Multi-Dimensional BERT: Bridging Versatility and Specialization in Multi-Task Learning

Stanford CS224N Default Project

Emma Casey, Luke Moberly
Department of Computer Science
Stanford University
{emcasey, lmoberly}@stanford.edu

Abstract

We implement an extension of the BERT model with architectural changes and learning and regularization techniques to optimize performance on three different tasks: sentiment analysis using the Stanford Sentiment Treebank (Socher et al., 2013 [1]); paraphrase detection on Quora Question Pairs (Fernando and Stevenson 2008 [2]), and semantic equivalence (Agirre et al., 2013 [3]). Specifically, we implemented the S-BERT architecture [4] to improve the performance of the paraphrase detection task, PCGrad [5] to reduce performance trade-offs, and SMART regularization [6] to address overfitting on STS and SST tasks. Additionally, we experiment with different loss functions (MSE and cross-entropy) for the STS task, as well as various hyperparameters (learning rate, dropout, weight decay, and batch size) and optimizers (AdamW, Adam, and SGD). Our approach effectively balances versatility and specialization in multi-task learning, which will become increasingly important as the complexity and size of language models continue to scale.

Key Information: Our TA mentor is Neil Nie. We have no external collaborators or mentors, and we are not sharing the project. We contributed equally to this project. Luke implemented the baseline BERT model and made stronger contributions to the code base, while Emma implemented optimizations and focused more on writing and analysis.

1 Introduction

Language models are increasingly expected to excel across a large, diverse set of tasks, as shown by the rise of benchmarks like MMLU (Massive Multitask Language Understanding). This reflects the belief that models should mirror a general, human-like form of intelligence rather than focus on performance for more specific, narrowly defined tasks.

In our project, we addressed a fundamental challenge of multi-task learning: balancing generalization while maintaining strong performance on specialized tasks, namely sentiment analysis, paraphrase detection, and semantic equivalence. We implemented various learning, regularization, and architectural techniques, making incremental changes based on the results of experiments. Ultimately, we arrived at our highest-performing model by implementing the S-BERT representation of multi-sentence embeddings, with various hyperparameter adjustments and the use of MSE loss. Surprisingly, we found that more advanced techniques, such as PCGrad and SMART regularization, did not substantially improve results, especially relative to architectural changes.

2 Related Work

Multitask learning (MTL) is a paradigm that leverages shared representations to solve multiple tasks with the same model, realizing large computational efficiency gains and generalizability relative to independent task training.

One of the first unified MTL architectures was proposed in 2008 by Collobert and Weston[7], who achieved state-of-the-art performance by training jointly on multiple tasks by using weight-sharing. A few years later, Turian et al.[8] and Dai and Le 2015 [9] showed that word features can be learned in an unsupervised, task-agnostic manner and improve in generalization accuracy on a variety of downstream tasks. Similar promise has been seen with transfer learning from supervised data, particularly in vision research. Yosinski et al. 2014 [10] showed the effectiveness of fine-tuning models after pre-training on ImageNet.

However, MTL has also proven difficult to optimize due to issues such as overfitting and conflicting gradients. Overfitting can arise when fine-tuning on downstream tasks is too aggressive, leading the model to overfit and perform poorly on unseen data. Dropout, proposed by Hinton et. al. in 2014 [11], is a regularization technique that randomly drops units in the hidden layer to zero to prevent over-indexing on any one unit to encourage greater generalization. More recently, Jiang et al. (2020) [6] propose SMART regularization, which uses adversarial regularization and Bregman Proximal Point Optimization (BPPO) to manage complexity and aggressive updating at the fine-tuning stage. Yu et al., on the other hand, attribute the primary issue to the conflicting gradient problem, defining gradients to be conflicting if they have negative cosine similarity. More specifically, they find that such conflicts are detrimental to optimization when they coincide with high positive curvature in the optimization landscape and are significantly different in magnitudes. Under these circumstances, the multi-task gradient is dominated by a singular task gradient, resulting in the improvement of the dominating task being over-estimated. This is shown visually in Figure 1

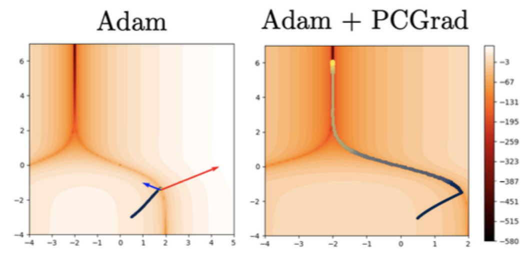

Figure 1: Visualization of 2D multi-task optimization. The gradient vectors of the two tasks are indicated by the blue and red arrows. Optimization trajectory goes from black to yellow. Without PCGrad, optimization stops in the first valley (at the intersection of the gradient vectors).

We attempt to combine different approaches to MTL optimization discussed above, using adjusted representations (i.e. absolute difference) of embeddings, gradient projection methods, and robust regularization to tackle the challenge of MTL.

3 Approach

We first implemented a very basic multi-task model, with only linear layers projecting the BERT embeddings onto the task-specific class space (five classes for sentiment classification and two classes for paraphrase detection, with cosine similarity being used for semantic textual similarity). This did rather poorly, performing slightly above random guessing for sentiment classification and under for paraphrase detection. We implemented five experiments, detailed below.

1. **S-BERT + Cosine Similarity**
   On the baseline, both the paraphrase detection and sentence similarity tasks were performing poorly (paraphrase detection was below the 0.5 accuracy of random guessing). We implemented the S-BERT architecture as outlined by Reimers and Gurevych [4] for the paraphrase detection task, shown in Figures 2 and 3. We also added cosine similarity for the STS task.

Rather than simply concatenating the two pooled embeddings from A and B and running that through a classifier, we concatenated the embeddings with the element-wise absolute difference $|u - v|$ and multiplied that with weights $W \in \mathbb{R}^{3n}$, projecting onto a binary subspace (paraphrase or no paraphrase). At the same time, because sentiment classification was more distinct from the other two tasks, we added a few task-specific layers to the architecture (two hidden layers with dropout and activation functions). Additionally, we experimented with adding additional full-connected layers with dropout and activation functions to the standard S-BERT architecture (after pooling).

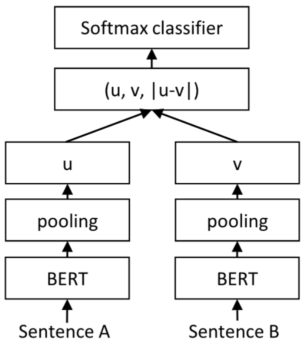

Figure 2: Training Architecture

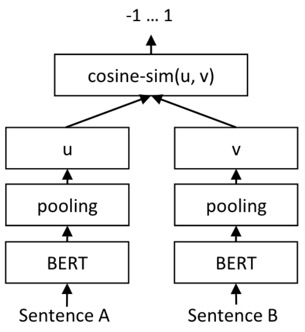

Figure 3: Inference Architecture

2. **PCGrad**

   As mentioned in Section 2, one issue with MTL is conflicting gradients. During our hyperparameter experiments with the S-BERT architecture, we noticed trade-offs (sometimes rather large) between the three tasks. For example, as SST or STS improved, paraphrase detection performance would worsen. As discussed by Yu et al. [5], this is sometimes the result of conflicting gradients. Using an existing implementation of PCGrad [12], we incorporated their proposed solution of PCGrad into our architecture, projecting the conflicting gradient onto a surface acute to the other gradient, to attempt to address this problem. Specifically, for two task gradients g_i, g_j, when $\cos \phi_{i,j} < 0$, then:

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

   For three tasks, the process is repeated for each task and then summed, such that $g^{PC} = \sum_i g_i^{PC}$. This process ensures that the gradients of any individual task interfere minimally with the gradients of other tasks.

3. **SMART Regularization**

   We observed severe overfitting with both the S-BERT and S-BERT + PCGrad implementations, especially on the STS and SST tasks. In some cases, dev accuracies were twice that of validation accuracies (e.g. 0.9 vs 0.4 for SST). As outlined by Jaing et al. [6], aggressive fine-tuning on downstream tasks can cause model overfitting. We used the public implementation of SMART regularized optimization by Jiang et al. to address the overfitting in SST and STS fine-tuning. We made a small modification to their implementation by dynamically changing the amount of regularization based on the degree of overfitting for each task.

4. **Changing Loss Functions**

   Although MSE loss is common for regression tasks, and thus theoretically better suited for the semantic similarity task, we wanted to experiment with our model sharing the same loss function (cross-entropy). It is important to keep gradients on the same scale when implementing multi-task learning, and having different loss functions can cause different gradient scaling, affecting the optimization. We used both MSE and cross-entropy loss on STS in each of the above architectures, noticing different results for all three.

5. **Hyperparameter Tuning**

On each of the above architectures, we did a hyperparameter sweep of different learning rates, dropouts, and weight decays. We also experimented with different optimizers (AdamW, Adam, and SGD), but found that AdamW performed best across the board.

4 Experiments

Data & Evaluation Method We used the Stanford Sentiment Treebank (SST) and CFIMDB dataset in the first part of the project. In the second, we used the SST dataset for sentiment analysis, the Quora dataset for paraphrase detection, and the SemEval dataset for semantic textual analysis. For sentiment analysis and paraphrase detection, we use accuracy as our measure of evaluation. For textual similarity, we use the Pearson correlation coefficient.

Baselines Our baseline architecture added one layer for each task (a linear layer projecting down onto the appropriate output space and an activation function) with the standard hyperparameters found in the first part of the project (learning rate = 1E-5, batch size = 8, epochs = 10). We achieved the following scores.

Model	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
minBERT	0.422	0.290	0.373	0.205

Table 1: Baseline Results

The baseline model performs very poorly, with SST accuracy just above random guessing (0.2) and paraphrase accuracy below random guessing (0.5).

Experimental Details We initially implemented each of the first four experiments with the same baseline hyperparameter settings (learning rate = 1E-5, batch size = 32, epochs = 10, dropout = 0.1, weight decay = 0.01). We then ran a hyperparameter sweep on each model to obtain the best settings. We decided to experiment with learning rates, dropout probabilities, and weight decay. As mentioned earlier, we also ran experiments with various optimizers (AdamW, Adam, SGD), but found that AdamW worked best across the board.

5 Results & Analysis

S-BERT + Cosine Similarity The implementation of S-BERT for paraphrase detection, cosine similarity on STS, and sentiment-specific layers for SST dramatically improved our baseline. As shown in Table 2, we saw improvements to both paraphrase and STS when using cosine similarity architecture on STS, and a larger performance jump for paraphrase when adding S-BERT. The sentiment-specific layers (specifically two hidden fully-connected layers with dropout and ReLU activation) also notably boosted SST performance.

Model	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
STS Cosine Similarity	0.636	0.497	0.598	0.626
S-BERT (para) + CosSim	0.687	0.486	0.771	0.610

Table 2: Initial S-BERT Experiments with Default Hyperparameters

We experimented with different variations of the embedding concatenation method. Like Reimers and Iryna Gurevych, we found that the concatenation mode used by Cer et al. in the Universal Sentence Encoder [13], in which $(u, v, |u - v|, u * v)$ is used as input to the classifier, slightly decreased performance relative to concatenating the embeddings with absolute difference alone. One possible explanation is that element-wise multiplication can be dominated by large values in either vector and lead to less stable gradient updates.

After running a hyperparameter sweep on this architecture, we found that the two most effective parameters to tune were weight decay and dropout, most likely because the sentiment and similarity tasks were overfitting. Increasing dropout had a fairly large effect on paraphrase detection and sentiment classification, while weight decay had large effects on sentiment classification and similarity. This is shown in Table 3.

Hyperparameters	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
Weight Decay: 0.1, Dropout: 0.3	0.687	0.486	0.771	0.610
Weight Decay: 0.1, Dropout: 0.5	0.718	0.482	0.826	0.694
Weight Decay: 0.3, Dropout: 0.3	0.720	0.500	0.782	0.758
Weight Decay: 0.3, Dropout: 0.5	0.720	0.510	0.784	0.731

Table 3: Experiments with Hyperparameters on SBERT Architecture

PCGrad During the hyperparameter sweep of the SBERT architecture (which included the layers added to the sentiment task), we noticed that the improvements in SST and STS reduced the performance of paraphrase detection. This is common in MTL optimization; the improvement of the dominating tasks is over-estimated while poor optimization of secondary tasks is under-estimated. As discussed in section 3, Yu et al. propose PCGrad as a method to fix conflicting gradients and improve optimization across all tasks in MTL [5]. We believed that adding PCGrad could help reduce the performance hit that the paraphrase task took when increasing weight decay. However, we were surprised to find that it didn't lead to any improvements, as shown in Table 4. Rather, in many cases, it performed worse than the simple S-BERT architectures. As expected, PCGrad diminished performance in SST, but we were surprised to find that it also harmed results in paraphrase detection, as that was supposed to be the secondary task that would benefit from adding PCGrad.

One possible reason may be that we had to use slightly smaller batch sizes (16 as opposed to 32), because of memory issues on our virtual machines. We tried various implementations of PCGrad, but each ran into memory issues. Future work could include utilizing different virtual GPU instances that allow for larger batch sizes when using PCGrad. However, in our initial baseline experiments, we didn't notice significant score differences between batch sizes, and STS and SST overfitting wasn't significantly different on the PCGrad architecture compared to the S-BERT architecture. Thus, it is unlikely that the batch size difference significantly hindered the performance of PCGrad.

Model	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
S-BERT (best)	0.720	0.510	0.784	0.731
S-BERT + PCGrad	0.707	0.489	0.767	0.733

Table 4: Initial S-BERT + PCGrad Experiments with Default Hyperparameters

Importantly, on some hyperparameter settings (such as weight decay = 0.1 and dropout = 0.3), S-BERT + PCGrad performed more optimally than S-BERT alone. However, the most optimal hyperparameters found for S-BERT alone were more optimal than when adding PCGrad. This is shown in Table 5.

Hyperparameters	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
Weight Decay: 0.01, Dropout: 0.1	0.457	0.253	0.632	-0.030
Weight Decay: 0.1, Dropout: 0.3	0.705	0.489	0.766	0.717
Weight Decay: 0.3, Dropout: 0.3	0.707	0.489	0.767	0.733

Table 5: Experiments with Hyperparameters on S-BERT + PCGrad Architecture

SMART Regularization We implemented SMART regularization in an attempt to reduce overfitting on the SST and STS tasks. We did see an improvement in the STS task, but it came with an equally significant reduction in improvement on SST, as shown in Table 6.

However, in both cases, SST and STS continued to overfit drastically. We were seeing dev accuracies of 0.95+ for both tasks, with validation fold performance stagnating at ~ 0.51 and ~ 0.76, respectively. This can be clearly seen in Figure 4 The issue of overfitting plagued many of our experiments. Adding SMART, changing model architecture, increasing dropout and weight decay, and increasing batch size did little to affect overfitting. We realized too late that the next step might be to try data augmentation. We were so focused on changing the model to reduce overfitting that we hadn't considered the data to be the primary culprit. Future work would include additional data generation, possibly using a tool like GPT-4, or masking on the original dataset.

Model	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
S-BERT (best)	0.720	0.510	0.784	0.731
S-BERT + SMART (best)	0.720	0.488	0.766	0.817

Table 6: S-BERT + SMART Experiments with Best Hyperparameters

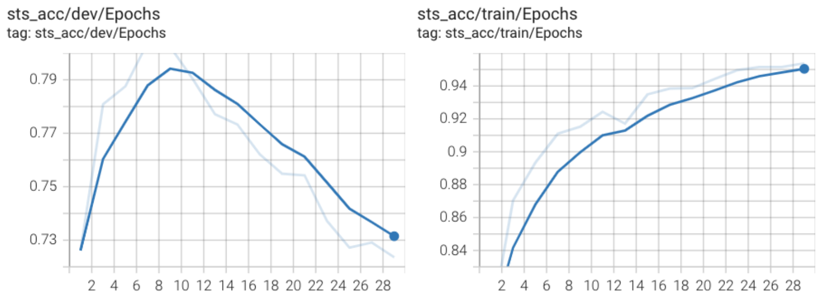

Figure 4: Overfitting on SST: Dev vs Validation Accuracy

Changing Loss Functions We had initially standardized our loss function across all three tasks as cross-entropy loss. This was done to maintain similar scale across gradients. However, we overlooked the common use of MSE on regression tasks (which STS was), as opposed to the use of cross-entropy on classification tasks (of which both SST and paraphrase detection are). We saw some improvements in the STS task, but the biggest jump in paraphrase, as shown in Table 7. Although initially surprising, we noticed that when using cross-entropy loss, the loss for STS was much larger than that of the other two tasks (often twice as large). It is possible that is dominated the optimization, reducing performance on the other tasks. Thus, when switching to MSE loss, it actually resulted in the gradients and losses for all three tasks to be on a more similar scale, boosting performance on paraphrase detection. Because MSE loss is more resilient to outliers, it is better suited for regression tasks with more noisy data. We did not attempt this configuration with PCGrad, but future work could look into doing so, as this may be a case of conflicting gradients.

Hyperparameters	Overall Score	SST Acc (Dev)	Para Acc (Dev)	STS Corr (Dev)
Weight Decay: 0.3, Dropout: 0.5	0.732	0.512	0.803	0.760
Weight Decay: 0.3, Dropout: 0.3	0.730	0.456	0.829	0.810

Table 7: Experiments with Hyperparameters on SBERT + SMART Architecture + STS MSE Loss

6 Conclusion

In this work, we investigated the challenge of balancing versatility and specialization in multi-task learning using a BERT-based architecture. Our primary focus was on three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Through a series of experiments, we introduced several advanced techniques, including S-BERT, PCGrad, Cosine Similarity, and SMART regularization, alongside hyperparameter tuning.

Our results demonstrated that the S-BERT architecture, coupled with STS-cosine-similarity and SST-specific layers, significantly outperformed the baseline model, particularly in paraphrase detection and semantic textual similarity tasks. Hyperparameter tuning further enhanced the performance, with optimal configurations leading to substantial gains. Interestingly, while PCGrad was expected to mitigate the conflicting gradient issue inherent in multi-task learning, it did not yield the expected improvements and, in some cases, resulted in decreased performance. Similarly, SMART regularization did little to address the issue of overfitting.

These findings suggest that simple architectural adjustments and meticulous hyperparameter tuning can substantially enhance multi-task learning performance, and techniques like PCGrad and SMART regularization may only lead to modest improvements with this foundation in place.

Overall, our study underscores the importance of iterative experimentation and careful evaluation in multi-task learning. As language models continue to scale in complexity and scope, balancing versatility and specialization remains a crucial challenge, and our findings contribute valuable insights towards achieving this balance.

7 Limitations and Future Work

Moving forward, our highest priority would be addressing the overfitting seen on SST, which was not sufficiently corrected with SMART regularization or increasing the dropout probability. As mentioned earlier, we would want to consider additional data generation, possibly using a tool like GPT-4, or implementing masking on the original dataset.

After addressing overfitting, we would want to enhance our ability to handle conflicting gradients. We believe PCGrad may have been limited in its approach due to the memory issues we encountered, limiting our batch size and the possible number of epochs run. To address the memory issues we encountered with PCGrad, we will explore strategies such as gradient checkpointing and memory-efficient implementations of gradient projection. Gradient checkpointing reduces memory usage by saving only certain layers' activations during the forward pass and recomputing them during the backward pass. This technique allows us to handle larger models and batch sizes without running out of memory. By focusing on these areas, we aim to improve upon our baseline to develop a robust and scalable multi-task learning model.

8 Ethical Considerations

In addressing tasks like sentiment analysis and semantic textual similarity, we are dealing with complex and culturally sensitive language data.

Given the reliance on datasets such as the Quora dataset, Stanford Sentiment Treebank, and STS 2012 and 2013 datasets, there is an inherent risk of bias, particularly against underrepresented dialects such as African-American Vernacular English (AAVE). More specifically, in the SEM 2013 paper from Diab, Gonzales-Agirre, and Guo, a dataset called Europeana was used for cultural heritage items. While this might be sufficient if use of the model was limited to Europe, the lack of training data could impair model performance across different linguistic and racial groups, a concern supported by Hofmann et al. [14]. This is particularly pertinent to our paper, as even if our model can generalize across tasks, if it cannot perform well across linguistic groups due to a lack of representation in training data, it is not right to call it a truly versatile model.

Some ways to address this are using fairness metrics for dialectic diversity (e.g., multi-group fairness) and augmenting the corpus to include a broader set of texts, such as those from AAVE to enhance to representativity of the model (e.g., use an African American cultural heritage dataset for STS). Lastly, as is the case in all NLP research, it is important to rigorously test for the amplification of biases (disparities in how the model handles content related to different genders, ethnicities, or other characteristics) and maintain transparency in our methods to facilitate external audits.

References

[1] 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*, pages 1631–1642, 2013.

[2] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*, pages 45–52, 2008.

[3] 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*, pages 32–43, 2013.

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

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

[6] 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.

[7] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[8] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. ACL, 2010.

[9] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

[10] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. Submitted 11/13; Published 6/14.

[12] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[14] Valentin Hofmann, Pratyusha Ria Kalluri, Dan Jurafsky, and Sharese King. Dialect prejudice predicts ai decisions about people's character, employability, and criminality, 2024.