Optimizing Multitask BERT: A Study of Sampling Methods and Advanced Training Techniques

Stanford CS224N Default Project

Andrew Wu
Department of Computer Science
Stanford University
acwu02@stanford.edu

Wesley Larlarb
Symbolic Systems Program
Stanford University
wlarlarb@stanford.edu

Mentor:
Chaofei Fan

Abstract

Our research investigates the effects of various sampling methods in the multitask fine-tuning of BERT (Bidirectional Encoder Representations from Transformers). BERT has demonstrated strong performance across various natural language processing (NLP) tasks, but the influence of scheduling methods during fine-tuning on multitask learning efficiency and effectiveness remains underexplored. We compare a baseline sequential task scheduling with round-robin oversampling and undersampling methods. Additionally, we explore extensions such as multitask fine-tuning with cosine similarity loss. Our results show that round-robin oversampling yielded the best accuracies, while round-robin undersampling appeared to overall hinder model comprehension. Imbalanced sampling highlights concerns about overfitting in some tasks and underfitting in others, suggesting the need for further investigation into optimal multitask scheduling parameters. These findings have significant implications for enhancing multitask learning models in NLP applications and optimizing computational resources.

1 Introduction

BERT has emerged as a powerful tool in NLP, achieving success across a variety of tasks. Extensive research has been done on to augment BERT's ability to perform across tasks, as well as the effects of different scheduling heuristics for batch multitask training. Despite this, there remains a gap in the literature regarding the impact of how scheduling can possibly mitigate overrepresentation of certain tasks in the aggregated dataset. Understanding this is crucial for optimizing BERT's performance, as without appropriate scheduling techniques, larger datasets can dominate the training process, leading to an imbalanced model that performs well on frequent tasks but poorly on less frequent ones. Additionally, datasets often contain biases, which can be exacerbated during the fine-tuning process. If a particular dataset is overrepresented, any biases within it can therefore become more pronounced in the model's predictions. Furthermore, in the absence of dimensionality reduction techniques, training on excessively large datasets can prove computationally inefficient.

Our research aims to fill this gap by investigating the effects of various scheduling strategies on BERT's performance and generalization capabilities when fine-tuned for multiple tasks. By identifying the most effective scheduling strategies, our findings can help in developing more balanced models that perform consistently across diverse tasks, thereby enhancing the robustness and fairness of NLP applications.

2 Related Work

BERT was first introduced by Devlin et al. (2018), which implemented the bidirectional transformer architecture as outlined by Vaswani et al. (2017) Since then, significant research has been done in augmenting the model's capabilities on different downstream tasks. Bi et al. (2022) introduce

MTRec, a multi-task learning model that incorporates information from different tasks, ie. news recommendation, category classification, and named entity recognition, using multi-task fine-tuning. To mitigate gradient conflicts between these tasks, they also employed a modified version of gradient surgery as proposed by Yu et al. (2020). Their results show significant improvements over baseline scores. Reimers and Gurevych (2019) introduce Sentence-BERT, which uses cosine similarity as a metric of semantic similarity between two word embeddings. However, the application of these approaches on alternative training approaches, and their relationship, if any, with task scheduling, remains to be seen.

Jean et al. (2019) introduces an adaptive scheduling method that oversample poorer-performing tasks, designed within the context of neural machine translation. This is in contrast to common heuristics, which generally sample tasks uniformly or in proportion to their respective datasets. Meshgi et al. (2022) introduces a similarly advanced scheduling method which utilizes reinforcement learning to develop an optimal scheduling policy. Their research provides important insights into the impacts of scheduling on multi-task learning. In spite of this, they do not directly address the applications of scheduling to mitigate overrepresentation of certain tasks in the aggregated dataset.

3 Approach

Our BERT model's basic architecture, ie. multi-headed self-attention, BERT layers, and the Adam optimizer, are in line with those outlined in Vaswani et al. Here, we detail our specific implementation.

3.1 Multitasking Architecture Overview

We employ a different post-bert architecture for each downstream task. For sentiment classification, we map the BERT pooled output of hidden size (768) to 5 sentiment classes, which outputs a 5-dimensional tensor of logits. For paraphrase detection, we take in pairs of sentences, concatenate them with the [SEP] token in between as in the original Bert paper, and pass the pooled output to a linear layer to predict a binary label. For semantic textual similarity tasks, we take in pairs of sentences, combine their BERT embeddings, and pass them through the same linear layer to produce a single logit indicating the similarity. This design enables the model to effectively perform multiple tasks by leveraging shared representations from the BERT model. During the forward pass, we pass the BERT output through the ReLU activation function to capture nonlinearity, and also employ a dropout layer for regularization and to combat overfitting.

3.2 Scheduling Approaches

In all of our scheduling approaches, we randomly sample each batch from each dataset with PyTorch's built in `shuffle=True`.

3.2.1 Baseline Scheduling

Algorithm 1 Random Sequential Sampling

procedure RANDOMSEQUENTIALSAMPLING
  for $i \leftarrow 1$ to num_epochs **do**
    for sst_batch $\in$ sst_dataloader **do**
      sst_loss $\leftarrow$ CrossEntropyLoss(sst_batch)
    end for
    for para_batch $\in$ para_dataloader **do**
      para_loss $\leftarrow$ BinaryCrossEntropyLoss(para_batch)
    end for
    for semeval_batch $\in$ semeval_dataloader **do**
      semeval_loss $\leftarrow$ CosineEmbedLoss(semeval_batch)
    end for
  end for
  return
end procedure

In random sequential sampling, during each batch we iterate over each dataloader in sequential order, moving on to the next when we exhaust the current one. The order of dataloaders was chosen arbitrarily; further research could be done on the effects of dataloader order.

3.2.2 Round-Robin Undersampling And Oversampling

Algorithm 2 Round-Robin Undersampling

```
procedure RoundRobinUndersampling
    for i ← 1 to num_epochs do
        j ← 0
        while j ≤ len(shortest_dataloader) do
            sst_batch ← Sample(sst_dataloader)
            para_batch ← Sample(para_dataloader)
            semeval_batch ← Sample(semeval_dataloader)
            sst_loss ← CrossEntropyLoss(sst_batch)
            para_loss ← BinaryCrossEntropyLoss(para_batch)
            semeval_loss ← CosineEmbedLoss(semeval_batch)
            j ← j + 1
        end while
    end for
    return
end procedure
```

In round-robin undersampling, we continuously retrieve one batch per dataloader until the shortest dataloader is exhausted, at which point we stop training.

Algorithm 3 Round-Robin Oversampling

```
procedure RoundRobinOversampling
    for i ← 1 to num_epochs do
        j ← 0
        while j ≤ σ do
            sst_batch ← Sample(sst_dataloader)
            para_batch ← Sample(para_dataloader)
            semeval_batch ← Sample(semeval_dataloader)
            sst_loss ← CrossEntropyLoss(sst_batch)
            para_loss ← BinaryCrossEntropyLoss(para_batch)
            semeval_loss ← CosineEmbedLoss(semeval_batch)
            j ← j + 1
        end while
    end for
    return
end procedure
```

In round-robin oversampling, we continuously retrieve one batch per dataloader for a fixed number of iterations σ. If any dataloader is exhausted before the last iteration, we simply resample from said dataloader in subsequent iterations. Note that σ is an arbitrary constant derived from empirical experimentation. Further work needs to be done to determine the optimal value of σ.

3.3 Extensions

3.4 Multitask Fine-Tuning

We augmented fine-tuning by sharing parameters across different tasks, which were updated based on the combined loss from all tasks:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{SA} + \lambda_2 \mathcal{L}_P + \lambda_3 \mathcal{L}_{STS} \tag{1}$$

where $\mathcal{L}_{SA}$ represents the loss on sentiment analysis, $\mathcal{L}_P$ represents the loss on paraphrase detection, and $\mathcal{L}_{STS}$ represents the loss on semantic textual similarity, and λ_1, λ_2, and λ_3 are coefficients that determine how much each task should be weighed.

3.5 Sentiment Analysis

We augmented sentiment analysis by performing a forward pass on the input sentence, then passing the resulting embedding through a linear layer of size 768 x 5 to generate an output $x \in \mathbb{R}^5$, where each element x_i is a logit equivalent to the score corresponding to a particular sentiment. We categorize sentiments as follows: 0: negative, 1: somewhat negative, 2: neutral, 3: somewhat positive, and 4: positive.

During training, we calculate the cross-entropy loss on sentiment analysis, in line with what is standard for classification tasks:

$$\mathcal{L}_{ST} = \sum_{i=1}^{N} -w_{y_n} \log \left(\frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \right) \tag{2}$$

where y is the label, w is the weight, C is the number of classes, and N is the minibatch dimension.

3.6 Paraphrase Detection

We implement paraphrase detection in line with how sentence pairs are processed in the baseline BERT architecture. We concatenate the two embeddings, then pass it through a linear layer of size $768 * 2$ to produce a single logit that corresponds to whether the input sentences are equivalent to each other.

3.7 Semantic Textual Similarity

To optimize our model's performance specifically on semantic textual similarity, we used cosine-similarity fine-tuning when calculating the loss.

$$\mathcal{L}_{STS} = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases} \tag{3}$$

where:

$$\cos(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4}$$

Because cosine similarity measures a continuous angle between two Bert embeddings, it is an efficient way of capturing similarity between sentences, allowing the two to be compared via a statically sized learned weight matrix applied to both embeddings before the cosine similarity.

3.8 Gradient Surgery

[1]

We also built off of WeiChengTseng's Pytorch-PCGrad, a publically available implementation of the gradient surgery technique described in Yu et al. (2020). This implementation utilizes a custom optimizer to iterate through all parameters in the model and modify their gradients with respect to losses across multiple tasks. In particular, for gradients $\mathbf{g_i}$ and $\mathbf{g_j}$ of losses for task i and j with respect to the model parameters, we project $\mathbf{g_i}$ onto the normal plane of $\mathbf{g_j}$ like so:

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

We do this bidirectionally for all pairs of tasks i and j.

[1] Tseng, Wei-Cheng. "Pytorch-PCGrad." GitHub, 2021, github.com/WeiChengTseng/Pytorch-PCGrad.

4 Experiments

4.1 Data

For sentiment analysis, we used the Stanford Sentiment Treebank (SST) from Socher et al. (2013). The SST consists of 11,855 sentences extracted from movie reviews. Each sentence and their constituent phrases are annotated with sentiment labels on a five-class system: negative, somewhat negative, neutral, somewhat positive, and positive.

For paraphrase detection, we used the Quora dataset from Fernando and Stevenson (2009). The Quora dataset consists of over 400,000 question pairs which are annotated as either duplicate, meaning the questions have the same meaning, or non-duplicate.

For semantic textual similarity, we used the SemEval dataset from Agirre et al. (2013). The SemEval dataset consists of 8,623 sentence pairs with corresponding score from 0 to 5, with 5 being the most similar.

4.2 Evaluation method

We evaluated sentiment analysis and paraphrase detection with the accuracy of our predicted labels. We evaluated semantic textual similarity with the Pearson correlation coefficient between our predicted scores and the real score. Following each iteration of training, we compared our results with the scores derived from our sequential scheduling baseline.

4.3 Training Procedure and Hyperparameters

We ran our experiments on PyTorch, on a GCP VM instance running on NVIDIA T4 GPU. We configured our hyperparameters mostly in line with the default settings outlined in the provided code, with a batch size of 64 and a dropout probability of 0.3. We begin with 5 epochs of fine-tuning on just the downstream layers, followed by 20 epochs of fine-tuning on the full model and the downstream layers, and ending with another 5 epochs of fine-tuning on just the downstream layers. The epochs of tuning on only downstream layers at the beginning and end of training were done for the sake of time efficiency, as training proceeded about five to six times as quickly for those epochs as compared to the full model tuning, due to the lower numbers of parameter updates required, and we were still seeing measurable training loss decreases even while just finetuning on the downstream layers.

We also tailored individual learning rates for each of the downstream architectures, as we found that the sentiment-analysis-specific layers benefitted from a significantly higher learning rate (9.5e-4) as compared to the shared downstream layer and paraphrase layers (1e-4), textual similarity layers (4e-5), and the Bert model itself (2e-5). Finally, we used Pytorch's MultiStepLR to adjust the learning rate at specified epochs over the course of training, with all learning rates being decayed by a factor of 0.7 at epochs 5 and 25. These values were hand-tuned; further work could more rigorously examine the impacts of these hyperparameters via parameter search.

4.4 Gradient Surgery Ablation

After achieving our best model performance with gradient surgery enabled, we decided to perform an ablation to test its effects, changing nothing about the model or training procedure except that we reverted to a standard Adam optimizer with no modifications to update gradients. We were not able to reproduce the results described in Yu et al.; our accuracies did not decline significantly without the gradient surgery. The accuracies for both the version trained with gradient surgery and without are shown in the table below.

4.5 Results

Table 1: Model Predictions On Downstream Tasks

	Sentiment Classification	Paraphrase Detection	Semantic Textual Similarity	Average Of 1st Three Columns
Milestone Baseline	0.316	0.369	-0.009	0.225
Random Sequential	0.304	0.670	0.212	0.395
RR Undersampling	0.301	0.632	0.181	0.371
RR Oversampling	0.432	0.691	0.272	0.465
RRO + HP + CS	0.497	0.864	0.387	0.582
RRO + HP + CS + GS	0.497	0.860	0.378	0.578

[2]

We see that as predicted, the Random Sequential method shows a noticeable improvement over the Milestone Baseline, particularly in Paraphrase Detection and Semantic Textual Similarity. Similarly, the RR Undersampling method, while slightly lower in average performance compared to Random Sequential, still shows considerable improvement over the Milestone Baseline.

RR Oversampling achieves the highest average score among the baseline methods, indicating that oversampling can significantly enhance model performance across all tasks, especially in Sentiment Classification and Paraphrase Detection (0.691).

Incorporating an improved training schedule, optimized hyperparameters, and cosine similarity results in a substantial performance boost. This method achieves the highest individual scores across all tasks and an impressive average of 0.582, showcasing the effectiveness of these advanced techniques.

5 Analysis

5.1 Analysis Of Scheduling Methods

Sequential sampling provides the most comprehensive overview of the aggregated datasets by iterating over each dataset in order until each is exhausted. However, it did not achieve the highest accuracy. This suggests that the imbalanced nature of the dataset may not provide the optimal learning conditions for the model.

Round-robin undersampling reduces the number of instances of majority tasks (paraphrasing in this case, which has a much larger dataset than the other two tasks). While this can help mitigate bias towards majority tasks, it also means that valuable data is discarded. This might lead to underfitting, where the model does not fully learn the patterns in the majority tasks, resulting in slightly lower performance.

Round-robin oversampling resamples minority tasks after their datasets are exhausted, thus providing a more balanced training dataset. This method ensures that the model is able to fit better for underrepresented tasks. The improvements across all tasks suggest that better class representation allows the model to more effectively generalize. It obtained the highest score among the baseline methods, indicates that oversampling effectively addresses class imbalance and enhances model learning.

[2]RRO = round robin overfitting. HP = tuned hyperparameters and training procedure as described above. CS = cosine similarity for textual similarity. GS = gradient surgery.

5.2 Analysis Of Gradient Surgery

According to our results, incorporating gradient surgery does not significantly alter the average performance. In general, gradient surgery would only increase performance in situations where task-specific gradients drastically overshadow one anothers' magnitude, as discussed in the visualization on page 2 in Yu et al. It may be the case that for the tasks in this paper, with an appropriate learning rate balancing and using Adam as an optimizer, each of the loss functions did not have considerable conflicts in the directions and magnitudes of their gradients with respect to the parameter space of the model. For instance, the more likely two sentences are to have high textual similarity, they are more likely to also be paraphrases of each other. It is less clear whether sentiment analysis would have gradients that are in line with the other two tasks or opposing them. It seems like the BERT model and shared layer were able to embed the data in such a way that the gradients did not conflict, but more experiments would be require to rigorously understand this. To start, future research could try minimizing the ability of downstream layers to compensate for gradient conflicts by freezing them and keeping them as shared as possible between tasks, and performing gradient surgery ablation more systematically with different task permutations, examining train loss over the epochs.

5.3 Examination of Misclassifications

We proceed to examine several misclassifications to analyze how our model might have come to these erroneous conclusions.

5.3.1 Example 1: Sentiment Classification

Sentence: "Half Submarine flick, Half Ghost Story, All in one criminally neglected file"

Actual Label: Neutral

Predicted Label: Negative

The phrase "criminally neglected" might have a strong negative connotation. Hence, the model may have focused on these words and inferred a strongly negative sentiment. This also indicates a lack of contextual understanding to recognize the true sentiment of the phrase. Furthermore, the sentence has a compound structure, which might have confused the model, leading it to weigh the negative-sounding part more heavily.

5.3.2 Example 2: Paraphrase Detection

:

Sentence 1: "Who will be the next President of the United States?"

Sentence 2: "Who should be the next president, and why?"

Actual Label: Not Paraphrase

Predicted Label: Paraphrase

Both sentences have similar structures and contain many of the same words, eg. "Who," "the next president". The model might have put too much emphasis on these surface-level similarities and concluded that the sentences are paraphrases. This also indicates a deeper understanding of the difference between "will be" (factual, future prediction) and "should be" (opinion-based, involving reasoning). This subtle difference changes the meaning significantly but can be missed if the model doesn't capture the context well.

5.3.3 Example 3: Semantic Textual Similarity

:

Sentence 1: "UN chief welcomes peaceful presidential elections in Guinea"

Sentence 2: "UN chief condemns attack against peacekeepers in Mali"

Actual Correlation: 1.0

Predicted Label: 4.326

The first sentence has a positive sentiment, while the second sentence has a negative sentiment. The model might be heavily influenced by the contrasting sentiments and fail to recognize other aspects of correlation. Furthermore, they discuss events in different countries (Guinea vs. Mali) and different topics (elections vs. attacks). The model might not effectively capture the underlying theme that both sentences are related to the actions and statements of the UN chief, leading to an incorrect correlation score.

6 Conclusion

This paper investigated the impact of various sampling methods and advanced optimization techniques on the performance of multitask BERT across three key downstream tasks: Sentiment Classification, Paraphrase Detection, and Semantic Textual Similarity. We show that data sampling and training optimization significantly influence model performance. Sequential sampling appeared to produce low accuracy by virtue of task imbalances, limiting its effectiveness. Round-robin undersampling balanced the dataset but led to underfitting due to the loss of valuable data. Conversely, round-robin oversampling emerged as the most effective baseline method, substantially improving performance by enriching the training dataset with minority task instances. Adding hyperparameter tuning, multitask fine-tuning, and cosine similarity yielded the highest overall accuracy. Although gradient surgery, intended to manage task-specific gradient conflicts, did not significantly alter performance, it highlighted a potential plateau when other optimizations are already in place.

However, our research has several limitations. Most importantly, time constraints meant we could not determine the optimal values of some hyperparameters of our models, for example the number of iterations during round-robin oversampling and the number of epochs. In addition, the datasets used may not fully represent the complexity of real-world data, potentially limiting the generalizability of our findings. Furthermore, we did not attempt to fine-tune our model on other datasets, which may have also resulted in improvements. Additionally, the comparatively low score we achieved on STS means that there is still significant work that can be done in improving our model in this regard.

Future research should address these limitations by incorporating more diverse and complex datasets, expanding the range of tasks studied, and systematically evaluating the effects of different preprocessing techniques and hyperparameter configurations. Additionally, to prove the effectiveness of cosine similarity methods over linear layers for textual similarity, ablations should be done with all other factors being held constant; we had planned to undergo these but ran out of time, making it hard to tease out the effects of cosine similarity from the hyperparameter tuning.

Future research should explore hybrid sampling methods, systematically evaluate gradient surgery under various conditions, and continue fine-tuning hyperparameters of our multitask model for their optimal values. Our research contributes valuable insights into effective strategies for enhancing multitask BERT models.

7 Ethics Statement

This research project presents several ethical challenges and societal risks. Most importantly, as mentioned in our introduction, overrepresentation or underrepresentation of certain tasks can result in a model that performs well on frequent tasks but poorly on infrequent ones, thereby limiting its generalizability and fairness. This can have significant repercussions in sensitive applications such as law enforcement and healthcare. Another significant concern is the potential for data bias. If the model is trained on datasets with inherent biases, these biases may be exacerbated during fine-tuning, leading to unfair or inaccurate predictions.

The crux of this project lies in exploring methods to potentially mitigate these risks, by investigating the use of alternative scheduling algorithms, which allow us to equally sample datasets. This ensures no task gets excessively prioritized during training. By comparing and contrasting the results between them, we can come to a conclusion as to which method is best in terms of both accuracy and equal representation of tasks. Additionally, we will implement robust bias detection and mitigation strategies throughout the data preprocessing and model training stages. By regularly auditing our datasets and model outputs for bias, we aim to identify and address any disparities that may arise.

References

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Samuel Fernando and Mark Stevenson. 2009. A semantic similarity approach to paraphrase detection. *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*.

Sébastien Jean, Orhan Firat, and Melvin Johnson. 2019. Adaptive scheduling for multi-task learning. *CoRR*, abs/1909.06434.

Kourosh Meshgi, Maryam Sadat Mirzaei, and Satoshi Sekine. 2022. Q-learning scheduler for multi task learning through the use of histogram of task uncertainty. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 9–19, Dublin, Ireland. Association for Computational Linguistics.

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.

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.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.