. Although it was surprising to see the poor STS task performance in the second variation, the improvement from baseline to end show that the extensions helped generate more robust embeddings.

Figure 2 below displays the dev set performance of three variations of the proportional sampling model on the three subject NLP tasks.

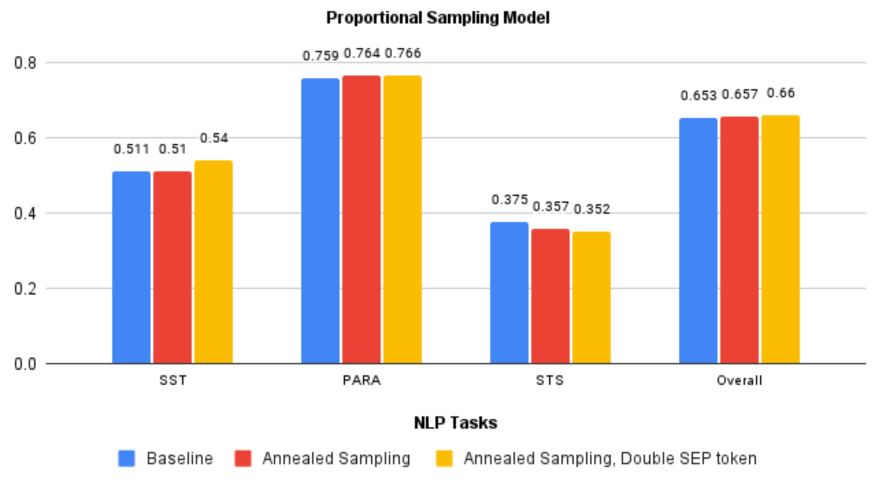

Figure 2: In contrast to the marked improvement displayed by our triple batch model across its variations, our extensions to the proportional sampling model - namely, augmenting the baseline with annealed sampling and then adding a separator token for both the textual comparison tasks - did not yield the same progression of model improvements. We found it strange that both the annealed sampling extension and two separator tokens failed to improve the model performance significantly.

6 Analysis

6.1 Proportional Sampling Evaluation

Strangely, the proportional sampling model's performance on the semantic textual analysis task barely improved after implementing annealed sampling. Instead of balancing scores and reducing over-performance on tasks with larger datasets, we saw a continuation of the baseline pattern. This could be attributed to the annealing process not being aggressive enough, as it shifts towards a uniform sampling distribution too late to effectively increase representation of smaller datasets and generalize embeddings. Additionally, it is possible that our model did not run enough epochs for the annealed sampling extension to fully take effect. Alternatively, the lack of improvement could be explained by different task difficulties. The semantic textual analysis task that classifies similarity using a 0-5 range is more complex than the paraphrase detection's binary outcomes. Thus, the pairing of less examples and a more complex task might be the reason for disparate task performances.

Adding the two separator tokens in the textual comparison tasks, surprisingly, did not yield significant improvements either. This stagnation could be attributed to the model not inherently benefiting from a clear demarcation between segments. The model's attention mechanism likely already learned to distinguish the different parts of the input text without the explicit separators.

6.2 Triple Batch Evaluation

Our triple batch model performance fell when cosine similarity and gradient surgery extensions were introduced. Without cosine similarity fine-tuning, we concatenate the two input embeddings and pass them through a dense layer to produce a single logit. Using cosine similarity affects gradient flow during backpropogation and the downstream learning by eliminating the direct path for gradients from the loss to the embeddings - which may interfere with the model's learning. Performance improved when we introduced a separator token between the concatenated embeddings in the paraphrase detection task. The token helped the model's attention mechanism better weigh the importance of different parts of the input data, serving as a boundary between the input texts.

Overall, we were surprised by the relative success of the triple batch model. We suspect that our gradient surgery approach helped reduce the model overfitting in the proportional sampling approach by preventing conflicting gradients from interfering with multiple task learning.

6.3 Error Analysis on Triple Batch Model (highest performing variation)

6.3.1 Sentiment Analysis Error

Figure 3 below displays the proportion of times that the model classifies sentences as being more positive or negative than they actually are using the SST dataset.

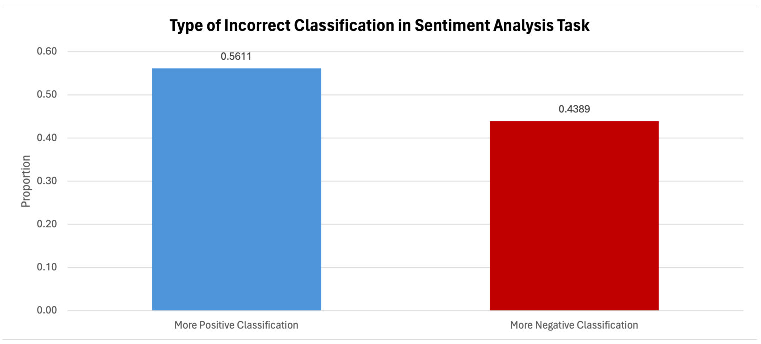

Figure 3: When incorrect, our model tends to classify sentences as being more positive than they actually are. This could be attributed to positive sentiment being expressed more directly than negative sentiment or positive words occurring more often in our training set. To address this, we can populate our training dataset with sentences that have more "negative" words or use a custom loss function that penalizes incorrect classification of underrepresented classes more severely than other mistakes.

6.3.2 Paraphrase Detection Error

Figure 4 leverages the Quora dataset to split the errors we observe as being false positive or negative.

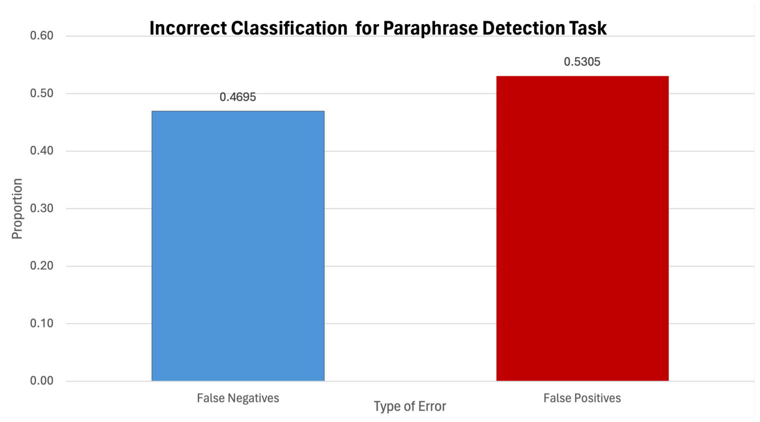

Figure 4: We see that the proportion of false positives compared to false negatives are roughly similar, with our model yielding false negatives 47% of the time and false positives 53% of the time. This balance suggests that the model is equally sensitive to both types of errors. Given that our loss function (BCE) treats the false positives and false negatives symmetrically and there is no indication that our dataset or task weights are skewed one way or another it makes sense that we observe a relative balance between these types of errors. This could indicate that our model has achieved a reasonable level of generalization for this task but struggles with attaining a more nuanced understanding of text which is needed in this task due to the inherent complexity of paraphrasing.

6.3.3 Semantic Textual Similarity Analysis Error

Figure 5 below displays the model's over- and under-prediction rate when performing the STS task.

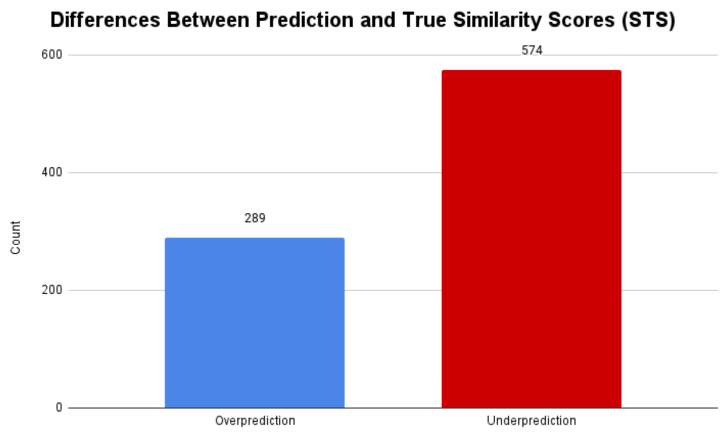

Figure 5: We observe a substantial disparity between under- and over-predicted scores, with the model under-predicting scores 66% of the time. Given that MSE loss does not penalize negative or positive errors differently, this prediction disparity could be attributed to the model's attention focus, which seemingly focuses too much on differences between embedding pairs rather than commonalities. This leads to to consistent under-predictions.

6.3.4 Disparity in Results Between Approachs for STS Task Error Analysis

We will focus on the following two examples displayed in Tables 3 and 4 to analyze why the triple batch approach significantly outperformed the proportional sampling approach in semantic textual analysis. Although these are just two instances out of thousands, a thorough analysis of each output file leads us to believe that these highlighted examples can be generalized to the discrepancies in our results.

Table 3: Example 1: Comparison of Approaches for STS Task

Example 1	**Description/Score**
Sentence 1	Some guy sitting on a couch watching television.
Sentence 2	A guy is sitting on the couch watching TV.
Truncated Triple-Batch Approach Score	4.9
Proportional Sampling Approach Score	2.0
True Score	5

Table 4: Example 2: Comparison of Approaches for STS Task

Example 2	**Description/Score**
Sentence 1	A man is speaking on a stage.
Sentence 2	A man is speaking at a podium.
Truncated Triple-Batch Approach Score	3.9
Proportional Sampling Approach Score	1.0
True Score	3.4

It is possible that the cosine similarity layer in the triple-batch approach contributed to the more accurate measure of textual similarity displayed by the embeddings, since it explicitly trains the model to optimize for an objective that mirrors the task's goal. However, the drop-off in performance from triple batch baseline to triple batch with cosine similarity and gradient surgery casts doubt on this hypothesis.

More likely, the addition of a SEP token in the paraphrase task helped the model better understand how to differentiate between comparison input texts. Paired with the gradient surgery, the model was able to balance learning and avoid over-fitting on one task. The proportional sampling approach likely experienced poorer performance on the textual similarity task because it lacked a formal mechanism for managing conflicting gradients.

7 Conclusion

We adapted the baseline BERT model to handle multiple tasks at once including sentiment analysis, paraphrase detection, and semantic textual analysis. We accomplished this with two different multitask training methods: a truncated triple batch approach and a proportional sampling approach. We extended the former with cosine similarity fine tuning, gradient surgery, and a separator token for paraphrase detection. We extended the latter using annealed sampling and a separator token for both textual comparison tasks. We observed that the triple batch model outperformed the proportional sampling model in overall performance, with the semantic textual similarity score serving as the biggest distinction. We attribute the more balanced scores to fine-tuning techniques directed towards guarding against model over-fitting and conflicting gradients during training.

Our models faced several limitations including: (1) losing valuable training data after truncating datasets; (2) a tendency to over-fit on the paraphrase detection task in our proportional sampling model; and (3) a cosine similarity implementation that hurt semantic textual analysis performance.

In the future, we hope to build upon this work by adjusting our triple batch approach so that it doesn't sacrifice significant amounts of training data, experiment with different hyperparameters that realize the benefits of annealed sampling, and build on our cosine similarity implementation so that it can better capture non-linear semantic relationships.

References

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing*, Online.

Asa Strickland Cooper and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, Online.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*, Online.

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