Single Task Pretraining for Multitask Natural Language Processing with Gradient Surgery

Stanford CS224N Default Project

Govind Chada
Department of Computer Science
Stanford University
`gchada@stanford.edu`

Kabir Jolly
Department of Computer Science
Stanford University
`kjolly@stanford.edu`

Abstract

Recent natural language processing models have demonstrated strong performance on a wide variety of language tasks. However, since many are trained for a specific task, they are not capable of doing multiple tasks in a generalized setting. The goal of this work is to build a model that performs well in a natural language processing context through a multi-task learning based approach. As a baseline model, we finetuned a BERT pretrained transformer model with task-specific heads. This baseline attained an accuracy of 0.494 on the SST dev set, a Pearson correlation score of 0.386 on the Paraphrase dev set, and an accuracy of 0.325 on the STS dev dataset. The baseline approach was expanded on through the inclusion of cosine-similarity fine-tuning and gradient surgery, as well as hyperparameter optimization. This resulted in outperforming our baseline and motivated the work for our custom training pipeline, which involved pretraining task-specific models to serve as a good initialization point for multitask finetuning. This ended up being our highest performing model and was evaluated on the test dataset, achieving an accuracy of 0.522 on the SST task, a Pearson correlation score of 0.824 on the Paraphrase task, and an accuracy of 0.759 on the STS task.

1 Key Information to include

- External collaborators (if you have any): None
- External mentor (if you have any): None
- Sharing project: No

2 Introduction

BERT, or Bidirectional Encoder Representations from Transformers, is a large language model based on the transformer architecture (Devlin et al., 2018). BERT is especially powerful when it comes to sequential textual data, and the pipeline for training such a neural network model involves pre-training on a large corpora of text and then fine-tuning it for a particular NLP task. We have seen BERT perform well on a wide variety of language tasks, however since many models are optimized during training for a particular task, the challenge lies in performing multiple tasks with high accuracy.

This could be a potential issue when we are working in low-data settings, because finetuning a large language model with few datapoints could lead to severe overfitting. Traditionally, this is countered by freezing earlier layers or by using more creative forms of data augmentation. However, more recent approaches involve bootstrapping the performance on a specific task by utilizing the data from other tasks which may have much more data available. This is especially useful when trying to perform well on further downstream tasks such as in the medical or education domains, where it can be challenging to collect the required quantity and quality of data due to privacy and regulation issues. By training in a multitask fashion with other similar tasks, we can mitigate the effects of not having enough data for a specific task.

Additionally, this topic is especially relevant given the increasing amount of text corpora in which tasks that are similar in nature can be performed. Ideally, we want training schemes that can leverage insights from individual learning tasks to enhance performance on multiple tasks as a collective.

This project works towards building a model that has strong performance across numerous tasks, namely sentiment analysis, paraphrase detection, and semantic textual similarity. Our main baseline approach entails training a multi-headed BERT, where each head is assigned a distinct task and loss function according to the desired output. This approach is then expanded upon through our three main expanded approaches which included gradient surgery using PCGrad and cosine-similarity fine-tuning. We also conducted extensive testing with a wide variety of hyperparameters to determine which combination would yield the highest performance overall across all of our experimental approaches. Finally, we created a novel approach to the problem by pretraining independent task-specific BERT models to generate stronger weights that are then used as a better initialization for multitask finetuning.

3 Related Work

This study seeks to build upon previous works that have been used for training multitask machine learning models, including models for natural language processing. A traditional issue in multitask learning has been destructive interference between the gradient updates for the different tasks. Destructive interference between the gradients means that the gradient updates for the different tasks are in conflicting directions, i.e. the cosine similarity between the gradients is negative. This would lead to a situation where the update needed to optimize for a certain task would largely cancel out the update needed to optimize for another task, and this will lead to a decrease in the overall performance of the model across all the tasks. Techniques like gradient surgery Yu et al. (2020) can project a gradient that has negative cosine similarity with another gradient onto the normal plane of that gradient, which will remove the conflicting component of the gradient.

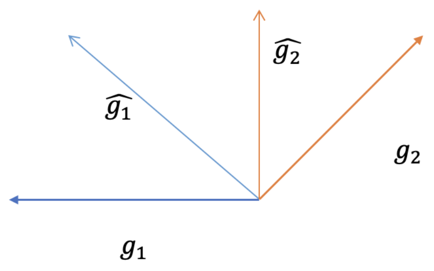


Figure 1: Example of conflicting gradients before and after application of PCGrad algorithm. g_1 and g_2 are the original conflicting gradients for task 1 and task 2 (negative cosine similarity), while $\hat{g_1}$ and $\hat{g_2}$ are non-conflicting (positive cosine similarity) after the application of PCGrad.

We also explored methods that could give us a good pretrained initialization point for our specific tasks, rather than using the general pretrained BERT parameters as our initialization point for multitask training. We first explored many meta-learning methods, such as optimization-based meta-learning methods and metric-based meta-learning methods. While the optimization-based techniques we looked at are designed to give us a good initialization of parameters that can quickly generalize to new tasks, the metric-based techniques we looked at are much simpler and could give us a good pretrained paremeter initialization that can be used for finetuning.

Model-Agnostic Meta-Learning (MAML) Finn et al. (2017) is a prime example of an optimization-based meta-learning method from the literature. MAML is used to train the parameters across multiple tasks during meta-training time in such a way that the parameters can then further be optimized quickly for new tasks at meta-testing time, as long as there is shared structure between the tasks at training time and inference time (i.e. the tasks should be drawn from the same underlying task distribution).

MAML uses an optimization process with an inner loop and outer loop, with normal gradient updates in the inner loop and meta-gradient updates in the outer loop (the gradient of the gradient). This means we will have to compute second-order derivatives, and this can only be done exactly by computing the full Hessian matrix. We can use first-order approximations to avoid computing the full Hessian with minimal accuracy tradeoff (Nichol and Schulman, 2018). We explored potential applications of optimization-based meta-learning techniques to our 3 tasks, and found that we would need to construct several dozen similar tasks that we could do meta-training on. Additionally, MAML shows the best performance in low-data regimes, so we decided to explore other techniques to use for our approach.

We also took a look a prototypical networks, which build representations of different classes in an embedding space and use Euclidean distance to classify new data points. The algorithm is simple, yet outperforms MAML on few-shot learning tasks Snell et al. (2017). However, prototypical networks in their original form only work with classification problems, so we would have to discretize the regression task into a classification task. Early experimentation showed that discretizing the task could not be done naively, because the parameters did not transfer well to the continuous version of the task.

We also examined cosine similarity Reimers and Gurevych (2019) as a method to finetune for the continuous tasks. Cosine similarity has been used for effective finetuning single tasks, so it seemed like a natural extension to apply it for some or all of the continuous tasks in a multitask setting.

4 Approach

We want a model that performs well in multiple tasks: sentiment analysis, paraphrase detection, and semantic similarity. To achieve this, we train a BERT model with a separate head for each of the tasks. For sentiment analysis, we need to predict different classes of sentiment, so we use Cross Entropy as our loss function, where $\hat{y}$ is a vector of predicted probabilities for each class, y is a one-hot vector of the ground truth label, and C is the number of classes.

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$

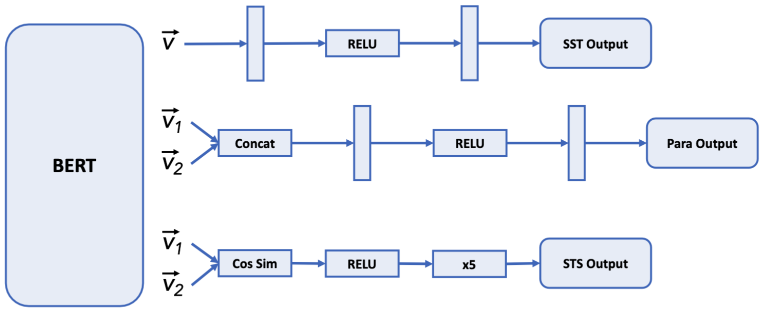
Paraphrase detection has a yes or no label as output (corresponding to either 1 or 0 respectively), so we use Binary Cross Entropy as the loss function for this task, where $\hat{y}$ is the predicted probability and y is the true label (either 0 or 1).

$$\mathcal{L}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y})$$

Semantic similarity has an output of a single number that represents the similarity of a sentence pair; thus, for this task we use Mean Squared Error as the loss function where $\hat{y}$ is the predicted value, y is the true value, and n is the length of the dataset.

$$\mathcal{L}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y_i} - y_i)^2$$

As our baseline strategy, we use BERT as a shared encoder between the three tasks. We then have task specific heads: for sentiment analysis, we have a linear layer followed by a ReLU non-linearity followed by another linear layer. We use this setup rather than a simple, single linear layer because this allows for the task specific head itself to encode non-linearities. For paraphrase detection, we concatenate the output of the shared encoder for the two sentences. This is then passed into a linear layer followed by a ReLU non-linearity followed by a linear layer. We originally had the same setup for the semantic similarity task head, but switched to a method where we take the output of the shared encoder for the two sentences and pass it through cosine similarity before applying a ReLU non-linearity and multiplying by 5. This switch is motivated in the Analysis section.



We have a training loop that at each step samples a batch from each of the three datasets. The loss for each task is calculated separately using the loss functions described above. Finally, we calculate the total batch loss as the sum of the three losses and use this to backpropagate.

We built upon cosine similarity fine-tuning by adding gradient surgery to counteract conflicting gradients between the three tasks. Specifically, we used the PCGrad algorithm as described in the paper by Yu et al. (2020) to project a gradient that has negative cosine similarity with another gradient onto the normal plane of that gradient. This allows for constructive interference between the gradients rather than destructive interference, which essentially means optimization for one task does not interfere (or destruct) the performance on another task.

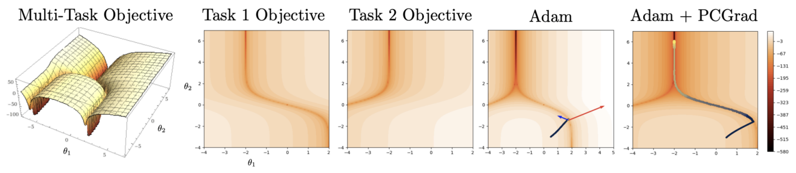
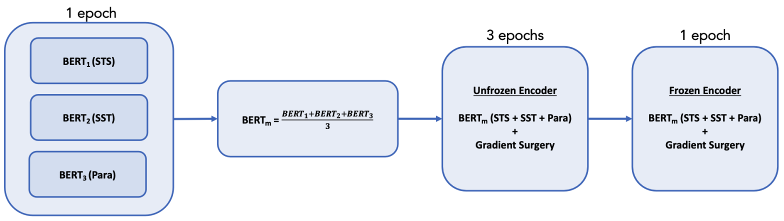


Figure 2: Depiction of PCGrad optimization given a multi-task problem. Figure from Yu et al. (2020).

Our last step was creating a novel 3-step training regime that maximized performance across the different methods mentioned above. We started from the general BERT model and finetuned with all parameters unfrozen on each task separately, essentially giving us the weights for 3 single task models. The parameters from each of the 3 BERT models are than averaged together to get an initialization point for multitask training that is empirically more effective than the general BERT parameters. We then further finetune on the 3 tasks jointly using the sum of the separate losses (with the optional addition of PCGrad), also with all parameters unfrozen, before jointly finetuning with only the non-shared layers unfrozen. This last step is to ensure that once the shared parameters have reached their maximum effectiveness, we can further optimize the task-specific parameters.



5 Experiments

5.1 Data

For the sentiment analysis task, the Stanford Sentiment Treebank dataset (Socher et al., 2013) was used. This dataset consists of 11,855 single sentences from movie reviews and is pre-processed using the Stanford parser (Chen and Manning, 2014) to result in 215, 154 unique phrases, each of which is annotated by 3 judges. The phrases are assigned labels of negative, somewhat negative, neutral, somewhat positive, or positive corresponding to a number between 0 to 4 respectively.

For the paraphrase detection task, a dataset from Quora (Iyer et al.) was used, consisting of 400,000 question pairs with accompanying labels that describe whether or not a pair of questions were paraphrases of each other. As such, the labels are binary.

Lastly, for the semantic textual similarity task, the degree of similarity between a pair of sentences is measured with a label that is on a scale from 0 (unrelated) to 5 (same meaning). The SemEval STS Benchmark dataset was used, consisting of 8,628 different sentence pairs and their accompanying labels.

The default train/dev/test split was used in all cases.

5.2 Evaluation method

For each of the three tasks, different evaluation methods were used. For the sentiment analysis and the paraphrase detection tasks, accuracy was used as the evaluation metric. Accuracy was a suitable metric for these tasks as it simply signifies the fraction of correct classifications made by the model. However, for the semantic textual similarity task, the Pearson correlation between the true and predicted similarity values are used to test the dataset as conducted in the default project provided code.

5.3 Experimental details

We ran a wide variety of trials to experimentally determine the hyperparameter combinations that performed the best across all three tasks. Our final hyperparameters (that were then used across all architectural experiments) were using a learning rate of $1e-5$, a batch size of 16, either low or no dropout regularization depending on the experiment. Our novel training scheme consisted of 1 epoch of single-task pretraining followed by 3 epochs of multitask finetuning with the encoder unfrozen, which is followed by 1 final epoch of multitask finetuning with a frozen encoder. The previously mentioned hyperparameters were used for all stages of the training pipeline. During training, the total number of iterations per epoch is determined by the length of the maximum dataset divided by the batch size. Experimentation was run using a NVIDIA A40 GPU which resulted in training time of 20 minutes per epoch for the baseline implementation as well as the cosine similarity implementation and 30 minutes upon adding gradient surgery.

5.4 Results

As mentioned in the Approach section, we consider the combined loss (sum from the three tasks and backpropagate accordingly) as our baseline approach. Our first step was to naively implement gradient surgery on the baseline model. This ended up performing worse than the baseline and we moved to implement cosine-similarity which ended up drastically increasing our performance on hte paraphrase and STS tasks, while the sentiment task remained roughly the same. Using our findings from cosine similarity, we combined this with gradient surgery to improve performance further. Our best performing model at this stage utilized this approach, and was found to produce the best results when dropout regularization was removed. Our quantitative results for the baseline model and experimental approaches are depicted in the table below.

Model	SST dev Acc	Paraphrase dev Correl	STS dev Acc	Overall dev Score
Baseline	0.494	0.386	0.325	0.402
Gradient Surgery (GS)	0.444	0.380	-0.003	0.274
Cosine Similarity (CS)	0.497	0.753	0.600	0.617
CS + GS	0.494	**0.785**	0.635	0.638
CS + GS, no dropout	**0.510**	0.781	**0.739**	**0.677**

We then expand on these results using our described three step approach, which involved single-task pretraining which was then followed by multitask finetuning. These next steps also utilized the findings from our earlier experiments and thus used both cosine and gradient surgery. The results for this modified approach can be seen in the table below. We hypothesized this to perform better than our initial approach, and this theory was supported by the quantitative results achieved by the model.

Novel Approach	SST dev Acc	Paraphrase dev Correl	STS dev Acc	Overall dev Score
CS + GS	0.517	0.814	0.762	0.698
CS + GS, no dropout	**0.528**	**0.821**	**0.776**	**0.708**

This final model, which performed the best in the dev set context, was then submitted to the test leaderboard as our final submission. The resulting performance can be seen as follows.

Novel Approach	SST test Acc	Paraphrase test Correl	STS test Acc	Overall test Score
CS + GS, no dropout	0.522	0.824	0.759	**0.702**

6 Analysis

From our results, we can see that baseline had a lot of room for improvement. We noticed from the training output that after a few epochs of training, the performance on the SST and Paraphrase dev datasets continued to improve while the performance on the STS dev set began to stall and decrease. Our theory was that this was due to conflicting gradients between the tasks, specifically between the SST/Paraphrase and STS tasks. We then implemented gradient surgery to counter this issue, because gradient surgery is meant to reduce the conflicting gradients between tasks by projecting the gradient on to the normal plane of the conflicting gradients. However, this did not change our performance on the SST or Paraphrase tasks but ended up damaging our results on the STS task.

Upon manually inspecting the gradients produced for the 3 tasks, we found that after a few iterations of training, the gradients for the third task were close to orthogonal with the first 2 tasks. Applying PCGrad to these gradients ended up almost annihilating the gradient for the STS task. We wanted to make the head for the STS task use non-learnable parameters because we theorized that the gradients backpropagating through the learnable linear head of the STS task were the reason for conflicts with the other two tasks. We thus decided to change our prediction head for the STS task to try and align the optimization goal more with the first two tasks.

We thought that cosine similarity would be a good approach for this task because we are trying to predict the similarity of two sentences, and these sentences can easily be reduced to 2 vectors through a BERT embedding scheme. We were able to use the vanilla BERT encoder to generate sentence vectors, then pass these through cosine similarity before multiplying by 5 to align the output with the specifications of the task. This lead to greatly improved performance as seen in row 3 of the results table. We noticed that even with this new task-specific head for the STS task, after a few epochs the STS/Paraphrase tasks would continue to increase in performance while the STS performance would stall.

After manually examining the gradients and finding that they were not as conflicting as before the addition of cosine similarity, we decided to try applying gradient surgery again, which led to improved results on the STS task. As expected, this wasn't a complete success because projecting the gradients using PCGrad actually led to some negative effects on the gradients for the SST task, leading to slightly decreased performance. However, the tradeoff was effective because we saw large improvements in both the Paraphrase and STS tasks.

Finally, we decided to see if we could get improved results by starting at a better initialization point than the general pretrained BERT model. Traditionally, multitask learning would be used to find a good initialization point for a single task model that we are trying to finetune downstream. However, in this case we are explicitly trying to get a good multitask model, and we want to share parameters between the tasks rather than employing the trivial solution of training a model that consists of 3 single task models with no shared parameters. Thus, we decided to initially train 3 separate models independently on the tasks, then average the parameters of these 3 models to use as the initialization point for the multitask model that was previously described. As expected, this led to improved results over the general BERT initialization, as shown in the results.

7 Conclusion

Throughout the course of this project, we were able to empirically determine certain modifications to our baseline multitask BERT implementation that resulted in stronger task-specific and overall performance. The combination of cosine-similarity fine-tuning alongside gradient surgery to mitigate the destructive impacts of conflicting gradients from different tasks resulted in better model performance. We were able to improve on this model further by modifying our training approach, adopting a strategy that trained three separate models (one for each task) and then using this as the initialization for multitask finetuning, which was our optimization objective. This ended up being our highest performing approach and final model. We also conducted hyperparameter tuning and removed dropout regularization, both of which ended up improved our results across all experimental approaches.

Prior to this project, we had never trained a system to solve multiple tasks, so this was a big learning for us. We also learned more about the pretraining and finetuning pipeline, and ways that these ideas could potentially be modified depending on the desired output, as we showed through our final approach to the problem. We also initially thought that gradient surgery would be beneficial to our baseline model, but our hypothesis was disproven based on the resulting performance. However, after better understanding the benefits of gradient surgery and the specific cases in which it is found to be powerful, using it on top of cosine similarity to enhance performance was a major step forward in our understanding and results.

To discuss limitations, all models seemed to perform a fair bit better on the paraphrase detection and semantic textual similarity when compared to sentiment classification. As a result, the average of scores across tasks is pulled down greatly by the poor performance on the sentiment task. Another limitation is that our cosine similarity approach was only to the benefit of the STS task and that we did not try any other task-specific approaches for the remaining tasks. This could potentially be a major point of improvement and result in performance gains across the board, giving us a more robust overall system.

One of the main things we would like to try in the future is an optimization based meta-learning approach such as MAML. Based on the work we did in this project, we saw that having a good initialization point before finetuning for our tasks downstream improved our performance on these downstream tasks. MAML would be great to try because it is explicitly optimizing for an initialization point such that we can learn these downstream tasks as quickly as possible. We didn't have enough time to construct enough related tasks to use during meta-training, but this is something we would like to try out in the near future.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. Quora duplicate question pairs dataset.

Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

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

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