BERT Goes to College: Exploring additional pretraining and multitask fine-tuning strategies with minBERT

Stanford CS224N Default Project

Michelle Fu, Peyton Lee, Eric Zhang
Department of Computer Science
Stanford University
{mifu67, peytonl7, zhyzhang} @stanford.edu

Abstract

Though the depth of the BERT model is key to its computational power, BERT's breadth of knowledge is not to be underestimated. Specifically, its ability to adapt to multiple downstream tasks calls for a thorough and expanded investigation into the possibilities of multitask fine-tuning, in which multiple tasks are optimized simultaneously after pretraining. Our objectives in this project were to implement the minBERT model and experiment with multitask fine-tuning paradigms to improve performance across three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. In our research we compare a number of different strategies, including sequential training, basic multitask training, and two algorithms of our own design (weighted and stochastic). Additionally, we highlight techniques and training methods that seem crucial to accomplishing the provided tasks, including contrastive learning and additional pretraining. Ultimately, we find that multitask fine-tuning (including stochastic and weighted methods) provides a small improvement on other downstream fine-tuning methodologies and more balanced performance across tasks, though other facets of our implementation—like training on additional datasets and freezing weights—are promising areas for the perfection of multitask fine-tuning.

1 Introduction

BERT—or Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)—needs no introduction. Since its conception in 2018, the BERT model's approach to contextual word embeddings has run the gamut of computer science research, finding foundational uses in everything from computer vision (Khan et al., 2022) to predicting protein structure (Jumper et al., 2021). It may still be considered state-of-the-art, and the generalizability of its architecture makes it an exciting playground for researchers of all experiences.

Our task is to optimize the minBERT model[1] for multiple simultaneous downstream tasks, i.e. using a single model to accomplish different NLP objectives with the same weights. In theory, such a model could provide proof of humanistic intelligence within the system if it could synthesize nuanced information about language and apply it across many domains. If assigned dozens of tasks, such a model would also constitute huge memory savings.

Our three provided tasks are sentiment classification, paraphrase detection, and semantic textual similarity. Sentiment classification involves detecting the overall tone (positive or negative) in a statement. Paraphrase detection involves determining if two statements have equivalent semantic meaning. Semantic textual similarity involves ranking the level of semantic equivalence between two statements.

The most straightforward solution we encountered in deciding what to implement was *multitask fine-tuning*, in which we simultaneously fine-tune on multiple tasks to improve cumulative performance. The most general method (Bi et al., 2022), which we call *additive multitask fine-tuning*, simply adds each task's loss into a cumulative loss, which is then optimized with gradient descent. This method often fails because tasks have conflicting gradients or because fine-tuning destroys knowledge gained from pretraining. (These two problems, in general, constitute the main difficulties in multitask fine-tuning.) In this paper, we propose and test two new methods for multitask fine-tuning: *stochastic multitask fine-tuning* and *weighted multitask fine-tuning*. Both methods seek to solve the issue of conflicting gradients by forcing progress, and both are described in detail in section 3.6.

To accomplish all these tasks at a high level, we decide to implement a number of extensions to the minBERT model that we believe are essential to high performance. The first is *additional pretraining*, in which we further optimize the base model by performing masked language model training on the domain-specific datasets. This calibrates the model to the distributions of the target data. The second is *contrastive learning*, which involves pushing sentence embeddings that are semantically similar close together and pushing apart sentence embeddings that are semantically different. Finally, we consider *additional datasets* for pretraining and fine-tuning, as access to more data should, in theory, improve the model's underlying knowledge (about language modeling) and its explicit knowledge (about the tasks at hand). Dataset information is detailed in section 4.1.

2 Related Work

The bases for multitask fine-tuning with BERT models, as best we could determine, are Stickland and Murray (2019) and Bi et al. (2022). Stickland and Murray use additional layers inserted on top of and into the BERT architecture, each corresponding to a task. These layers are then trained (one task at a time) in a scheme that initially prioritizes tasks with more training data, then evens out among tasks toward the end of training. Bi et al. build on this training scheme by capitalizing on the fact that all their tasks share the same data: at *every* training step, they compute a loss for each task and sum them to create a cumulative loss for optimization.

Many studies use a sampling strategy, fitting to applications like biomedical text mining (Peng et al., 2020) and summarization (Lamsiyah et al., 2023). Other studies share the additive loss strategy used by Bi et al., fitting to applications like product ranking (Wu et al., 2022) and translation quality estimation Kim et al. (2019). Our work necessitates squaring these two approaches; in other words, we must be able to combine losses even if the tasks have different training datasets.

Bi et al. also implement a form of Gradient Surgery, developed by Yu et al. (2020). This model-agnostic method seeks to resolve the problem of conflicting gradients by performing a clever projection between the gradients of opposing tasks. Its success inspired us to find alternative solutions to this issue that require less computation, hence stochastic and weighted multitask fine-tuning.

A final study worth mentioning is Goodwin et al. (2020), which examines numerous sampling methods (what they call "mixing") that seek to simultaneously adjust for variant dataset sizes, the difficulty of tasks, and the distribution of task improvement. Though we cannot apply these methods to our study, which is not sample-based, they present a promising approach that addresses many of the critical failures associated with multitask fine-tuning.

3 Approach

3.1 minBERT and Adam

As specified, we implement the minBERT model using multi-headed self attention, position-wise feed-forward networks, and add/norm in each of the stacked transformer layers. We also implement

[1]The minBERT model is adapted from the "minbert" assignment developed at Carnegie Mellon University's CS11-711 Advanced NLP, created by Shuyan Zhou, Zhengbao Jiang, Ritam Dutt, Brendon Boldt, Aditya Veerubhotla, and Graham Neubig.

the Adam optimizer for training. For the remainder of the study, we use the provided pretrained BERT embeddings and weights as a starting point.

3.2 Additional pretraining

We do additional pretraining on domain-specific datasets, conducting unsupervised masked language modeling as in Devlin et al. (2018) and Sun et al. (2019). To create a dataset, we concatenate all tokenized input sentences, then split the concatenated tokens into chunks of length 128. Then, we randomly mask 15% of the tokens. As we are performing masked language modeling (MLM), the labels to this data are the (unmasked) tokens themselves.

To train on an MLM dataset, we add a single linear layer to the BERT model. This layer takes in the embeddings from the final hidden state of the BERT model and projects the embeddings to the size of the vocabulary. We return the linear layer's output as the logits.

3.3 Fine-tuning on additional datasets

Other than pretraining on additional datasets, we also did additional fine-tuning on the SNLI (Bowman et al., 2015) and MultiNLI datasets (Williams et al., 2018), which are collections of sentence pairs annotated with textual entailment information. We believe that fine-tuning on these datasets will help the model learn better sentence representations, and should be especially helpful for the semantic similarity and paraphrase detection tasks, which are closely related to textual entailment.

To train on these datasets, we add a single linear layer on top of BERT. This layer takes in a concatenation of the two [CLS] token embeddings h_1, h_2 from BERT and their element-wise difference $|h_1 - h_2|$. The input to this layer is:

$$h_i = [h_1 \quad h_2 \quad |h_1 - h_2|] \tag{1}$$

We use cross entropy loss to evaluate the output logits.

3.4 Contrastive learning

To learn better embeddings when fine-tuning on SNLI and MultiNLI, we also experiment with using a contrastive learning framework, which aims to push sentences that are semantically similar closer together in representational space and make sentences that are semantically different far apart. We follow Gao et al. (2021)'s approach for supervised contrastive learning. In the NLI dataset, every premise sentence is given an entailment sentence and a contradiction sentence. For a batch of premise sentence embeddings $h_1, \ldots, h_N$, we have corresponding entailment embeddings $h_1^+, \ldots, h_N^+$ and contradiction embeddings $h_1^+ -, \ldots, h_N^-$. Then for each sentence embedding h_i, we minimize the following loss function

$$-\log \frac{\exp(sim(h_i, h_i^+)/\tau)}{\sum_{j=1}^{N} \left(\exp(sim(h_i, h_j^+)/\tau) + \exp(sim(h_i, h_j^-)/\tau)\right)}$$

where N is the batch size, τ is a temperature parameter and sim is the cosine similarity between embeddings. We set $\tau = 0.5$ following Gao et al. (2021).

3.5 Fine-tuning and predicting on given downstream tasks

Here, we describe the model's training and prediction scheme for each of the downstream tasks using the provided SST, Quora and STS datasets.

Sentiment analysis Our model uses a linear layer and a dropout layer on top of the BERT model. These layers transform an output sentence embedding from BERT and return a vector of 5 logits (one for each possible label). In training, we use cross-entropy loss to evaluate the logits.

Paraphrase detection Our model uses a single linear layer on top of the BERT model. This layer takes the same structure as the SNLI layer (equation (1)), taking in a concatenation of the two output sentence embeddings from BERT and the absolute value of their element-wise difference. In training, we use cross entropy loss to evaluate the logits.

Semantic textual similarity Our model simply computes cosine similarity between the two output sentence embeddings from BERT, then scales the result between $[0, 5]$. In training, we use mean squared error loss to evaluate the logits.

3.6 Multitask fine-tuning[2]

Our goal is to compute a loss for every task at every training step. To do so, we implement a modified version of *round-robin sampling* as described by Stickland and Murray (2019). We set the number of training steps to be N, the number of batches in the largest dataset, then duplicate and concatenate all other datasets to themselves until their lengths are greater than or equal to N. At each step, we take a batch from each of the modified datasets and compute that task's respective loss. The weighting strategy for these losses is the subject of our research, and our three methods are described below.

Additive multitask fine-tuning This is the most basic method and that implemented by Bi et al. (2022). Generally, for n tasks we compute a final loss:

$$\mathcal{L}_{\text{Final}} = \sum_{i=1}^{n} \mathcal{L}_i \tag{2}$$

and optimize according to this loss.

$$\mathbf{G_t} = \nabla_\theta \mathcal{L}_{\text{Final}}(\theta) \tag{3}$$

Weighted multitask fine-tuning First, let $\boldsymbol{\ell_t} = [\ell_{t,1}, \ell_{t,2}, \ldots, \ell_{t,n}] \in \mathbb{R}^n$ be the vector of *loss improvements* across tasks at timestep t. The ith element of ℓ_t is given by

$$\ell_{t,i} = -\left(\mathcal{L}_{t,i}(\theta) - \mathcal{L}_{t-1,i}(\theta)\right) \tag{4}$$

Note that $\mathcal{L}_{t,i}(\theta)$ denotes the loss for task i at the current update step and $\mathcal{L}_{t-1,i}(\theta)$ denotes the loss for task i at the previous update step. We define a weight vector $\mathbf{w_t} \in \mathbb{R}^n$ which takes the softmax over $\boldsymbol{\ell_t}$, as such:

$$\mathbf{w_t} = \text{softmax}(\boldsymbol{\ell_t}) \tag{5}$$

We propose two possible usages of this weight vector. The first involves direct weighting of each task's gradient, which we call **weighted multitask fine-tuning**. Specifically, for n tasks we compute the gradient of the loss $\mathbf{G_t}$ as:

$$\mathbf{G_t} = \sum_{i=1}^{n} \mathbf{w_{t,i}} * \nabla_\theta \mathcal{L}_{t,i}(\theta) \tag{6}$$

Stochastic multitask fine-tuning The second is **stochastic multitask fine-tuning**. This involves sampling from $\mathbf{w_t}$ at each step and choosing the gradient of that task's loss function as the direction to step in, ignoring all other tasks. We sample task i from the distribution given by $\mathbf{w_t}$, and compute the gradient simply as

$$\mathbf{G_t} = \nabla_\theta \mathcal{L}_{t,i}(\theta) \tag{7}$$

[2]As a baseline, we test these multitask fine-tuning methods against *sequential fine-tuning*, in which we train on each dataset one at a time.

4 Experiments

4.1 Data

4.1.1 Pretraining

The Yelp[3]dataset consists of 700k Yelp reviews labeled with their star ratings (1 to 5). We pare the size of the dataset down to 27,500 reviews, and use this as an in-domain dataset for sentiment analysis.

The SICK[4]dataset consists of 10k sentence pairs labeled with relatedness (semi-continuously between 0-5) and entailment relations between the two sentences (possible labels *contradiction*, *entailment*, and *neutral*). We use this as an in-domain dataset for semantic textual similarity.

The PAWS[5]dataset consists of 108,463 human-labeled and 656k noisily labeled sentence pairs labeled with 0 (not paraphrases) or 1 (paraphrases). We use the human-labeled portion as an in-domain dataset for paraphrase detection.

Because we are doing masked language modeling in this task, we only use the text from these datasets.

4.1.2 Fine-tuning

The SNLI[6]dataset consists of 570k sentence pairs annotated with textual entailment information, with the labels *contradiction*, *entailment*, and *neutral*. We use this as an in-domain dataset for semantic textual similarity and paraphrase detection.

The MultiNLI[7]dataset consists of 430k sentence pairs with the same structure as SNLI but covers a wider range of genres of spoken and written text.

The SST[8], dataset consists of 215,154 phrases extracted from movie reviews with their semantic polarity labeled from 0-4, with 0 indicating negative and 4 indicating positive. We pare down to 8,544 examples for training.

The Quora[9]dataset consists of 400k sentence pairs with binary labels indicating if they are paraphrases of each other. We pare down to 141,506 examples for training.

The SemEval STS Benchmark[10]dataset consists of 8,628 sentence pairs with their similarity labeled semi-continuously between 0-5, with 0 indicating complete dissimilarity and 5 indicating identical statements. We pare down to 6,041 examples for training.

4.2 Evaluation method

We evaluate the sentiment classification and paraphrase detection tasks on accuracy, given the labels are categorical. We evaluate the semantic textual similarity task using Pearson correlation. To determine the best model, we average the three scores from each task to produce an **overall performance** score. This score is a crude measure of the model's robustness across all tasks, though it does not account for the distribution of performance across the tasks.

4.3 Experimental details

We use the provided pretrained BERT embeddings and weights as a starting point. First, we determine the ideal additional pretraining setup by pretraining with all four datasets (4.1.1) individually. We also test each of the fine-tuning paradigms individually without pretraining. Next, we test combinations (ensembles) of fine-tuning paradigms and the ideal pretraining setup. Finally, drawing from what we've learned through our comprehensive testings, we experimented with a few more additional training setups in order to achieve the best overall performance score, which are discussed more in detail in section 5. We empirically find the following hyperparameters suitable for model training.

- Learning rate: 10^{-5}
- Batch size: 64 for contrastive learning, 8 for all other training

More details can be found in Table 8 in the appendix.

4.4 Results

Tables 1-6 show our testing results across model configurations; tables 1-5 contain dev set results and table 6 contains test set results. Table 1 contains baselines by task, computed by fine-tuning individually on the SST, Quora, and STS datasets. Table 2 contains additional pretraining results. Table 3 contains baselines for SNLI/MNLI fine-tuning. Table 4 contains results from multitask fine-tuning only. Table 5 contains results from full models trained on a combination of our main approaches, including pretraining, SNLI/MNLI fine-tuning, and multitask fine-tuning. Table 6 contains results of the additional training setups ran with the explicit aim of getting the best performance score. These experiments are ran with weights after pretraining and fine-tuning on SNLI and MNLI with contrastive learning. Table 7 contains our final test set results.

For each of the three tasks, we are able to exceed baseline performance (training on just that task) in at least one ensemble model. However, no single training scheme stands out as significantly better in all tasks.

We can observe that fine-tuning on SNLI and MNLI is able to improve the accuracies of the semantic similarity task compared to just training on STS. Using contrastive learning is also able to give a significant performance improvement on the similarity task.

Neither of stochastic or weighted multitask fine-tuning seems to improve significantly on additive multitask fine-tuning. Weighted multitask produces a significantly better score than the other two when training alone, but performs only middling in the ensemble tests. Stochastic multitask has the best performance when pretraining is applied, but the worst in all other tests. In fact, none of the multitask strategies improve significantly on sequential training, in which we fine-tune on one task at a time in succession: this result suggests that, at least in our model configuration, multitask fine-tuning provides minimal improvement over other fine-tuning strategies.

Finally, we were able to observe better results in our additional experiments by making some changes to our fine-tuning paradigm. We discuss these changes in more detail in the next section.

5 Analysis

We may trace the generally unimpressive performance of multitask fine-tuning to several issues. The first is because the three downstream tasks have separate objectives and conflicting gradients, thus gradient updates often improves the performance on one task but worsens another. Gradient Surgery (Yu et al., 2020) may be able to mitigate this to some extent. The second issue may be variance in training data size. We notice that the Quora training set is much larger than the STS and SST training sets. Thus in our training we loop over the smaller STS and SST datasets many times, and continue the training until the Quora dataset is exhausted. We find that the performance of each task is somewhat correlated with the size of training set, so it may be possible that multitask fine-tuning is systematically favor progress on the largest dataset, which has continuously new data, at the expense of the other tasks. Our stochastic and weighted multitask fine-tuning results may also suffer due to issues with *loss improvement*. The loss functions for the different tasks may be scaled differently. Since we compute the loss improvement as the absolute difference in loss between time steps, it is possible that the weights are affected by the different scaling and do not accurately reflect the improvement of each task. In future work, it would be helpful to standardize the losses for each task based on the specific loss functions used.

It is important to note that although multitask fine-tuning does not offer a significant improvement in overall performance over sequential fine-tuning, it does offer more balanced performance across different downstream tasks. We can observe that sequential training gives higher performance on similarity but worse performance on sentiment, likely because the last dataset it trained on was the similarity one. In contrast, multitask fine-tuning offers more even accuracies across the board. Optimizing multiple objectives necessitates tradeoffs on individual performance, and achieving this optimal balance between objectives is where we believe multitask fine-tuning is most valuable.

We also found that fine-tuning with SNLI and MNLI greatly improved performance in paraphrase detection and semantic textual similarity over the baseline. This result is almost certainly due

Table 1: Dev set accuracies of baseline models

Model	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
With no pretraining or fine-tuning	0.144	0.380	0.019	0.181
Fine-tuning on SST dataset	**0.508**	0.421	0.223	0.384
Fine-tuning on Quora dataset	0.163	**0.836**	0.409	0.469
Fine-tuning on STS dataset	0.180	0.400	**0.450**	0.343

Table 2: Dev set accuracies of pretrained models

Pretraining Dataset	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
IMDb	0.186	0.608	0.086	0.293
Yelp	0.152	0.597	0.126	0.292
SICK	0.156	0.614	0.034	0.268
PAWS	0.150	0.624	0.121	**0.298**

Table 3: Dev set accuracies after fine-tuning on SNLI and MNLI

Model	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
Linear fine-tuning on SNLI/MNLI	0.283	0.657	0.668	0.536
Contrastive fine-tuning on SNLI/MNLI	0.168	0.476	**0.816**	0.487

Table 4: Dev set accuracies after multitask fine-tuning

Fine-tuning scheme	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
Sequential: SST, Quora, then STS	0.375	0.812	0.571	0.586
Additive multitask	0.483	0.802	0.493	0.593
Stochastic multitask	0.502	0.803	0.428	0.578
Weighted multitask	0.501	0.854	0.509	**0.621**

Table 5: Dev set accuracies from combination of methods

Model	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
SNLI/MNLI linear + Additive multitask	0.503	**0.860**	0.572	0.645
SNLI/MNLI linear + Stochastic multitask	0.493	0.810	0.594	0.632
SNLI/MNLI linear + Weighted multitask	0.496	0.830	0.599	0.642
SNLI/MNLI contrastive + Additive multitask	0.502	0.851	0.595	0.649
SNLI/MNLI contrastive + Stochastic multitask	0.498	0.807	0.597	0.634
SNLI/MNLI contrastive + Weighted multitask	0.492	0.837	0.581	0.637
Pretraining + SNLI/MNLI contrastive + Additive multitask	0.489	0.798	0.443	0.577
Pretraining + SNLI/MNLI contrastive + Stochastic multitask	0.500	0.796	0.624	0.640
Pretraining + SNLI/MNLI contrastive + Weighted multitask	**0.517**	0.826	0.530	0.624

Table 6: Dev set accuracies from additional experiments (after SNLI/MNLI contrastive)

Model	Sent. Dev Acc	Para. Dev Acc	Sim. Dev Corr	Overall
Frozen Quora/STS	0.453	0.746	0.816	0.672
Replace STS with SNLI in multitask fine-tuning	0.506	0.759	0.798	0.688
Multitask fine-tuning Quora/SNLI + frozen STS	0.460	0.830	0.789	**0.693**

Table 7: Test set accuracies

Model	Sent. Test Acc	Para. Test Acc	Sim. Test Corr	Overall
Multitask fine-tuning Quora/SNLI + frozen STS	0.443	0.829	0.776	0.683

to simple quantity of training data; the more there is, the better the model becomes. The use of contrastive learning is able to improve the performance in the similarity task even further, likely because the contrastive learning framework is able to generate a better representational space for sentence embeddings based on the sentence's semantic meaning.

Additional pretraining seems to have little to no benefit on the overall performance of the model. We posit this result is related to the problem of *catastrophic forgetting*, in which training beyond the initial pretraining destroys some of the model's fundamental knowledge, reducing performance. As we train the model for our downstream tasks, the weights learned during pretraining may have been overwritten. Indeed, many of the models in Table 5 that combines multiple methods suffer from the same problem: additional pretraining negatively affects the initial weights, SNLI/MNLI fine-tuning negatively affects additional pretraining, and multitask fine-tuning negatively affects SNLI/MNLI fine-tuning. We observe this behavior during training by evaluating the performance metrics at intervals; for all tasks, performance tends not to proceed continuously upward, but to fluctuate between training segments. One proposed solution comes from Sun et al. (2019), who implement a layer-wise decreasing learning rate to combat catastrophic forgetting.

Drawing on these results, we devised several additional experiments with the aim of getting even better performance. These experiments were motivated by the observation that contrastive learning is able to achieve a remarkably high score on semantic similarity and performs moderately well on paraphrase detection, but these gains are lost during the multitask fine-tuning step when we train on all datasets at once. We wanted to find ways to retain the performance gain from contrastive learning and combat catastrophic forgetting, thus we experimented with the following changes. First, after training on SNLI and MNLI, we freeze the weights of base BERT when fine-tuning the Quora and STS datasets, so that only the final linear layers of the respective tasks can be modified during training. This makes it so that the previously learned sentence embeddings stay the same, thus the performance on similarity is retained even while training on other tasks. Next, we tried replacing STS with SNLI during the multitask training step, and using contrastive loss instead of mean squared error. We did this because after training on SNLI and MNLI, we observed that training on STS seems redundant and does not offer performance gains. Instead, we would like to fine-tune the model with the superior contrastive learning framework rather than the standard MSE loss during the multitask fine-tuning process. Finally, we also tried to do multitask fine-tuning on just the Quora and SNLI datasets (ignoring SST sentiment dataset), then freezing the weights and training on SST after. This combination of multitask fine-tuning and freezing weights gave us our overall best result, so this is the final model we submitted to the test set leaderboard. We believe the reason that this model turned out to be the most effective may be because the objectives of the paraphrase detection and semantic similarity tasks are closely related, and thus these two tasks benefit the most from having gradients updated at the same time, while the sentiment task is trained later with frozen weights to prevent its gradient from conflicting with the paraphrase and similarity tasks.

6 Conclusion

We implement additional pretraining, fine-tuning on additional data, contrastive learning, and multitask fine-tuning (including two methods original to our research) to optimize the minBERT model for multiple downstream tasks. We see that all our methods offer significant improvements over the baseline model. We found that contrastive learning is especially effective for the semantic textual similarity task, and multitask fine-tuning is valuable for getting balanced accuracies across separate tasks. However, we notice that our models suffer from the catastrophic forgetting problem, and there is no single model that performs the best across all tasks. Our best model, which combines our main strategies along with slight changes to solve the catastrophic forgetting problem, achieves an average accuracy of 0.693/0.683 on the dev/test sets, respectively.

Using a single BERT model to accomplish multiple downstream tasks is a relatively unexplored challenge in NLP and there is much more to be done. For future work, we could explore more multitask fine-tuning strategies, such as standardizing the losses in our weighted multitask learning method or using gradient surgery. We could also explore ways to mitigate catastrophic forgetting, such as using decreasing layer-wise learning rate or freezing weights of certain layers. The fact that we are able to substantially improve performance of our main approaches by freezing weights when training on separate tasks suggests that there is strong potential for even better performance if we could handle this problem well.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Mtrec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2663–2669. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Travis R. Goodwin, Max E. Savery, and Dina Demner-Fushman. 2020. Towards zero shot conditional summarization with adaptive multi-task fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3215–3226. Association for Computational Linguistics.

John M. Jumper, Richard O. Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russell Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, R. D. Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Johannes Söding, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David L. Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with alphafold. *Nature*.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s).

Hyun Kim, Joon-Ho Lim, Hyun-Ki Kim, and Seung-Hoon Na. 2019. QE BERT: Bilingual BERT using multi-task learning for neural quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 85–89, Florence, Italy. Association for Computational Linguistics.

Salima Lamsiyah, Abdelkader El Mahdaouy, Saïd El Alaoui Ouatik, and Bernard Espinasse. 2023. Unsupervised extractive multi-document summarization method based on transfer learning from BERT multi-task fine-tuning. *J. Inf. Sci.*, 49(1):164–182.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on BERT for biomedical text mining. *CoRR*, abs/2005.02799.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Xuyang Wu, Alessandro Magnani, Suthee Chaidaroon, Ajit Puthenputhussery, Ciya Liao, and Yi Fang. 2022. A multi-task learning framework for product ranking with bert. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 493–501, New York, NY, USA. Association for Computing Machinery.

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

[3] https://huggingface.co/datasets/yelp_review_full

[4] https://huggingface.co/datasets/sick

[5] https://huggingface.co/datasets/paws

[6] Bowman et al. (2015) https://nlp.stanford.edu/projects/snli/

[7] Williams et al. (2018) https://cims.nyu.edu/~sbowman/multinli/

[8] Data from Stanford Sentiment Treebankparsed with Stanford Parser

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

[10] https://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark

7 Appendix

Table 8: Experiment details

Training Task	Epochs	Time per epoch
IMDb pretraining	2	17.5min
Yelp pretraining	3	9.5min
SICK pretraining	10	0.75min
PAWS pretraining	1	10min
SNLI fine-tuning	1	60min
MNLI fine-tuning	1	60min
Additive multitask	2	42min
Stochastic multitask	1	30min
Weighted multitask	1	44min