BERT++: Trustworthy MultiTask Learning with BERT

Stanford CS224N Default Project

External collaborators/External mentor/Sharing project: NA

Mentor: Tathagat Verma

Two late days were used in this project

Zilu Wang zilu@stanford.edu

Yuwei Wu yuweiwu@stanford.edu

Anh Hoang Nguyen anhng@stanford.edu

Abstract

Bidirectional encoder representation transformers have demonstrated state-of-the-art performance in a variety of NLP tasks. However, there is still room for improvement in finetuning pre-trained LLMs for downstream tasks such as sentiment analysis, paraphrase detection and semantic textual similarity analysis. In this paper, we aim to establish robust and generalizable sentence embeddings to improve performance in the three downstream tasks through experiments include additional pretraining on domain-specific corpora, cycled multitask finetuning with task-specific loss functions, gradient surgery to mitigate the vanishing gradient problem and extensive explorations on the prediction layers. We conduct extensive experiments on several benchmark datasets and show that our improved BERT models outperform the original BERT and other baselines by significant margins. We also provide ablation studies and error analysis to understand the effects of each improvement and iterate our model accordingly.

1 Introduction

As one of the most influential breakthroughs in NLP in recent years, BERT (Bidirectional Encoder Representations from Transformers), is a pre-trained language representation model that can be fine-tuned for various downstream tasks. BERT leverages a large amount of unlabeled text data and a novel bidirectional training to learn rich contextual representations of words and sentences. It is shown in the original BERT paper that it has achieved state-of-the-art results on several benchmark datasets across different NLP tasks[1]. In this paper, we propose several improvements to BERT for three downstream tasks: sentiment classification, semantic similarity classification and paraphrase detection. These tasks are important for understanding natural language semantics and pragmatics. However, it remained hard for these tasks in a way that natural language is complex and ambiguous. Different words could have different meanings depending on the content. In addition, for example, semantic similarity detection requires domain knowledge and common sense, whereas sentiment classification may involve in subjective comments or personal preferences. The training data could face data scarcity and imbalanced class issues, which makes the downstream tasks especially hard.

Through experimentation, extensions like additional pretraining, multitask finetuning, gradient surgery and task specific loss function do achieve better performance comparing to the baseline implementation. Our final BERT++ model is built based on iterations of searching the optimal extensions that could enhance the performance on the dev set. We got this idea from the Forward Stepwise Selection[2], where we started from the baseline model and added one extension each time that results improvement on the dev set, evaluated by the overall score. We conduct extensive experiments on several benchmark datasets and show that our improved BERT models outperform the original BERT and the baseline model significantly.

2 Related Work

As shown in the BERT original paper, it can be fine-tuned for various downstream tasks by adding additional output layer on top of the transformer output, where many researches in the field have been done for improving the performances in downstream tasks. One way to improve BERT is to perform additional pre-training on a domian specific corpus, which can reduce the data or concept drift from the original training distribution for the main BERT model. Sun et al.[3] proposed the two-step learning framework that first pretrain the BERT model on domain corpus and then finetune it on well-labelled datasets for downstream tasks, which obtained new SoTA results on eight widely studied text classification datasets.

Multitask learning by Bi et al.[4] proposed a finetuning framework that instead of finetuning the downstream tasks individually, those tasks can be finetuned simultaneously to leverage the common knowledge among related tasks. Stickland and Murray[5] proposed to optimize the single model using weighted sum of task-specific losses, in addition to gradient clipping, learning rate decay, etc.

However, such vanilla multitask learning may have divergent gradient directions or conflicting optimization objectives. Yu et al. proposed a gradient surgery[6] technique that can mitigate these issues by dynamically adjusting the gradient update directions based on task similarity and importance, which outperforms their concurrent existing multitask learning frameworks on several NLP tasks. In addition, Caruana et al.[7] indicates that shared representation in neural networks can benefit from multitask learning, and work has been done in shared hidden layers for both face detection and pose estimation to improve generalization by using domain information.

Some other domain specific advances in the field includes the SBERT[8], which introduced a siamese network architecture that trains BERT on pairs of sentences that is useful in natural language inference, semantic textual similarity, and paraphrase identification. The paper evaluates SBERT on several sentence-level tasks and shows that it achieves competitive results with more efficient and stable computation for large scale tasks. SpeBert proposed by Liu et al.[9] discussed using sentence parts with respect to downstream tasks to enhance sentence representations through encoding sentence parts based on dependency parsing and downstream tasks, and extracts embeddings through a pooling operation can significantly improves the performance on benchmark datasets. TBert proposed by Peinelt et al.[10] combines the topic models such as LDA with pretrained contextual representations from BERT to capture local or global semantic information that achieves a better performance in similarity prediction.

In this paper, we will examine these advances in the field and try to establish a trustworthy model with more accurate, more reliable and more robust performance comparing to the baseline model performance.

3 Approach

Our model was built upon "minBERT", which is minimalist implementation of BERT that preserves the main structure, tokenizers, multi-head self-attention, etc. The minBert model contains an encoder that converting the input texts to embeddings that captures the meaning of each word, in addition to the tokenization that splits the input words into subwords based on frequency to handle rare or unknown words. The transformer part[11] is for processing the embeddings and generate representations that could be further used for other tasks. Some major advances, such as multi-head self-attention, allows the model to jointly attend to the representations across different positions.

Our baseline model was built upon this minBERT model with simple prediction heads in each of the downstream tasks without any multitask training(so the parameters used for prediction are purely from initialization). This aligns with what we expected. The performance for paraphrase and semantic textual similarity is not satisfactory, because the weights for the last two downstream tasks are not trained.

We adopted the round-robin style individual training for three downstream tasks. In each of the training epoch, the three tasks are trained in parallel sequence. The model first reads in batches of data from SST and train the model and perform gradient update, does the same for the other two tasks. However, the gradient update for the three downstream tasks might conflict with each other in such individual training. Hence the model might tend to be specifically finetuned to STS task and the performance for the first two tasks could be affected in a negative way.

To tackle the potential problems in individual fine-tuning BERT, we used multitask finetuning discussed by [4] that defines the loss function as follows:

$$L_{Total} = L_{Sentiment} + L_{Paraphrase} + L_{SemanticSim}$$

Then perform gradient update for each batch. It is shown [12] that adjusting and cross validating weights in loss functions could improve the performance. We experimented with different weights as hyperparameters to put more weights on evaluating the performance of the earlier two tasks (SST, Paraphrase) than the last task trained (STS).

As mentioned above, the gradient update directions for different tasks in multitask learning could be contradictory with each other, hence, for example, training on the semantic similarity task could influence the sentiment classficiation in a negative way. Yu et al. proposed the following gradient update rule that can project the gradient of the current task (g_i^{PC}) on to the normal plane of other conflicting tasks gradients g_j as following to mitigate the gradient interference:

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

We also tried a large number of prediction layer structures, the corresponding activation functions, and as well as the shared weights between the paraphrase detection and semantic similarity because they both have the same input structure (i.e. a pair of sentences) and they both intrinsically measure the similarity between the input sentences. Hence our first approach was to define a cosine similarity layer for the paraphrase task and the semantic similarity task. The specific configurations for the returning logits will be discussed below. Then we experimented on sharing the linear layers between paraphrase detection and semantic similarity task. The corresponding loss functions were chosen carefully with respect to each of the task.

Additional pretraining on in-doman datasets (datasets that are obtained from the same domain of training data for target task) have been shown to improve the performance of finetuning BERT for downstream tasks [3]. Thus we further pretrained the BERT model on 3 datasets similar to our training datasets on masked language model task: 1) Large Movie Review Dataset (IMDB), 2) Multi-Genre Natural Language Inference (MultiNLI) corpus [13], and 3) Sentences Involving Compositional Knowledge (SICK) [14]. The latter two datasets include sentence pairs labeled for Semantic Textual Similarity tasks. Our implementation was adapted from HuggingFace's Fine-tuning a masked language model tutorial [1] where we expanded the dataset to include MultiNLI and SICK data.

4 Experiments

4.1 Data

For additional pretraining: 1.1 Large Movie Review Dataset (IMDB) [2]: 50,000 single sentences of movie reviews with binary labels. The splits are train 25,000, test 25,000 and unsupervised (no label) 50,000. 10,000 examples were sampled from the unsupervised split for additional pretraining. 1.2. Multi-Genre Natural Language Inference (MultiNLI) [13] 392,702 sentence pairs labeled for their similarity level ('entailment', 'neutral' and 'contradiction') 1.3. Sentences Involving Compositional Knowledge (SICK) [14] consists of 10,000 sentence pairs that include many examples of the lexical, syntactic and semantic phenomena. The train set split with 4,439 pairs (labeled same as MultiNLI in addition to a relatedness score on scale of 0-5) was used for additional pretraining.
2. Stanford Sentiment Treebank (SST) Dataset [15][3]: 11,855 single sentences of movie reviews with 5-level labels. The splits are: train set with 8,544 examples, dev 245 and test 488. This data is used to train sentiment analysis task. 3. Quora Dataset[4]: 400K question pairs with binary labels. The splits

[1]`https://huggingface.co/course/chapter7/3?fw=pt#fine-tuning-a-masked-language-model`
[2]`https://https://huggingface.co/datasets/imdb`
[3]`https://nlp.stanford.edu/sentiment/treebank.html`
[4]`https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`

are: train set with 141,506 examples, dev 20,215 and test 40,431. This data is used to train paraphrase detection task. 4. SemEval STS Benchmark dataset[16]: 8,628 sentence pairs with similarity labels rated 0-5. The splits are: train set with 6,041 examples, dev 864 and test 1,726. This data is used to train semantic textual similarity.

4.2 Evaluation method

To evaluate the model performance, we chose several metrics for both the development set and the test set. In particular, we used accuracy for both the sentiment classification and the paraphrase detection tasks, because the true labels are discrete labels. Originally, we tried to use Balanced Accuracy(using the reciprocal of number of occurrences for each true labels as weights) to prevent the possible consequences of class imbalanced problem. However, the training SST Dataset and Quora Dataset are not suffered from class imbalance problems. The label-count distribution is listed as follows: ids-sst-train.csv: {0: 1092, 1: 2218, 2: 1624, 3: 2322, 4: 1288} and quora-train.csv: {0: 89225, 1: 52273}. Hence we anticipate that class imbalance will not be a problem in evaluating the performance for our models.

For the STS task, we used the Pearson correlation coefficient metric to evaluate the similarity between our predicted class and the true label classes. This is because the acutal labels in the STS Benchmark Dataset is continuous from 0 to 5. Using the Pearson correlation coefficient will not have the problem of scaling, comparing to other methods like Mean Squared Error. At first, we experimented with using MSE as the evaluation method; however, MSE will limit our flexibility in experimenting the prediction heads for the STS task. Specifically, MSE is not robust to different scaling of the true labels and our predictions(For example, using cosine similarity as prediction heads). Hence we used Pearson Correlation to capture the relative accuracy that is robust to different scales. The overall score is then calculated by simply taking the average of all the metrics in these three tasks.

4.3 Experimental details

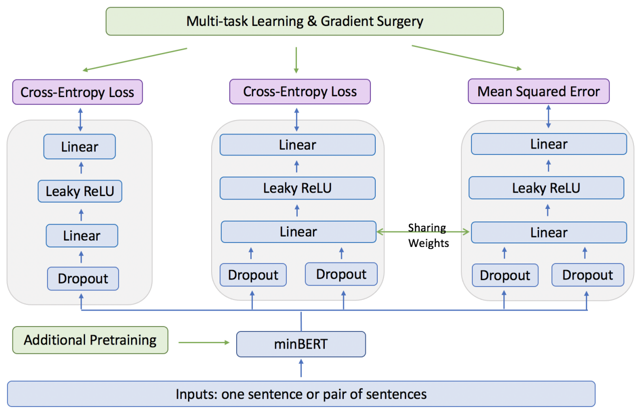


Figure 1: Model Main Structure

As mentioned above, our strategy for improving the model takes the concepts from "Forward Stepwise Selection", which is a statistical greedy method that make improvements from the baseline model, and "Beam Search", which has proven to leverage the computation cost as well as accuracy. For a given model, we experimented several extensions(discussed below) and select one that leads to the greatest improvement in our model. As we repeat the procedure, the models are iterated and improved. All the model iterations are listed in Table 1, and details are included in Table 4 in the appendix.

4.3.1 Additional Layers in Prediction Heads

From the baseline model, which was built upon the minBERT with simple prediction heads and with no additional finetune training on the paraphrase detection and STS tasks, we gradually improve our model by first adding the prediction layers and perform individual task finetuning(Model A). The prediction heads structure for the downstream tasks is shown in Figure 1. The structure for both paraphrase detection and STS are highly similar due to their intrinsic similarity in the task nature.

4.3.2 Additional Pretraining

Additional Pretraining(Model B) was performed on a combined dataset of IMDB unsupervised split with 50k examples, MultiNLI 20k examples and SICK 4,439 examples (without label). For each example in the MultiNLI and SICK datasets, two sentences in the pair were concatenated to make a new text example. These 3 datasets were then concatenated into a new dataset. The next preprocessing step was to concatenate all the examples and then split the whole corpus into chunks of equal size of 128. Finally, these chunks are tokenized using the AutoTokenizer from the transformers library. To train the masked language model, we used the BertForMaskedLM from the transformers library applied on the pretrained BERT model and masked 15% of the tokens.

4.3.3 Loss Functions and Multitask Learning

As discussed in Section 4.1, we know that the true labels for both Sentiment Analysis and Paraphrase Detection are discrete labels. Hence we frame this two as classification problems and we used cross-entropy loss for the sentiment analysis task(multi-label) and binary cross-entropy loss for the Paraphrase Detection task(binary label). For the STS task, we used Mean Squared Error loss to minimize the distance from the predicted logit and the actual labels.

We experimented Multitask Learning in parallel(Model C), which takes the "bert-base-uncased" pretrained BERT model weights that has been trained on a large corpus of text data mainly from Wikipedia and BooksCorpus. The multitask Learning was implemented by incorporating the gradient update for all three tasks by using a total loss function:

$$L_{Total} = L_{Sentiment} + L_{Paraphrase} + L_{SemanticSim}$$

The corresponding weights were initialized to 1 for all three tasks, and they were experimented along the way when error analysis was made for performance evaluation.

4.3.4 Gradient Surgery

To address the potential direction conflicts in gradient updates, we also used gradient surgery(Model D) to mitigate the interference between gradients from different tasks in multitask learning. To accomplish this, we used PCGrad package[17] to wrap the optimizer function, then we stored the losses from each of the downstream tasks in an array and perform gradient surgery on the array described in the original paper[6].

4.3.5 Cosine Similarity Layers

Cosine Similarity is known to be a similarity measure for two vectors in an inner product space ranging from -1 to 1. The intrinsic characteristics of the cosine similarity could be perfectly applied to tasks that are measuring the similarity of two inputs. Thus we applied the cosine similarity as the output layer of both paraphrase detection and semantic similarity task(Model E). These two tasks are first using linear layers to learn representations from the BERT sentence embeddings(with dropout layers), then they were passed into activation functions and cosine similarity layers to return a logit. A ReLU layer is added to the end of paraphrase detection layers as the true label for paraphrase detection is 0 or 1. More experimentations have been done to rescale the cosine similarity output with the STS task, which has true labels from 0 to 5. For example, the output was multiplied with a scalar, then passed into a sigmoid function so that the output ranges from 0-1, then times 5.

4.3.6 Cycled Training and Hyperparameter Tuning

The training data is highly imbalanced across different tasks, and this the resulted performance on the dev set illustrates this data imbalance nature. Hence we designed a cycled training procedure that within each epoch, the layers will consistently be trained by the data until the training from the largest dataset(Quora) is over(Model H). Repeated training may result in overfitting to the training data for Sentiment Analysis and STS, hence proper regularization has been made in, for example, increasing the dropout probability, decreasing the training batch size and learning rate, etc.

Version	Improvement choices						
	AdjLayers	AddPretrain	MulFinetune	GradSurg	CosSim	HyperTune	CycTrain
Baseline							
Model A	✓						
Model B	✓	✓					
Model C	✓		✓				
Model D	✓		✓	✓			
Model E	✓		✓	✓	✓		
Model F	✓		✓	✓		✓	
Model G	✓	✓	✓	✓		✓	
Model H	✓	✓	✓	✓		✓	✓

Table 1: Model Experimentation

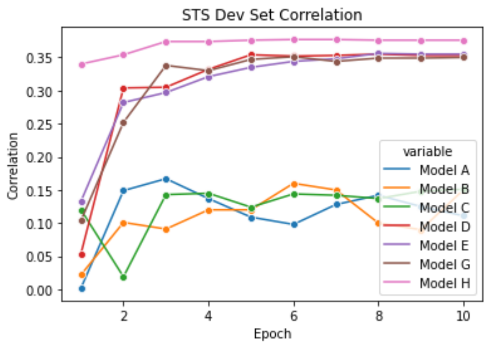
4.4 Results

Our models are evaluated on the Dev Set, which the performances can be found in Table 2. By comparing Model A and baseline in Table 2, we see that training for the last two tasks can significantly improve the dev set accuracies, comparing to the baseline model where we only trained the sentiment prediction task. Across all models with/without additional pretraining (by comparing Model A with Model B, or comparing Model F with Model G), we see that with pretraining, the performance for sentiment analysis and paraphrase detection in general could be improved. However, with additional pretraining, the overall performance for STS task is not better. The improvement for the sentiment analysis can be explained by the fact that the training data for this task is more niche (movie reviews) compared to the Quora dataset and SemEval STS Benchmark dataset which are used for the other two tasks. The difference in the impact of further pretraining on paraphrase detection and STS task can be attributed to their different objectives. Since the masked language model used in additional pretraining does not look at the labels, the type of data used for these tasks become identical. Moreover, the training data size for paraphrase detection is much larger than that for STS, thus using the pretrain data does not alleviate this issue.

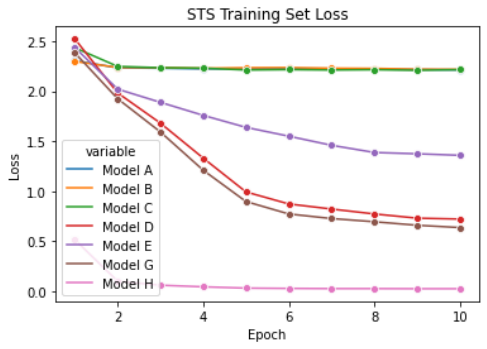
By comparing the performances for Model C and Model D, it is shown that gradient surgery have a significant improvement for all three downstream tasks comparing to vanilla multitask learning BERT(Model C). This verifies that the conflict gradient updates are indeed detrimental to the model performance, and the gradient surgery provides an effective solution to this problem in multitask learning. On the contrary, the cosine similarity prediction layer for Paraphrase Detection(with sigmoid transformation in its loss functions) is not performing as we originally expected to capture the embedding adjacency in dot product space for this PD task.

Over the iterations of models, we found that the performance for STS is not as good as the other two in general. A lineplot of the STS task performance on the Dev Set is shown in Figure 2(Left). One possible reason could be that the amount of data in sts-train.csv is far less than the other two tasks, so that the scarcity of data could lead to a poor performance. One improvement we made to tackle the scarcity of training data, in addition to the additional pretraining, is to adopt a cycled training framework that kept training on Sentiment Analysis and STS task until the Paraphrase Detection training finishes. This cycled training framework is corresponding to Model H here, which has the highest performance across all models. Despite the fact that overtraining on the same small dataset may lead to overfitting, the Dev Set performance justifies our approach. The overfitting problem can be visualized from the STS Training Loss in Figure 2(Right).

Some of the selected models with best Dev Set performances are tested on the Test data, and their test set performances are reported in Table 3. By comparing the test set performances and the dev set performances, we can conclude that the metrics evaluated on the dev set serve as a great approximation to the generalization error, and the test set performances for Model D, G and H are similar to those in the dev set across all tasks. Model H achieves the best overall performance in both Dev set and Test set, with leading performances in Sentiment Acc and STS corr. However, the accuracy in paraphrase detection task is slightly worse than the other reported models. We anticipate the boost of performance in both sentiment analysis and STS task is because of the cycled training that those two tasks are trained on more iterations of data. The slight decrease in the paraphrase detection accuracy could come from the reason that, with more training iterations on Sentiment Analysis and STS task, the gradient update between them could be conflicting with each other originally. In other



(a)



(b)

Figure 2: STS Dev Set Correlation(Left) and Training Loss(Right)

Version	Dev Set Performance			
	Sentiment Acc	Para Acc	STS Corr	Overall
Baseline	0.312	0.437	0.019	0.256
Model A	0.463	0.788	0.246	0.499
Model B	0.486	0.794	0.162	0.481
Model C	0.426	**0.801**	0.121	0.449
Model D	0.446	0.8	0.332	0.526
Model E	0.508	0.522	0.352	0.461
Model F	0.505	0.796	0.343	0.548
Model G	**0.523**	0.79	0.336	0.550
Model H	0.520	0.788	**0.375**	**0.561**

Table 2: Model Performance on Dev Set

models, such as model G(without cycled training), the gradient update leans slightly towards the paraphrase detection task because more data(and hence more batches) were trained on that task. However, with same amount of batches across all three tasks training, the problem is solved, resulted in an increase in Sentiment Acc and STS Corr for Model H at a cost of slightly decrease in Para Acc.

5 Analysis

With every iteration of the model, we performed error analysis on the predictions and rationalize the performance outcome from each of the training epochs. Throughout the entire model iteration, sufficient comprehension and understandings are made from the models, and we made informed improvements on the model structures or hyperparameters.

From the model with multitask learning loss, we printed out the gradient updates for each of the three tasks during gradient update in training, and we found that despite some of the gradient updates have similar magnitudes, sometimes the gradient updates are contradictory if we use the same batch of data in individual training. This indicates the conflicting gradient directions for updating could severely influence the model performance. That is the motivation why we adopted gradient surgery to tackle with conflicting gradient update problem.

Version	Test Set Performance			
	Sentiment Acc	Para Acc	STS Corr	Overall
Model D	0.514	**0.797**	0.326	0.546
Model G	0.516	0.793	0.303	0.537
Model H	**0.53**	0.787	**0.35**	**0.555**

Table 3: Model Performance on Test Set

In the cycled training framework, we found that with batch size equals to 8, the STS task will train 17688 batches, comparing to the original training batches of 755. This is 2243% increase in the amount of training batches for the STS task. This could lead to the overfitting problem for the prediction layers in the STS task. To tackle this problem, we increased the dropout rate specifically for this task reduce the problem of overfitting. Surprisingly, despite the training loss for STS shows a trend of overfitting, the performance on the Dev Set is still improving slowly. We anticipated that the feed forward layers in the STS task is still too shallow so that the model cannot learn further information from the repeated training of data.

As discussed in previous sections, the motivation for using cosine similarity for the last two tasks was the nature of the tasks are trying to capture the semantic or pragmatic relationship between two embeddings. Despite the fact that Cosine Similarity prediction failed to succeed in our experimentation, we analyzed the cosine similarity for specific inputs of data in the STS task and found that rescaling technique we used for the Cosine Similarity output is not robust comparing to the actual labels. Our predicted labels were centered at around 2.5, which was because we used the following formula: sigmoid(CosSim)*5. The sigmoid function for small numbers(cosine similarity are scaled between -1 to 1) are centered around 0.5. That was the initial reason for the failure of cosine similarity. We experimented additional rescaling methods and recorded the best performance in Table 2, but we can observe that the performance is worse than the other models using linear prediction layers.

Last but not the least, we motivated from the similar nature in paraphrase prediction and the STS task to experiment on sharing the linear layers between the two tasks. Specifically, we first experimented on sharing the linear layer for further learning the embeddings from the BERT model. Then we found that the performance is specially bad when we run 3 epochs. We anticipated that the linear layers for learning the embeddings are different across those two tasks. With that being said, the linear layer that learns the similarity between the two inputs should be somewhat overlapping with each other, and we also experimented with sharing the "fusion" linear layer that takes two input embeddings and returns one representation for their similarity.

6 Conclusion and Future Work

We proposed BERT++ in this paper, a model built on minBERT with carefully designed downstream prediction layers and finetuned with multitask learning loss, with gradient surgery, cycled training for smaller benchmark datasets, and with additional pretraining of the BERT model on domain specific data. The proposed best model is shown to achieve an increase of overall performance by 119.1% comparing to the baseline model on the Dev Set. The accuracy of sentiment task improved by 66.7%, the accuracy of the paraphrase task improved by 80.32%, and the correlation of the STS task improved by 1847% by comparing the Model H evaluations and baseline evaluations on the Dev set.

Throughout the experimentation, performing the error analysis on predictions and rationalizing the performance outcomes greatly facilitated the model iteration and made models more comprehensible. The overall model iteration framework is much more efficient with our strategy of model experimentation that was drawn insights from Forward Stepwise Selection and Beam Search algorithm. In addition, given the large size of the Quora Dataset, we limited the number of training batches of the Paraphrase detection task during our exploration stage to reduce the iteration turn around time.

More adjustments in the layer structure such as deeper feed forward layer structure, sharing weights, etc. can further improve performance from our preliminary experimentation. The intuition behind that is that the layers in both Paraphrase detection and the STS task should share some commonalities in capturing the semantic similarity between the input embeddings. We have trained the model with fewer epochs and the results are promising. However, because of the sharing layers, we increased the number feed forward layers for both paraphrase and STS tasks. The training time became much longer than we expected. We think that this is one of the potential improvements for our model. For future work, We propose to train the BERT++ model with deeper prediction layers and more epochs. The motivation comes from the deep double descent[18] analysis in modern deep learning frameworks. In addition for future work, we propose to both pretrain and finetune our model on larger, domain specific datasets so that the model can have better semantic and pragmatic interpretations for the input, as well as could be well suited for a wider range of topical inputs in the three downstream tasks.

References

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

[2] Trevor Hastie, Robert Tibshirani, and Ryan J Tibshirani. Best subset, forward stepwise, or lasso? analysis and recommendations based on extensive comparisons. *Statistical Science*, 35(4):579–592, 2020.

[3] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings*, pages 194–206. Springer, 2019.

[4] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, 2022.

[5] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning (ICML)*, pages 5986–5995, 2019.

[6] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836, 2020.

[7] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, page 41–48, 1993.

[8] Nils Reimers and Iryna Gurevych. 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, 2019.

[9] Cheng Liu, Wei Zhu, Xiaoxiao Zhang, et al. Sentence part-enhanced bert with respect to downstream tasks. *Complex Intell. Syst.*, 2022.

[10] Nicole Peinelt, Dong Nguyen, and Maria Liakata. tbert: Topic models and bert joining forces for semantic similarity detection. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 7047–7055, 2020.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 5998–6008, 2017.

[12] Tao Gong, Tae-Hwan Lee, Chris Stephenson, Vikramjit Renduchintala, Suchismita Padhy, Abigail Ndirango, Chia-Hung Kuo, and Omer Hakki Elibol. A comparison of loss weighting strategies for multi task learning in deep neural networks. *IEEE Access*, 7:141627–141632, 2019.

[13] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.

[14] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[15] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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*, page 1631– 1642, 2013.

[16] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. * 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*, page 32–43, 2013.

[17] Wei-Cheng Tseng. Pytorch-pcgrad. `https://github.com/WeiChengTseng/Pytorch-PCGrad/blob/master/pcgrad.py`, 2021.

[18] P. Nakkiran, H. Kaplan, A. S. Talwalkar, E. D. Cubuk, and F. Doshi-Velez. Deep double descent: Where bigger models and more data hurt. *ArXiv*, abs/1912.02292, 2020.

A Appendix

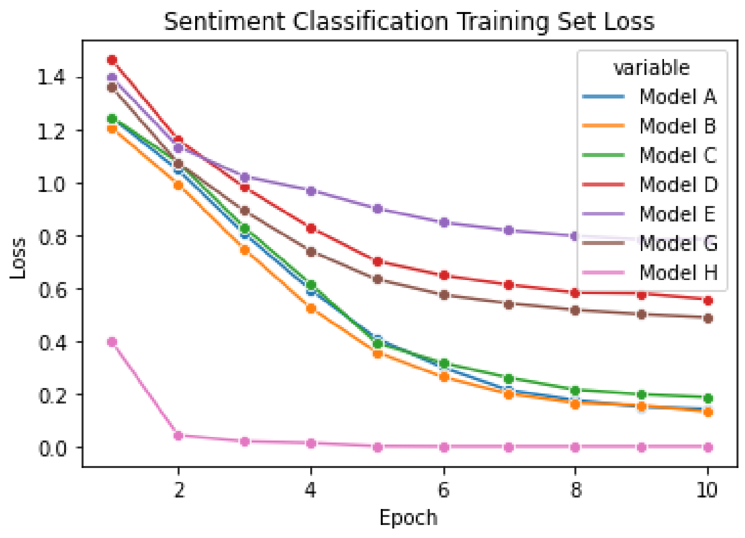


Figure 3: Sentiment Training Loss over Training Epochs

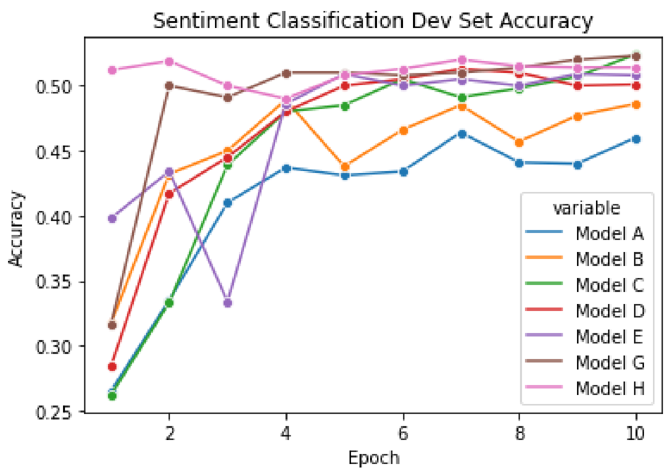


Figure 4: Sentiment Dev Accuracy over Training Epochs

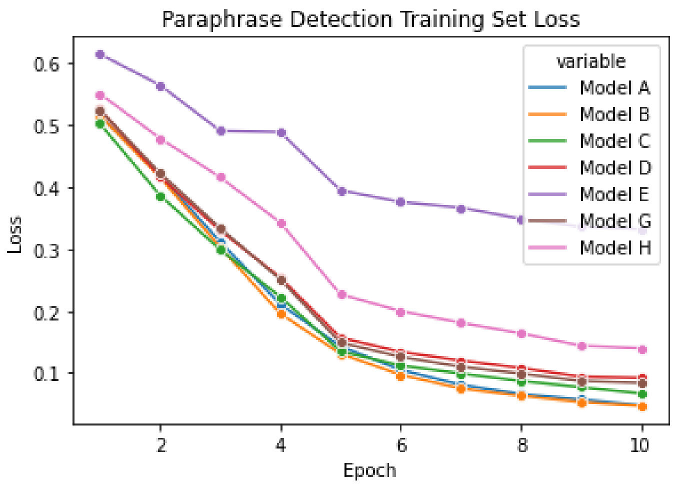


Figure 5: Paraphrase Training Loss over Training Epochs

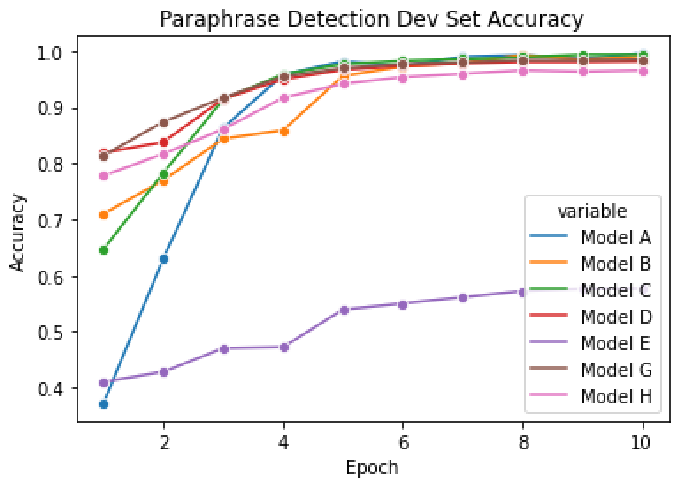


Figure 6: Paraphrase Dev Accuracy over Training Epochs

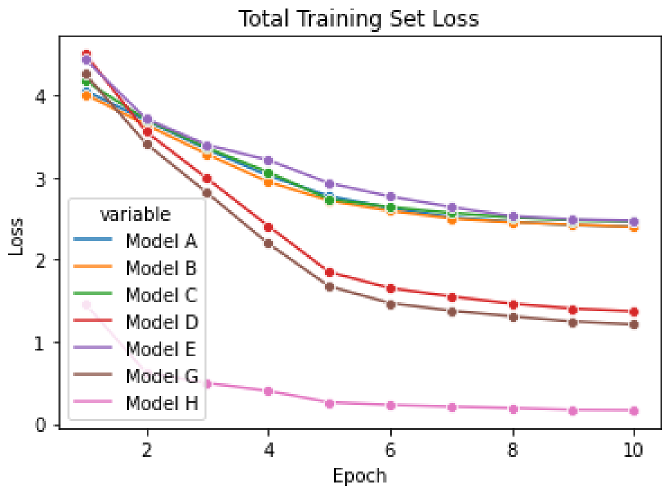


Figure 7: Total Training Loss over Training Epochs

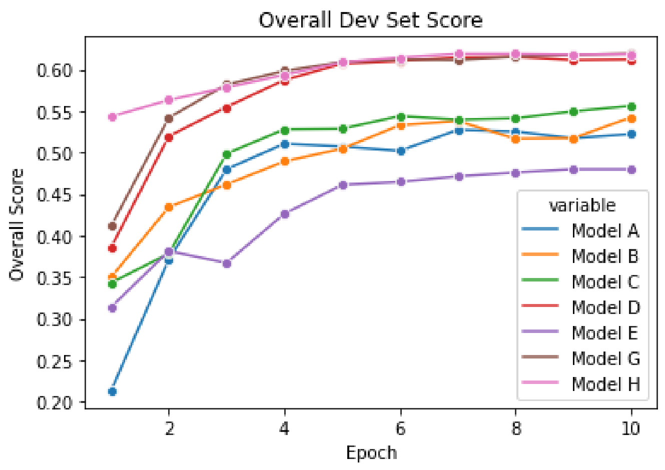


Figure 8: Overall Dev Score over Training Epochs

Version	Model Specifics
Baseline	simple prediction heads, no training on Paraphrase Detection Task and STS Task, 3 Epochs of training
Model A	more layers in prediction heads, experimented with activation functions, individual training on each of the tasks
Model B	Built on top of Model A, pretrained the BERT model on additional datasets with batchsize 16
Model C	Built on top of Model A, performed multitask learning by summing the loss functions and updated gradients simultaneously for all tasks
Model D	Built on top of Model C, performed gradient surgery with batchsize 16
Model E	Built on top of model D, where the prediction heads for PD task were changed to one linear layer and then cosine similarity formula
Model F	Built on top of Model D, with batchsize 16, Tuned the step and gamma in LR scheduler to be 5 and 0.1 respectively. Dropout rate=0.2
Model G	Built on top of Model F, Dropout for para = 0.8 and 0.3 for the other two. Pretrained on IMDB (10k), MultiNLI ($\sim$20k) and SICK ($\sim$4.5k), batchsize=16
Model H	Built on top of Model H, with cycled training on Sentiment Analysis and STS until PD finishes training, batchsize = 16

Footnote: # Epochs(10), batchsize(8), lr and dropout prob(0.3) are set to default unless specified.

Table 4: Model Specifics