BERT and Beyond: A Study of Multitask Learning Strategies for NLP

Stanford CS224N Default Project

Jack Le
Department of Computer Science
Stanford University
jackle@stanford.edu

Febie Lin
Department of Computer Science
Stanford University
febielin@stanford.edu

Abstract

Transfer learning has become a transformative method, allowing pre-trained language models to be extended to downstream tasks. In this paper, we apply a variety of techniques to significantly improve the ability of pre-trained language models, like BERT, to understand nuanced language features on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We observe that incorporating cross-encoding into BERT's input, adopting a staggered round-robin training approach, employing SMART to mitigate overfitting, and using a novel dual-objective loss function for fine-grained sentiment classification yields the greatest improvements on multitask learning ability. We achieve a score of 0.793 on the test set, surpassing existing baselines. At the time of writing, we hold fourth and third place on the dev and test leaderboards, respectively.

1 Introduction

In recent years, the field of natural language processing (NLP) has witnessed significant advancements, largely driven by the development of powerful pre-trained language models such as BERT (Devlin et al., 2019). However, the effective application of these models to real-world scenarios often requires fine-tuning them for specific downstream tasks, which can be a challenge.

In this paper, we present our approach to enhancing the performance of $\text{BERT}_{\text{BASE}}$ on fine-grained sentiment classification, paraphrase detection, and semantic textual similarity. We propose a multitask learning strategy that incorporates several key techniques, including cross-encoding for sentence-pair tasks, staggered fine-tuning for efficient training on datasets of varying sizes, SMART regularization, and a novel Dual-Objective Cross-Entropy loss function. Our aim is to develop a robust and efficient model that can effectively leverage the power of pre-trained BERT while addressing the specific challenges associated with multitask learning. Through experiments and analysis, we demonstrate the effectiveness of our approach and provide insights into the strengths and limitations of our model.

2 Related Work

The success of pre-trained language models like BERT (Devlin et al., 2019) has sparked a surge of research aimed at improving their fine-tuned performance on out-of-domain downstream tasks. Here we discuss select works from which we draw inspiration.

In the original BERT paper, Devlin et al. (2019) introduced the concept of cross-encoding, where input sentences are concatenated with a special separator token (`[SEP]`) before being fed into the model. This approach has been widely adopted for tasks involving sentence pairs, such as paraphrase detection and semantic textual similarity. However, an alternative approach, known as bi-encoding (Reimers and Gurevych, 2019), has also gained attention. In bi-encoding, sentences are processed independently by the model, and their embeddings are then combined for downstream tasks. This

approach has been shown to be particularly effective for similarity-based tasks, as it allows for more efficient computation of similarity between sentence embeddings.

Another challenge in fine-tuning pre-trained language models is the presence of imbalanced datasets. When training on multiple tasks with varying dataset sizes, the model may overfit on smaller datasets or underfit on larger ones. To address this issue, techniques like annealed sampling Stickland and Murray (2019) have been proposed. Annealed sampling adjusts the sampling probabilities of tasks based on their dataset size, allowing for a more balanced training process.

In addition to sampling strategies, the choice of loss function can also significantly impact model performance. For ordinal classification tasks, such as fine-grained sentiment analysis, traditional cross-entropy loss may not fully capture the ordinal nature of the labels. Castagnos et al. (2022) explored the use of ordinal log-loss (OLL), which not only encourages correct predictions but also penalizes predictions that are far from the true label. Their results demonstrate the potential of OLL for improving performance on ordinal classification tasks.

Finally, overfitting remains a major concern when aggressively fine-tuning large pre-trained models on relatively small downstream tasks. To mitigate this issue, regularization techniques like SMART (Jiang et al., 2020) have been proposed. SMART combines smoothness-inducing adversarial regularization with Bregman proximal point optimization to achieve a balance between learning task-specific patterns and maintaining generalization to unseen data.

3 Approach

Our enhancements are built on top of a $\text{BERT}_{\text{BASE}}$ model, which closely follows the architecture in Devlin et al. (2019). The model consists of an embedding layer that contains a word embedder and a position embedder, as well as 12 BERT encoder transformer layers. Each transformer layer consists of multi-head self attention as described in Vaswani et al. (2017), followed by an additive and normalization layer with a residual connection, a feed-forward layer, and another additive and normalization layer with a residual connection. For optimization, we use the Adam Optimizer with weight decay regularization.

3.1 Baseline

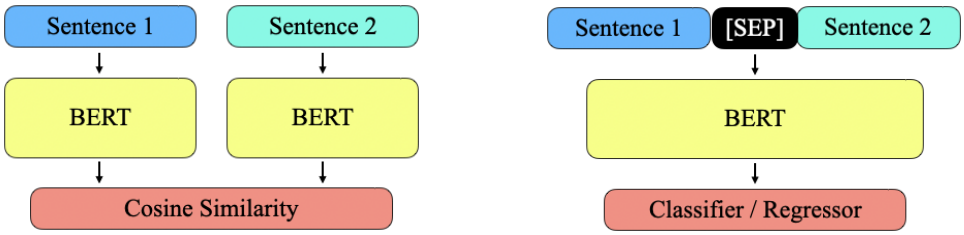
To establish a comprehensive understanding of our project's performance, we incorporated three distinct baseline comparisons. The first baseline draws from the scores reported in the foundational literature. Specifically, Devlin et al. (2019) reported an accuracy of 0.893 on the Quora Question Pairs (QQP) dataset and a Pearson Correlation of 0.876 on the SemEval STS-B dataset using the $\text{BERT}_{\text{LARGE}}$ model[1]. Notably, these results pertain to the larger variant of BERT, since no equivalent metrics were provided for the $\text{BERT}_{\text{BASE}}$ model. For sentiment classification, Munikar et al. (2019) achieved an accuracy of 0.532 on the Stanford Sentiment Treebank (SST-5) dataset using the $\text{BERT}_{\text{BASE}}$ model.

To further analyze our performance, we introduced two additional baselines, each with distinct objectives. Our first supplemental baseline employs a minimally trained $\text{BERT}_{\text{BASE}}$ model, where all original BERT parameters are frozen except for the task-specific classification and regression heads. This approach aims to discover the capability of the pre-trained BERT model on our selected downstream tasks with minimal parameter adjustments. Our second supplemental baseline consists of an ensemble of three individually fine-tuned BERT models, each dedicated to one of the tasks at hand. This is designed to showcase the potential upper-bound performance of BERT, absent any extensions or fine-tuned transfer learning mechanisms. Together, these baselines provide a method for evaluating our model's effectiveness relative to established benchmarks.

3.2 Bi-Encoding, Cross-Encoding, and Cosine Similarity

Semantic Textual Similarity and Paraphrase Detection are sentence-pair tasks, meaning they necessitate the evaluation of two input sequences to generate a prediction. This presents a challenge for the BERT model since it is traditionally limited to processing single input sequences. To handle these sentence-pair tasks, we explored two strategies: bi-encoding and cross-encoding.

[1]These results were retrieved from their official submission to the GLUE Benchmark.



(a) Bi-encoding with Cosine Similarity

(b) Cross-encoding with linear layer

Figure 1: Encoding strategies for multiple input sequences

The first approach is the bi-encoding (1a), which was proposed by Reimers and Gurevych (2019). Here, we independently process each input sequence through BERT, then combined the embeddings to produce a prediction for the sentence-pair. The combining process varied for each task: for paraphrase detection, we concatenated the embeddings and applied a linear classifier, while for semantic textual similarity, we used the cosine similarity of the two embeddings to gauge the degree of resemblance.

The second approach is cross-encoding (1b), which was used by Devlin et al. (2019) when they originally introduced BERT. Instead of individually processing each input sequence through BERT, the cross-encoding strategy begins by merging each input sequence with a special `[SEP]` token, before passing it forward through the model. It then directly applies a classification or regression layer on the output embedding to produce a prediction. While the bi-encoding strategy is more efficient for sentence similarity tasks, the cross-encoding strategy has the potential to learn stronger inter-sentence nuances since it considers both sentences in a pair simultaneously during processing, which allows it to harness a braoder context to learn dependencies between the sentences.

3.3 Loss Functions for Sentiment Classification

One of our downstream tasks is fine-grained sentiment classification using the SST-5 dataset, where the model is supposed to evaluate a sentence and predict a label of negative, somewhat negative, neutral, somewhat positive, or positive. In our experiments, we noticed that most of the errors within SST-5 were around adjacent boundaries, which is an issue since the labels of SST-5 are ordered.

Traditional multi-task classification problems use cross-entropy as its loss function. However, cross entropy does not account for the ordinal nature of the labels, since it treats the task purely as categorical with no ordinal relationship between classes. To this end, we decided to experiment with different loss functions to hopefully build stronger decision boundaries between adjacent labels.

We first considered the Ordinal Log-Loss function proposed by Castagnos et al. (2022). The loss function is defined as:

$$\mathcal{L}_{OLL}(P, y) = -\sum_{i=1}^{N} \log(1 - p_i) \cdot d(C_y, C_i)^{\alpha}$$

where P is the softmax probability distribution of the logits, $d(C_y, C_i)$ is the distance between the true label's class C_y and the predicted label's class C_i, and α is a hyperparameter. The inclusion of distance allows the loss function to better consider the ordinal nature of the labels, giving greater loss to predictions that are further away from the true labels.

To further explore loss functions, we took inspiration from the One-versus-All view of multi-class classification [2] to create an original loss function that we call "Dual-Objective Cross-Entropy," which, for the purpose of the loss function, conceptualizes the SST-5 classification task as five separate binary classification problems, one for each class, and averages their loss. The function is defined as:

$$\mathcal{L}_{Dual}(x, y) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \cdot \log \sigma(x_{ij}) + (1 - y_{ij}) \cdot \log(1 - \sigma(x_{ij}))$$

[2] Explained in the textbook Machine Learning Refined: Foundations, Algorithms, and Applications

where C is the number of classes, σ is the Sigmoid function, x_{ij} is the unnormalized logit for sample i and class j, and y_{ij} is the one-hot encoded target label for sample i and class j.

This method not only seeks to maximize the accuracy of predicting the correct class but also introduces a structured penalty for incorrect classifications. We hypothesized that this nuanced approach would be particularly suited for handling the fine-grained nature of the SST-5 dataset, since by optimizing each class boundary separately and averaging their losses, the loss function encourages the model to create more distinct and well-separated decision regions for each sentiment class, thereby reducing misclassifications between adjacent classes.

3.4 Training Strategies for Multitask Learning

Our three downstream tasks had significantly different dataset sizes, which prompted us to investigate several approaches aimed at optimizing the learning process across tasks with varying dataset sizes. Initially, we adopted a sampled round-robin strategy, iterating through batches from each dataset in turn until the smaller SST-5 and STS datasets were fully utilized. Given the substantial size difference, with the QQP dataset being notably larger, we randomly sampled batches from QQP to maintain some balance in training exposure. This means we did not use the full QQP dataset during training.

To address the limitation of the sampled strategy, we sought to leverage the full QQP dataset to improve performance on the paraphrase detection task. This led to the implementation of a max-size-cycle round-robin strategy, which stops after the longest dataset is exhausted, and cycles through the rest of the smaller datasets. This ensured continuous training and exposure across all three datasets. However, given that the QQP dataset was significantly larger than the rest, we observed that if we used the same batch sizes for all datasets, completing one cycle through the QQP dataset resulted in the SST-5 dataset being cycled through sixteen times, causing significant overfitting.

In an attempt to synchronize convergence, we experimented with a staggered fine-tuning strategy. In this strategy, we dedicated the first two epochs to training solely on the QQP dataset, based on the observation that good performance for this dataset was typically observed by the end of the second epoch. Starting from the third epoch, we resumed the sampled round-robin strategy mentioned previously, where we train across all datasets until the SST-5 and STS datasets were exhausted, while randomly sampling batches from the QQP dataset. This approach allowed us to fully utilize the extensive QQP dataset without compromising the integrity of training on the smaller datasets caused by excessive cycling, effectively balancing the training process across all tasks.

3.5 SMART Regularization

To address the challenge of overfitting, particularly evident in the SST-5 dataset, we integrated the SMART regularization technique proposed by Jiang et al. (2020). SMART incorporates two principal components: Smoothness-inducing Adversarial Regularization and Bregman Proximal Point Optimization, though our implementation primarily focuses on the former.

This approach adjusts the fine-tuning optimization process to foster model smoothness and restrain overfitting, producing better generalization to the target domain. Specifically, the optimization objective is reformulated to include a smoothness-inducing adversarial regularizer alongside the standard loss function, defined as:

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta), \tag{1}$$

where $\mathcal{L}(\theta)$ is the loss function for our downstream task, λ_s is a tuning parameter, and $\mathcal{R}(\theta)$ is the smoothness-inducing adversarial regularizer, defined as:

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{||\widetilde{x}_i - x_i||_p \leq \epsilon} l_s(f(\widetilde{x}_i; \theta), f(x_i, \theta)). \tag{2}$$

As Jiang et al. (2020) suggest in the original paper, l_s is chosen as the symmetrized KL-divergence loss for classification tasks, and as the mean-squared-error loss for regression tasks.

In our final model, SMART was applied selectively to the QQP and SST-5 classification tasks, where overfitting was particularly pronounced. The decision to exclude the STS regression task from this regularization was due to the absence of signs of overfitting. Furthermore, we diverged from Jiang et al. (2020) by opting to use the ADAMW optimizer to solve the minimization, instead of their proposed Bregman Proximal Point method.

4 Experiments

4.1 Data

We use three datasets to fine-tune our pre-trained BERT model to each of our downstream tasks. We have `train`, `dev`, and `test` splits for each dataset, which were provided by the CS 224N staff.

For **fine-grained sentiment classification**, we use the Stanford Sentiment Treebank (SST-5) dataset, which consists of sentences extracted from movie reviews, where each sentence has a label of negative, somewhat negative, neutral, somewhat positive, or positive. We have 8,544 `train` examples, 1,101 `dev` examples, and 2,210 `test` examples.

For **paraphrase detection**, we use the Quora Question Pairs (QQP) dataset, which consists of questions pairs with labels indicating whether they are paraphrases of one another. We have 141,498 `train` examples, 20,212 `dev` examples, and 40,431 `test` examples.

For **semantic textual similarity**, we use the SemEval STS Benchmark dataset, which consists of sentence pairs of varying similarity on a scale of 0 (not at all related) to 5 (equivalent meaning). We have 6,040 `train` examples, 863 `dev` examples, and 1,725 `test` examples.

4.2 Evaluation method

We use the task-specific metrics described in the default final project handout to evaluate our model. For SST-5 and QQP, we measure the model's performance using accuracy between the true and predicted labels. For STS, we use the Pearson correlation coefficient, which measures the linear correlation between the true and predicted similarity values.

4.3 Experimental details

All of our experiments used the $\text{BERT}_{\text{BASE}}$ model with the provided pre-trained weights. For our first baseline, where we finetune with all BERT layers frozen, we use a learning rate of 1e-3. For all the other models where we allow all model parameters to be trained, we follow the advice of Devlin et al. (2019) and Sun et al. (2020) in our hyperparameter selection. Unless otherwise noted, the learning rate is 2e-5 and the hidden layer dropout probability is 0.3. For batch sizes on multitask models, we performed a non-exhaustive grid search and settled on a batch size of 16 for SST-5 and QQP, and a batch size of 10 for STS. Refer to Section A.1 for more details about batch sizes. All models were trained using the AdamW optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay regularization of $\lambda = 0.01$.

SMART Regularization Using the notation introduced by Jiang et al. (2020), the hyperparameters for our SMART regularizer were $T_{\tilde{x}} = 1$, $\sigma = 1e-5$, $\epsilon = 1e-6$, and $\eta = 1e-3$. For the regularization weight λ_s, we performed a small hyperparameter grid search and found that $\lambda_s = 0.45$ gave us the best results. Refer to Section A.2 for more details about SMART.

4.4 Encoding Strategies and Cosine Similarity

Table 1: Dev performance of different encoding architectures on sentence-pair tasks

Architecture	QQP Accuracy (Dev)	STS Correlation (Dev)
Bi-Encoding	0.712 (-0.128)	0.324 (-0.549)
Mixed + Cosine Similarity	**0.840**	0.612 (-0.261)
Cross-Encoding	0.832 (-0.008)	**0.873**

Our first experiment compared the strategies used by Devlin et al. (2019) and Reimers and Gurevych (2019) on sentence-pair tasks. We tried three approaches: using a bi-encoder with a linear classification layer for both tasks, using a cross-encoder for QQP and a bi-encoder with cosine similarity for STS, and using a cross-encoder with a linear classification layer for both tasks.

Based on the results in Table 1, the introduction of cosine-similarity saw meaningful improvements over a pure bi-encoder strategy. However, the use of a cross-encoder for sentence-pair tasks signifi-

cantly improves model performance over all of the other approaches, since the model can better learn the dependencies between the two input sequences. As such, we chose to use the cross-encoder as our final model architecture for tasks with multiple input sequences.

4.5 Loss functions

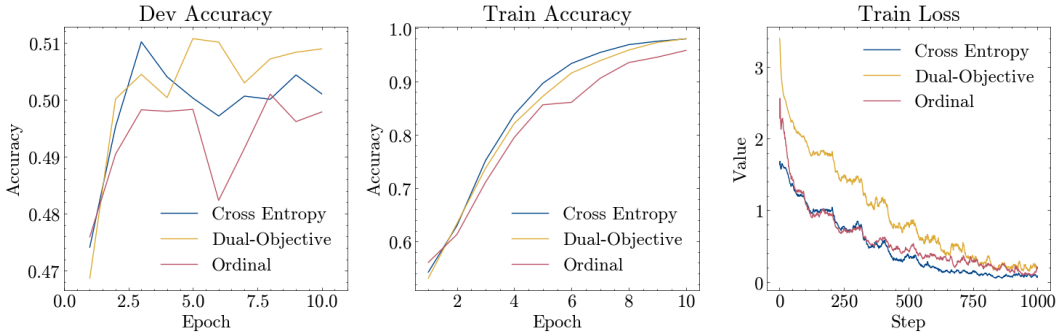


Figure 2: Effect of different loss functions on SST-5 performance

We also compared the performance of Cross Entropy, Dual-Objective Cross-Entropy, and Ordinal Log-Loss on the SST-5 dataset. Looking at Figure 2, the dev accuracy for Cross Entropy loss peaked early and then decreased, suggesting possible overfitting to the training data. In contrast, the Ordinal Log-Loss did not show signs of overfitting, but it was not as effective as the other two loss functions. The Dual-Objective Cross-Entropy loss demonstrated a stable and steady increase in development accuracy, indicating a consistent improvement in the model's generalization ability over time and aligning with our hypothesis that this method could help to construct stronger decision boundaries for fine-grained sentiment classification. Based on these results, we chose to use Dual-Objective Cross-Entropy for the SST-5 task in the remainder of our experiments.

4.6 Training Approaches

Considering the large differences in dataset sets, we wanted to identify a training approach that not only ensured good performance across all tasks but also remains computationally efficient. We compared three approaches: sampled round robin, max-size cycling, and staggered fine-tuning (SFT).

Table 2: Dev performance of the three different training approaches

Training Approach	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score	**Training Time** hh:mm
Sampled Round Robin	0.510	0.847	0.876	0.765	**00:57**
Max-Size Cycle	0.500	0.873	**0.879**	0.772	04:03
Staggered Fine-tuning (SFT)	**0.519**	**0.887**	0.876	**0.781**	01:21

Table 2 reveals that max-size cycling improves performance on QQP, though at the expense of training time and significant overfitting on the SST-5 dataset. On the other hand, SFT performs remarkably better than the other two approaches, though it still suffers from overfitting on SST-5. Importantly, SFT did not significantly increase training time over sampled round-robin, converging quickly to satisfactory performance levels after dedicating the first two epochs exclusively to the QQP dataset. Based on these results, we choose to use SFT as our final strategy.

4.7 Results

We now present our model's performance on the Dev and Test sets. Our final model incorporates cross-encoding for handling sentence-pair tasks, staggered fine-tuning, SMART regularization to mitigate overfitting, and Dual-Objective Cross-Entropy loss for stronger classification on SST-5. This approach has demonstrated superior performance and enabled us to surpass most of the baseline metrics established by prior foundational works in the field.

Table 3: Dev performance of single and multitask models on downstream tasks

Model	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score
Baseline (Devlin, Munikar)	0.532	**0.893**	0.876	0.788
Baseline Frozen Single-task	0.351	0.655	0.473	0.581
Baseline Fine-tuned Single-task	0.521	0.885	0.876	0.781
SFT + SMART	0.525	0.888	0.878	0.783
SFT + SMART + Dual	**0.540**	0.888	**0.885**	**0.790**

As seen in Table 3, our model surpasses the baseline on the SST-5 and STS tasks, while falling close behind on the QQP task. These improvements demonstrate the model's enhanced ability to discern nuanced semantic relationships and sentiments within text.

Table 4: Test performance of our best model compared to baseline

Model	**SST-5** Accuracy	**QQP** Accuracy	**STS** Correlation	**Overall** Score
Baseline (Devlin, Munikar)	0.532	**0.893** (-0.005)	0.876	0.788
SFT + SMART + Dual	**0.553** (+0.021)	0.888	**0.880** (+0.004)	**0.793** (+0.005)

Our performance on the test sets, as shown in Table 4, further solidifies our approach's effectiveness. The model exhibits a notable increase in SST-5 accuracy to 0.553, and a Pearson correlation coefficient of 0.880 on STS. In the case of QQP, our model's accuracy again slightly underperforms the baseline. We hypothesize that this may be attributed to our model's balanced focus across all tasks, rather than being optimized for a single task, as well as the fact that the baseline uses the $\text{BERT}_{\text{LARGE}}$ model. Altogether, our model attains an overall score of 0.793, surpassing the baseline score of 0.788. At the time of writing, we are third place on the test leaderboard, behind first-place by 0.003 in overall score.

5 Analysis

To gain a deeper understanding of our model's performance, we conducted a thorough qualitative evaluation, focusing on strengths, weaknesses, and key characteristics across the downstream tasks.

5.1 Sentiment Classification

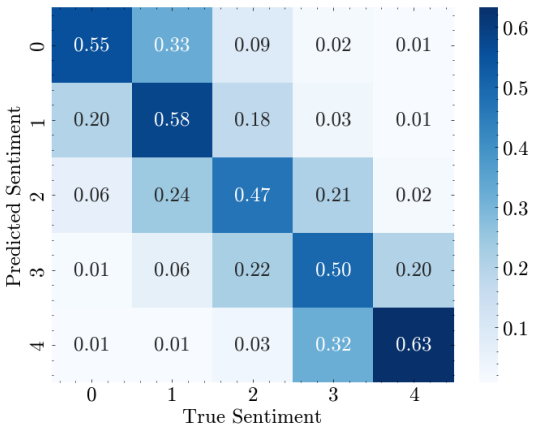


Figure 3: Normalized confusion matrix for SST-5

We visualized our model's performance on the SST-5 dataset using a confusion matrix (Figure 3). The high concentration of predictions along the diagonal indicates that our model has a solid grasp

of sentiment analysis. However, the substantial number of predictions in adjacent classes reveals the challenges in fine-grained sentiment distinction, particularly between similar sentiment levels (e.g., somewhat positive vs. positive). We believe that this may be attributed to variations in human judgement of sentiment during the labeling process, which may impact the accuracy of our model.

Interestingly, we observed instances where our model's predictions seemed more accurate than the ground truth labels. For example, the review "Makes even the claustrophobic on-board quarters seem fun." was classified as somewhat positive (3) by our model, whereas the true label was neutral (2). In this case, we favor our model's predicted class over the true label, since the review is indeed somewhat positive. This discrepancy highlights the subjectivity in human labeling and suggests that our model may even be capturing nuances that the original annotators missed.

5.2 Paraphrase Detection

Upon examining our model's performance on the QQP dataset, we found that our model struggles with sentence pairs that share strong high-level similarities but differ in specific details, often incorrectly classifying them as paraphrases of one another. For instance, the questions "What are the best and profitable ways for saving money?" and "What are your best ways to save money?" are misclassified as paraphrases of one another when they are not. By observation, they have similar general intent but differ in the emphasis on profitability and target (e.g. "your best" vs. "the best"), which the model fails to pick up on. This example and many others highlight our model's limitation in capturing specific semantic differences when high-level similarity is present. We think that for future research, it might be worthwhile to explore more sophisticated attention mechanisms that can better distinguish nuanced details for paraphrase detection.

5.3 Semantic Textual Similarity

To analyze our model's performance on the STS benchmark, we visualized the relationship between predicted and target similarity scores using a scatter plot (Figure 4 in the appendix). This analysis revealed a generally strong correlation; however, it also revealed outlier cases where our model inaccurately assessed the similarity of sentences with substantial word overlap but divergent meanings.

For example, the sentences "I'm going to shout for justice for Trayvon." and "The people shouting for justice for Trayvon are all hypocrites." share many words but convey significantly different meanings. However, our model predicted a high similarity score of 3.92, compared to the ground truth of 1.6. This suggests that our model relies heavily on surface-level word matching, rather than deeper semantic understanding. It would be interesting to further research different model architectures to increase learned deep semantic meanings beyond surface-level word matching.

6 Conclusion

In this paper, we presented a multitask learning strategy for enhancing the performance of BERT on downstream tasks. We experimented with extensions encompassing a range of techniques, including bi-encoding with cosine similarity, cross-encoding, staggered fine-tuning, SMART regularization, and novel loss functions. Through extensive experiments, we demonstrated the effectiveness of our proposed strategy, presenting a heavily extended version of $\text{BERT}_{\text{BASE}}$ that surpasses all baseline models on the SST-5 and STS tasks, while coming close on the QQP task. Our qualitative analysis shed light on the strengths and limitations of our model, highlighting its ability to capture overall sentiment, paraphrase relationships, and semantic similarity, while also identifying challenges in handling fine-grained distinctions and nuances. We hope that our work will inspire further research into the development of more robust and versatile multitask learning models.

7 Key Information to include

Our mentor is Annabelle Wang. We had no external collaborators, and we are not sharing this project.

Contributions: Jack owned develop of the cross-encoder and loss functions. Febie owned development of the bi-encoder and cosine similarity. We contributed equally to the default $\text{BERT}_{\text{BASE}}$ model, training strategies, SMART regularization, design choices, and the writing of the paper.

References

François Castagnos, Martin Mihelich, and Charles Dognin. 2022. A simple log-based loss function for ordinal text classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4604–4609, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

A Appendix

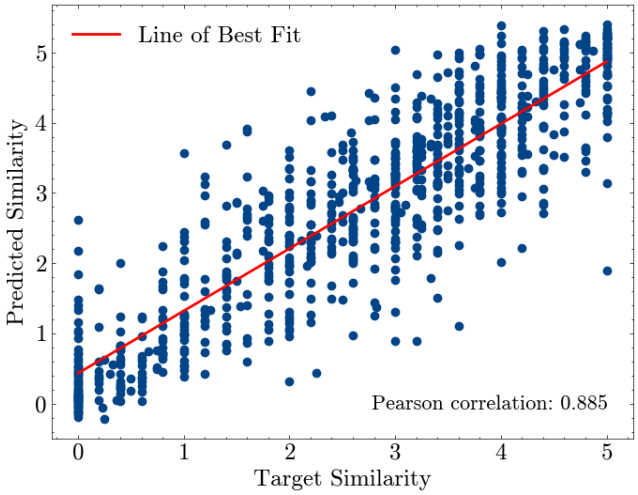


Figure 4: Scatter plot of predicted and target similarities for STS

A.1 Batch Size Selection

To determine the optimal batch size configuration for our multitask learning model, we conducted experiments with various batch sizes and evaluated their impact on the model's performance. All experiments used a SMART regularization weight of $\lambda_s = 0.45$ and a learning rate of 2e-5. The batch size is denoted by [SST]-[QQP]-[STS] in the first column of Table 5.

Table 5: Performance comparison of different batch size configurations

Batch Size	**SST-5 (Dev)** Accuracy	**QQP (Dev)** Accuracy	**STS (Dev)** Correlation	**Overall (Dev)** Score	**Epoch of Max Dev**
32-16-32	0.521	0.888	0.878	0.783	15
8-16-8	0.533	0.888	**0.892**	0.789	3
16-16-16	0.530	**0.894**	0.886	0.789	8
16-16-10	**0.540**	0.888	0.885	**0.790**	5

The 32-16-32 configuration led to a slow increase in scores, indicating that larger batch sizes may not be optimal for the SST and STS tasks. The 8-16-8 configuration performed well on the SST and STS tasks, achieving the highest STS score, but it exhibited overfitting after just 3 epochs, suggesting that the model may be learning task-specific patterns too quickly. The 16-16-16 configuration allowed the QQP task to converge, achieving the highest QQP score, but left room for improvement on the other tasks. Finally, the 16-16-10 configuration achieved the best performance on the SST task, while still achieving balanced performance on the other two tasks and the highest overall score. Consequently, we utilized this batch size configuration for our test submission.

A.2 SMART Regularization Weight Selection

To determine the optimal regularization weight for the SMART technique, we conducted a hyperparameter grid search, evaluating the model's performance on the multitask setting for each weight value. We fixed the batch size to 16 to accommodate the memory requirements of SMART regularization. All experiments employed the staggered fine-tuning approach and were trained for 10 epochs.

We utilized the `smart_pytorch`[3] module to incorporate SMART regularization into our model. However, the inclusion of SMART led to a more than two-fold increase in training time for each task,

[3] `https://github.com/archinetai/smart-pytorch`

due to the calculation of both normal and perturbed states during the regularization step. Consequently, we chose to apply SMART regularization only to the SST-5 and QQP tasks, as the STS task did not exhibit signs of overfitting.

Table 6: Performance comparison of different SMART regularization weights

SMART Weight	**SST-5** Dev	**SST-5** Train	**QQP** Dev	**STS** Dev	**Overall** Dev	**Epoch of Max Dev**
0.0	0.519	0.952	0.887	0.876	0.781	8
0.3	0.533	0.852	0.888	**0.892**	0.789	3
0.45	**0.540**	0.874	0.888	0.885	**0.790**	5
0.5	0.535	0.818	**0.890**	0.878	0.788	5
0.7	0.523	0.990	0.886	0.885	0.784	6
1.0	**0.540**	0.921	0.885	0.882	0.789	9
3.0	0.539	0.994	0.883	0.889	0.789	13

The results in Table 6 provide insights into the impact of different SMART regularization weights on the model's performance in the multitask setting. Without SMART regularization (weight = 0.0), the model achieved an overall dev score of 0.781, with the SST-5 task showing signs of overfitting, as evidenced by the very high training accuracy compared to the dev accuracy.

As the SMART regularization weight increased, we observed improvements in the model's performance. A weight of 0.3 led to a higher overall dev score of 0.789, with notable improvements in the SST-5 and STS tasks. The model reached its maximum dev performance at epoch 3, indicating fast convergence.

Weights of 0.45 and 1.0 both achieved the highest SST-5 dev accuracy of 0.540, with 0.45 resulting in the highest overall dev score of 0.790 at epoch 5. This suggests that a SMART regularization weight of 0.45 strikes a good balance between improving the model's performance and maintaining computational efficiency, since a weight of 1.0 required 9 epochs to reach this performance.

Interestingly, a weight of 0.5 yielded the highest QQP dev accuracy of 0.890, indicating that different tasks may benefit from slightly different regularization strengths. However, the overall dev score for this weight was slightly lower at 0.788.

Higher SMART regularization weights, such as 0.7 and 3.0, led to increased training accuracies for the SST-5 task (0.990 and 0.994, respectively) but did not translate to improvements in dev accuracies. This suggests that excessively high regularization weights may hinder the model's ability to generalize well to unseen data.

Based on these findings, we selected a SMART regularization weight of 0.45 for our final model, as it achieved the highest overall dev score and demonstrated good performance across all three tasks while maintaining computational efficiency.