swissBERT: A Ready-to-Use Multitask Transformer

Stanford CS224N Default Project

Yixin Liu*
Stanford University
yixinliu@stanford.edu

Tom Shen*
Stanford University
tomshen@stanford.edu

Violet Yao*
Stanford University
vyao@stanford.edu

Abstract

In this paper, we propose swissBERT, a multi-task learning approach that builds on BERT and incorporates several extensions, including direct encoding of sentence pairs, automatic weighted loss, cosine similarity loss, gradient surgery, biased task sampling, and weight sharing. We first demonstrate the effectiveness of minBERT in transfer learning for downstream tasks on sentiment analysis datasets through individual fine-tuning. We then extend minBERT to simultaneously perform three different downstream tasks using multitask fine-tuning, including sentiment analysis, paraphrase detection, and semantic textual similarity. Our experiments show that direct encoding of sentence pairs and auto weighted loss are highly effective extensions that consistently improve the model's performance across tasks. Our experiments demonstrate an overall dev accuracy of **0.754** and an overall test accuracy of **0.758** on the leaderboard.

1 Key Information to include

- Mentor: Manasi Sharma
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Pre-training and fine-tuning of large language models, such as BERT [1], have proven to be effective for various Natural Language Processing (NLP) tasks. However, these models have limitations, including the need for task-specific fine-tuning and the inability to effectively perform multiple tasks simultaneously. Multi-task learning has emerged as a promising approach to overcome these limitations, by enabling models to jointly learn from multiple tasks with potentially conflicting objectives without sacrificing performance. In this paper, we propose swissBERT, a multi-task learning approach that builds on the effectiveness of BERT and incorporates several extensions, including cosine similarity loss by Reimers et al.[2], gradient surgery by Yu et al. [3], automatic weighted loss by Liebel et al.[4], layer sharing, direct encoding of sentence pairs similar to the method used by the Next Sentence Prediction (NSP) pretraining task in BERT [1].

This report is organized as follows. In Section 3, we review related work on pre-training, fine-tuning, and multi-task learning for NLP tasks, highlighting their limitations and inspirations. In Section 4, we present the details of our approach, including the modifications to BERT and the extensions we introduced. In Section 5, we discuss our experimental setup, including the datasets and evaluation metrics. Our model achieves an overall dev accuracy of 0.754 and an overall test accuracy of 0.758 on the leaderboard. We then present a qualitative analysis of our model in Section 6, demonstrating the effectiveness of swissBERT on sentiment analysis task (SST), paraphrase detection task (Para), and semantic text classification task (STS), while also identifying potential areas for improvement. Finally, in Section 7, we conclude by summarizing our contributions and discussing future directions for research.

3 Related Work

The authors of the BERT paper [1] introduced a novel approach for pretraining language models, which has since become a reliable backbone of many downstream NLP tasks. Our work builds on this approach by leveraging the effectiveness of BERT in transfer learning for downstream tasks. One key contribution of BERT that inspired `swissBERT` was its unambiguous representation of a pair of sentences in a single token sequence, which we utilized when implementing the prediction head for the paraphrase detection and semantic text similarity tasks. Unlike BERT, our multitask model does not require fine-tuning for each downstream task separately, making it more efficient and effective for simultaneously performing multiple tasks.

Several prior works have explored transfer learning and multi-task learning in various domains, including computer vision. Yu's Gradient Surgery approach [3] performed multi-task learning on computer vision tasks and introduced a technique to resolve conflicts in gradients, which we have adopted as an extension in our work. Similarly, Liebel's work on Auxiliary Tasks in Multi-task Learning [4] also performed multi-task learning on computer vision tasks and introduced automatic weighted loss for different tasks, which we have adopted in our final model. In the field of NLP, Reimers' Sentence-BERT model [2] utilized BERT to detect the similarity of two sentences, which is also a task supported by our model. We used the Sentence-BERT model without cosine-similarity as our baseline, as it is easy to implement. However, Sentence-BERT has limitations in that the words in one sentence cannot attend to another sentence, which limits its performance. Additionally, it is not a multi-task model, unlike our approach which simultaneously performs multiple NLP tasks.

4 Approach

In this project, we aim to build a multitask BERT model that can simultaneously perform sentiment analysis, paraphrase detection, and semantic text similarity.

4.1 Baseline

BERT. To accomplish this, we first build and train a BERT model using the default architecture described in the BERT paper [1], which includes self-attention, multi-head attention, and transformer encoder layers. We add a prediction head for sentiment analysis, which includes a linear layer and sigmoid output. This concludes part 1 of the default final project handout.

Multitask BERT. For multitask BERT, each of the three tasks has its own prediction head, but they all share the same BERT parameters. We adapt the multitask training pipeline described in Peng et al. [5]. During training, we create a heterogeneous dataset by merging minibatches of each task. Then, for each iteration, we sample each task without replacement, selecting the corresponding prediction head based on the task of the current minibatch, computing the loss, and updating the model. To ensure that each task is considered roughly the same number of times during training, we implement early stopping for each epoch if all examples from any one task have been traversed. We take the pooled output of BERT embeddings for each task and feed them to its prediction head.

For paraphrase detection and semantic text similarity, we concatenate the BERT embeddings for each sentence in the sentence pair and feed them to a linear layer. The paraphrase detection task uses a sigmoid output, while the semantic text similarity task outputs logits that we convert to a scale from 0 to 5. For sentiment analysis, we use a dropout layer, a linear layer, and a softmax layer for the prediction head.

Multitask BERT Baseline. To establish a baseline for our multitask BERT model, we freeze the BERT backbone weights and only update the parameters of the prediction head. By doing so, we want to determine how much performance gain we could achieve by fine-tuning the entire BERT model rather than simply treating the BERT embeddings as an input feature. This allows us to gauge the effectiveness of our multitask learning approach and determine the potential benefit of updating the BERT weights during training.

4.2 Extensions

We explored several non-trivial extensions to our baseline, each with its own benefits and drawbacks.

Direct Encoding of a Sentence Pair (Pair Encoding). To encode the information from two sentences in a single token sequence, we adopt a similar approach as the NSP task of the BERT paper [1]. Specifically, we concatenate the two sentences in the pair with a special `[SEP]` token, and assign a unique token type ID to each sentence. We modify the starter code to allow our BERT model to take token IDs as input and directly encode a sentence pair. This approach enables the BERT model to effectively capture the interactions between the words in both sentences, which can improve the performance of downstream tasks such as paraphrase detection and semantic text similarity. Additionally, this method reduces the computational overhead of fine-tuning BERT, as it allows us to run BERT once for each sentence pair rather than once for each sentence separately. After encoding the sentence pair, we pass the pooled BERT embedding through a linear layer followed by an appropriate output layer (either softmax or identity) for each task. We find that this approach significantly improves the performance of our model on all three tasks.

Automatic Weighted Loss. In the Multitask BERT baseline, we backpropagate loss per task batch. However, the losses for each task may vary in scale and may require adjustments in relative weighting during training. Manually tuning these weights is an expensive process. To address this limitation, we experiment with a linear weighted sum of losses for each task, where task weights are made learnable. However, this approach poses an issue because the model may trivially decrease the loss by decreasing all task weights. In practice, it would be less extreme than converging to zero since the learning rate is small, and we are not training for long epochs. We then implement multitask learning with *homoscedastic uncertainty*, which refers to a task-dependent uncertainty with respect to information that our data cannot explain and varies between tasks. Formally, *homoscedastic uncertainty* loss is given by $\sum_i \frac{1}{\sigma_i^2} Loss_i + \log \prod_i \sigma_i$, where i denotes each task, and σ_i represents the noise parameter for the target variable y_i. As σ_i increases, the weight $\frac{1}{\sigma_i^2}$ for task i decreases, so it can be seen as learning the relative confidence between tasks. The objective also depends on the task's representation or unit of measure, as $\frac{1}{\sigma_i^2}$ is scaled by $Loss_i$. Regularization of σ_i by the term $\log \prod_i \sigma_i$ prevents the noise from increasing too much. This objective is well-formed, ensuring that task weights will not converge to zero. In practice, to avoid numerical instability, we directly model the log variance $log(\sigma_i^2)$ instead of σ_i. We further experiment with automatic weighted loss [4], which improves *homoscedastic uncertainty* by preventing the loss from becoming negative during training. Formally, automatic weighted loss is given by $\sum_i \frac{1}{2\sigma_i^2} Loss_i + \log(1 + \sigma_i^2)$. To account for potential differences in the optimal learning rates of task weight parameters and model parameters during training, we introduce the option for learnable task weights to have a distinct learning rate from model parameters. We find that a larger learning rate for task weights than model parameters is more effective in practice. This approach results in consistent enhancements in performance across tasks.

Gradient Surgery During the training process, the direction of gradients for each task may be different, introducing instability in training. In this approach, we experimented with the Gradient Surgery technique[3], which projects the gradient g_i of each task i onto the normal plane of the gradients g_j of other conflicting tasks j, using the formula $g_i = g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} g_j$. This extension can be combined with any other extensions we implemented.

Biased Task Sampling. In our initial approach, for each gradient update, we obtain one batch from each task, resulting in a total of one batch for SST, one for Para, and one for STS. However, we observed that some tasks are more challenging to train. For instance, the training error for the Para task is relatively high, indicating underfitting. To address this issue, we design a biased task sampling method. For each gradient update, we randomly sample tasks according to a biased distribution, assigning greater weight to more difficult tasks. For example, it is possible to have two batches of the Para task and one batch of the SST task in a single gradient update. Despite these adjustments, our experimental results show that the method is not highly effective and leads to a trade-off in the performance of tasks with smaller weights.

Cosine-Similarity Fine-Tuning. First introduced in SentenceBERT [2], this alternative prediction head computes the cosine similarity between sentence pair embeddings for the paraphrase detection and semantic text similarity tasks. Note that this extension only works if we feed each sentence to a BERT model separately, meaning we do not tokenize a sentence pair directly. Instead of concatenating the embeddings and feeding them to a linear layer, we calculate the cosine similarity between them and utilize `CosineEmbeddingLoss`. This approach enables us to better capture the semantic relationship between sentences in these specific tasks, leading to improved model performance.

Weight Sharing. In this original approach, we let each prediction head have two linear layers with a Leaky ReLU instead of one layer. Then, we share the weights of the first linear layer for the paraphrase and semantic text similarity tasks.

In our investigation, we have explored numerous extensions for enhancing the performance of our multi-task learning model. However, in our final model for the leaderboard, we have only incorporated *direct encoding of sentence pairs* and *automatic weighted loss* as key components. An architecture diagram detailing these specific features is provided in the appendix, as shown in Figure 5. These selected extensions have proven to be most effective in improving the model's overall performance across the various NLP tasks under consideration.

5 Experiments

5.1 Data

In part 1, we perform movie review sentiment classification on the SST[6] and CFIMDB datasets. SST has five categories, with the somewhat negative and somewhat positive categories having a larger number of training examples. CFIMDB is a binary classification task with a balanced label distribution. For Part 2, we train a multitask `swissBERT` model on the SST, Quora[1], and STS[7] datasets. Label distribution is presented in Figure 4 in Appendix. Quora is a binary classification task for paraphrase detection, while STS is a semantic textual similarity task. Since Quora has significantly more training examples than SST and STS, the model may become biased towards learning paraphrase detection if we traverse all the training examples during multitask training.

5.2 Evaluation method

We employ different loss functions and evaluation metrics for each specific task. For the classification problems, such as Sentiment Analysis and Paraphrase Detection, we use cross-entropy loss during training and accuracy as the evaluation metric. An exception is made for the *cosine similarity* extension of the paraphrase detection task, where we compute the `CosineEmbeddingLoss` between the embeddings of the two sentences output by the BERT backbone. For the regression problem of Semantic Textual Similarity task, we use Mean Squared Error as the loss function and Pearson correlation as the evaluation metric. We evaluate the overall performance by averaging the evaluation metric of three tasks to calculate the overall dev score.

5.3 Experimental details

For part 1 sentiment classification on CFIMDB and SST dataset, we conducted two experiments using *pretrain* and *finetune* options. In the first experiment minBERT (*pretrain*), we froze the loaded BERT parameters during training. The model was trained for 10 epochs with a learning rate of 1e-3, batch size of 64, and dropout probability of 0.3. In the second experiment, we used the *finetune* option, enabling the updating of all parameters in the model. The model was trained for 10 epochs with a learning rate of 1e-5, batch size of 64, and dropout probability of 0.3. Model performance is reported in Table 1.

To evaluate the Multitask-BERT and its extensions, we performed twelve experiments with different model settings: Baseline Multitask-BERT (*pretrain*), Multitask-BERT with no extensions (*finetune*), Multitask-BERT with different combinations of extensions, including *layer sharing*, *cosine similarity*, *pair encoding*, *learnable weights*, *homoscedastic uncertainty*, *automatic weighted loss*, *gradient surgery*, *sample task*. All experiments shared the same training configuration of 20 training epochs, batch size of 32, learning rate of 1e-5, dropout probability of 0.3, and weight decay of 0. For the model with *sample task* extension, the weight is set to $sst = 1, para = 1.7, sts = 1$. All models used the *finetune* option except the baseline model which used *pretrain* option. The performance on the dev set is presented in Table 2.

After the experiments with different extensions, we selected the *pair encoding* extension and *automatic weighted loss* extension since their combination achieves the best overall score on the dev set. To further improve the model performance, we conduct hyperparameter tuning to search for the optimal

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

learning rate, dropout probability, weight decay, and automatic weighted loss learning rate (AWL LR). We perform 7 experiments with different hyperparameters. All experiments shared the same training configuration of 20 training epochs and a batch size of 32. All models used the *finetune* option and *pair encoding* and *automatic weighted loss* extensions. The performance on the dev set is presented in Table 3.

5.4 Results

The performance of minBERT on Sentiment Classification is presented in Table 1, and it is comparable to the accuracy reported in the project handout.

Table 1: minBERT Dev Accuracy for Sentiment Classification.

	SST Acc	CFIMDB Acc
minBERT(*pretrain*)	0.391	0.788
minBERT(*finetune*)	0.518	0.967

Table 2 shows the performance of Multitask `swissBERT` with different extensions, where all models using the *finetune* option outperform the baseline model using *pretrain* parameters. By allowing the updating of parameters in both the prediction head and BERT, the fine-tuned models are better adapted to the three tasks, leading to better performance compared to the baseline. The *layer sharing* extension shares a fully connected layer between the paraphrase task and semantic textual similarity task, which slightly improves the performance of paraphrase prediction. However, since it reduces the number of parameters in the model, its overall performance falls behind that of the *finetune* model with no extension. In the *cosine similarity* extension, we experimented with a prediction head that uses simple cosine similarity computation without neural network layers. Unfortunately, the limitation in expressiveness is reflected in the experiment results. The Multitask-BERT model with only the *pair encoding* extension significantly boosted the dev score, as presented in Table 2, and we retained this extension for all subsequent experiments. This extension allowed each sentence to attend to the other in the same pair, increasing the model's expressiveness and improving performance in sentence pair prediction tasks. However, incorporating *learnable weights*, *homoscedastic uncertainty*, *gradient surgery*, or *sample task* extension on top of the *pair encoding* extension did not result in a significant improvement in performance. For instance, *gradient surgery* might not have improved performance because conflicting gradients were not a significant issue between the three tasks we trained on. Among all the experiments, the *pair encoding* and *automatic weighted loss* extension achieved the highest overall score. The *automatic weighted loss* extension weighed multiple loss functions by considering the homoscedastic uncertainty of each task and prevented the loss from becoming negative during training, facilitating the training process. Therefore, we selected this model for further hyperparameter tuning.

Table 2: Performance of `swissBERT` with Different Extensions.

Extensions	SST Acc	Para Acc	STS Corr	Overall Score
Baseline: no extension(pretrain)	0.334	0.649	0.267	0.417
no extension(finetune)	0.508	0.755	0.492	0.585
layer sharing	0.500	0.756	0.464	0.573
cosine similarity	0.482	0.506	0.463	0.484
pair encoding	0.506	0.855	0.842	0.734
pair encoding, learnable weights	**0.510**	0.847	0.848	0.735
pair encoding, homoscedastic uncertainty	0.500	0.855	0.846	0.734
pair encoding, auto weighted loss(awl_lr=1e-5)	0.508	0.862	0.849	**0.740**
pair encoding, gradient surgery	0.503	0.837	0.843	0.728
pair encoding, gradient surgery, learnable weights	0.503	0.837	0.842	0.727
pair encoding, gradient surgery, auto weighted loss(awl_lr=1e-4)	**0.510**	0.850	**0.851**	0.737
pair encoding, sample task, auto weighted loss(awl_lr=1e-4)	0.495	**0.865**	0.841	0.734

Table 3 shows the result of hyperparameter tuning. We found that the combination of a learning rate of 1e-5, dropout probability of 0.3, weight decay of 1e-5, and automatic weighted loss learning rate

of 1e-4 achieves the highest overall dev score of 0.747. This is a slight improvement over the default hyperparameters, which resulted in an overall dev score of 0.740.

Table 3: Hyperparameter Tuning Experiments for `swissBERT`

LR	Dropout	Weight Decay	AWL LR	SST Acc	Para Acc	STS Corr	Overall Score
1e-5	0.3	0	1e-5	0.508	**0.862**	0.849	0.740
1e-4	0.3	1e-5	1e-4	0.467	**0.862**	0.834	0.721
1e-5	0.3	1e-5	1e-4	**0.525**	0.849	0.868	**0.747**
1e-5	0.3	1e-4	1e-3	0.501	0.861	**0.875**	0.746
1e-6	0.3	1e-5	1e-4	0.489	0.788	0.793	0.690
1e-5	0.5	1e-5	1e-4	0.507	0.851	0.842	0.733
1e-5	0.1	1e-5	1e-4	0.517	0.854	0.848	0.740

We achieve the best overall score on the dev set by incorporating a learning rate scheduler into `swissBERT` model. We submitted the predictions to the dev and test leaderboard and obtained the performance in Table 4.

Table 4: `swissBERT` Performance on Dev/Test Leaderboard

Dev Leaderboard				Test Leaderboard			
Overall Score	SST Acc	Para Acc	STS Corr	Overall Score	SST Acc	Para Acc	STS Corr
0.754	0.510	0.883	0.868	0.758	0.527	0.885	0.862

6 Analysis

In this analysis section, we present a comprehensive examination of our `swissBERT` performance on the SST, Para, and STS tasks. By scrutinizing the model's strengths and weaknesses, we aim to understand its underlying capabilities and limitations, offering valuable insights for future research and improvement.

SST Task. Our model works well on reviews that convey a clear and explicit sentiment. The model has correctly identified negative reviews that use phrases like "meaningless downer", "load of junk", and "hackneyed message". Positive reviews that have been correctly identified contain language such as "remarkable camerawork", and "deliriously funny".

True: 0, Pred: 0
Sentence: i got a headache watching this meaningless downer.

On the other hand, the model is struggling with reviews that use more complex language, where the sentiment is more ambiguous. The phrase "glorious failure" is a contradiction in terms, as it suggests that the movie is both a failure and something to be admired or celebrated. This type of language can be challenging for models to interpret because it requires a deeper understanding of the context to disambiguate, such as the case of "Solaris" being considered a failure by other critics. Nuanced language adds to the difficulty in accurately classifying sentiment, as it requires understanding specific connotations. Our model also struggles with reviews that discuss multiple aspects of the movie with varying sentiments, such as the second review, which praises the film's technique and ideas while simultaneously criticizing it for its lack of substance or content to be a good movie overall.

True: 4, Pred: 2
Sentence: if steven soderbergh's 'solaris' is a failure it is a glorious failure.

True: 3, Pred: 1
Sentence: So much facile technique, such cute ideas, so little movie.

Para Task. Our model effectively recognizes paraphrases when sentence pairs exhibit similar structures, contain common keywords, or use synonyms to express the same idea. Additionally, it accurately differentiates non-paraphrased sentence pairs when they address distinct aspects, even when sharing common keywords. For example, both sentences below relate to Hillary Clinton but focus on entirely different aspects. Our model successfully identifies this distinction.

True: 0, Pred: 0
Sentence 1: What made Hillary Clinton join Quora?
Sentence 2: What is Hillary Clinton really like?

However, the model tends to misclassify sentence pairs as paraphrases when they share a similar topic or context, but one sentence poses a more specific question while the other addresses a broader subject. In the following example, both sentences concern improving English learning, but the second sentence specifically targets English vocabulary. In this case, due to the subtle difference, the model incorrectly predicts them as paraphrases of each other.

True: 0, Pred: 1
Sentence 1: What are the best ways to improve English?
Sentence 2: How can I improve my English vocabulary?

The model also struggles with paraphrases that have significantly different structures. In the example below, they are paraphrases of each other, although the first one uses the subject "I" and the second one uses the subject "my school."

True: 1, Pred: 0
Sentence 1: Am I forced to say the pledge of allegiance at school when I have a different religion?
Sentence 2: Can my school legally force me to say the pledge of allegiance, and punish me if I refuse?

STS Task. In our analysis of the STS task, we observed that the model performed well when sentence pair with high similarity score shared similar structures, clear contexts, or described similar actions with slightly different wording. For example, the model correctly predicted a high similarity score for the following pair:

True: 4.0, Pred: 3.9961
Sentence 1: rallies demand ‘justice for trayvon’
Sentence 2: across us, people rally for ‘justice for trayvon’

This pattern can also be explained through the attention heatmap on layer 11, as shown in figure 3. We can see that word in one sentence tends to have large attention on the exact match of another sentence.

On the other hand, our model does not have the best performance when the context of two sentences are ambiguous, sentences have high similarity in structure but differ significantly in meaning, or there is a mismatch of details between two sentences. For example, when the sentences that have similarities in structure but differ significantly in meaning or focus, the model tends to overestimate the similarity score:

True: 0.0, Pred: 2.3796
Sentence 1: you should do it.
Sentence 2: you should prime it first.

In this example, the two sentences share a similar structure, but their meanings are quite different. The first sentence is a generic statement encouraging someone to perform an action, while the second sentence specifically advises someone to prime an object before proceeding. Despite their distinct

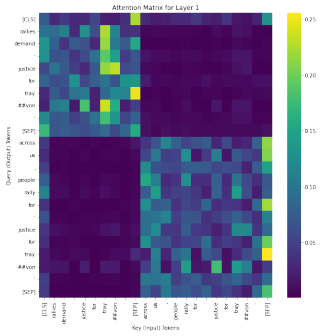


Figure 1: First-head attention heatmap for layer 1

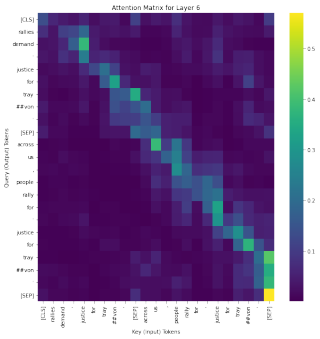


Figure 2: First-head attention heatmap for layer 6

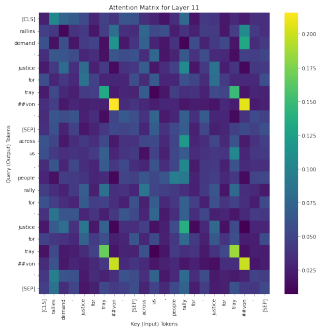


Figure 3: First-head attention heatmap for layer 11

meanings, the model predicts a similarity score of 2.3796, indicating a notable discrepancy between the true score and the predicted score. We may further improve our model by providing more training data that have similar characteristics.

An interesting finding from the attention heatmap is observed in the early phase of our `swissBERT` model, as shown in figure 1. In this layer, words in one sentence do not attend to words in the other sentence. This is likely because the model is initially focused on understanding the individual sentences, processing the syntax, and local dependencies within each sentence.

As the model progresses through the layers, it builds a more complex understanding of the sentences. For instance, in layer 6 (figure 2), words tend to attend to their local neighbors within each sentence, indicating a more contextualized understanding of the input text. By layer 11, the model effectively attends to the exact matches between sentences, capturing the semantic relationships and establishing connections between words in both sentences.

7 Conclusion

In this study, we demonstrate the effectiveness of `swissBERT` in transfer learning for downstream tasks on two sentiment analysis datasets through individual fine-tuning. Building on this, we extend `swissBERT` to simultaneously perform three different downstream tasks using *multitask finetuning*, including sentiment analysis, paraphrase detection, and semantic textual similarity. We implement a number of additional extensions and conduct an ablation study to examine their benefits and drawbacks. Among these extensions, we find *pair encoding* and *auto weighted loss* to be highly effective and consistently improve the model's performance across tasks. Inspired by the NSP task from BERT [1], *pair encoding* concatenates sentence pairs from the paraphrase detection and semantic textual similarity tasks, allowing for the direct encoding of the concatenated sentence pair in one BERT forward call. This approach effectively captures the interactions between the words in both sentences. Furthermore, our experiments show that *auto weighted loss* provides an effective way to learn relative task weighting, using a well-formed objective that avoids task weights convergence to zero and considers homoscedastic uncertainty. Our experiments demonstrate an overall dev accuracy of 0.754 and an overall test accuracy of 0.758 on the leaderboard.

In light of our findings, we propose several avenues for future research to improve our model's multitask performance. First, we plan to train the BERT weights on additional datasets from the three task categories for the model to generalize to diverse human language, such as the Twitter Sentiment Analysis Dataset [8], before fine-tuning. We also plan to experiment with synthesizing training data for the major buckets of errors we identified in Section 6. Additionally, we have observed overfitting to be a significant challenge, particularly for the sentiment analysis task. To mitigate this, we intend to explore the use of Smoothness-inducing regularization and Bregman Proximal Point Optimization, as suggested by Jiang et al. [9], to help our model generalize better on unseen data.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

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

[4] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning, 2018.

[5] Yifan Peng, Qingyu Chen, and Zhiyong Lu. An empirical study of multi-task learning on BERT for biomedical text mining. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 205–214, Online, July 2020. Association for Computational Linguistics.

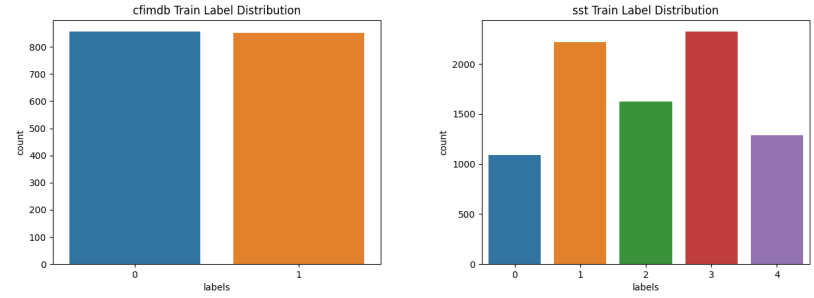
[6] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[7] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics.

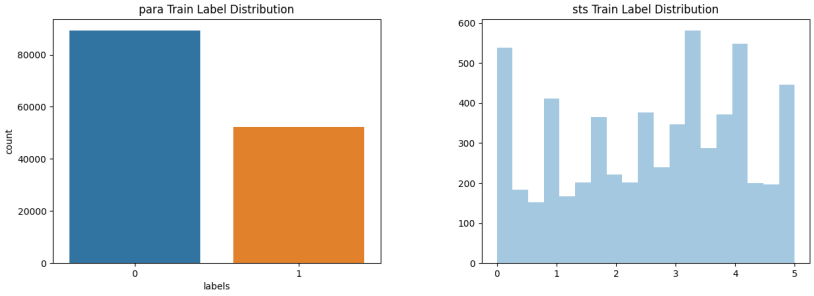
[8] Ibrahim Naji. TSATC: Twitter Sentiment Analysis Training Corpus. In *thinknook*, 2012.

[9] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 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*. Association for Computational Linguistics, 2020.

A Appendix



(a) CFIMDB (b) SST



(c) Qoura (d) STS

Figure 4: Training Data Label Distribution

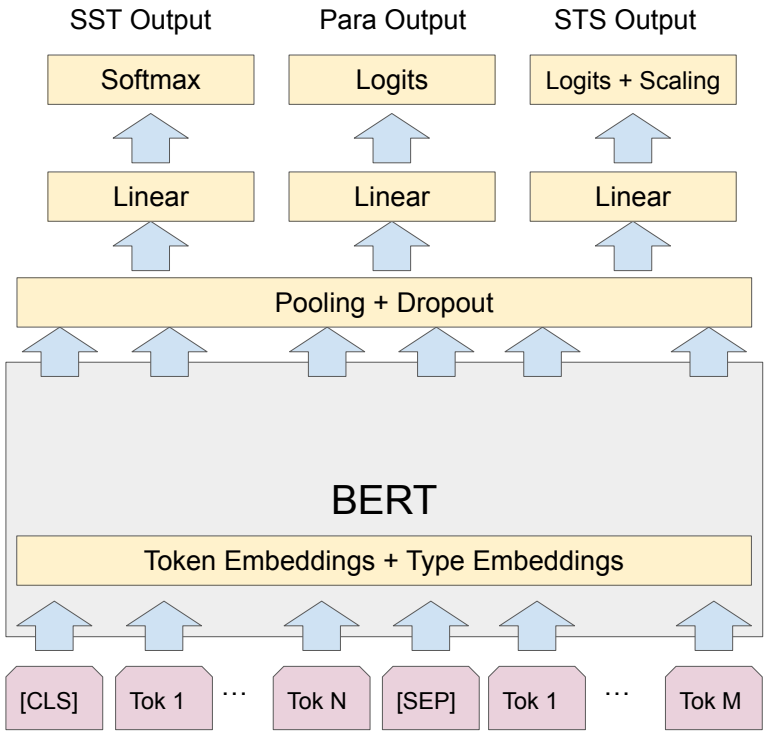


Figure 5: Model Architecture with `pair-encoding` extension. Note that For SST task, there is no `[SEP]` token, because the input is only one sentence.