BERT Multitask Methods for Low-Parameter Fine-Tuning

Stanford CS224N Default Project

Elton Manchester
Department of Computer Science
Stanford University
eltonm@stanford.edu

Abstract

This project aims to implement and compare low-parameter strategies for multitask learning in a BERT language model. These include projected attention layers (PALs), task-embedded attention (EmBERT), and a basic multi-head classification system. These methods allow for good scalability, allowing a model to perform well on a large range of tasks without the additional overhead of a large number of added parameters. An additional advantage of effective low-parameter techniques is that the majority of information encoded by the BERT embedding must be stored in the base model, leading to more robust embeddings. However, basic multitask learning is vulnerable to conflicting gradients between tasks, which leads to interference between tasks and worse performance overall. This motivates the addition of task-specific functions other than the classification heads, as these functions can encode information about each task without interfering with other tasks. The main differences between different techniques are the specific function used, and the integration location of this function. I investigated varying combinations of function and location to determine which methods are most effective. This included the simultaneous use of multiple functions and locations, which was not tested in previous experiments. The best performing technique involved adding task-specific PALs before classification, which differs from preexisting literature. I propose that this disparity is due to the low-resource setting, which emphasizes the importance of the pretrained weights of the model. Attempting to integrate task-specific parameters can interfere with the base model, and must be done cautiously to avoid overfitting. This suggests that distinct task-specific layers are more robust than parallel integration in this setting because of the reduced interference with the embeddings of the pretrained BERT model.

1 Project Information

- TA mentor: Default Project
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Multitask training is particularly important in language model applications. With the current prevalence of AI assistant models, language models are expected to respond well to almost any request a user makes. Due to the wide range of tasks, it is unfeasible to attempt to fine-tune a model for each potential application. This experiment focuses on a small set of three tasks, but the intention is for the methods used to be scaled to a much larger set of tasks.

Current methods have demonstrated state-of-the art performance on standard benchmarks, in some cases even beating individually trained single-task models. However, each of these results is based on the use of only one of the many techniques that have been explored. It may be valuable to combine several due to the different effects on the encoding that they can provide. Some approaches simply add additional parameters or linear transformations to each layer, while others add task-specific attention. It is reasonable to hypothesize that each of these transformations conveys a unique aspect of each task, in which case models would benefit from multiple methods. The goal of this work is to evaluate this possibility in comparison to individually applied multi-task techniques. Adding too many task-specific functions has diminishing returns and may also dilute the base embeddings, so a secondary goal is developing methods to ensure that added parameters augment the embeddings rather than dominate them. For this goal, propose the simple addition of a trained scalar value, initialized to a small float, which multiplies the contribution of each task-specific function.

3 Related Work

The two main papers this work is based on are BERT and PALS [1] and Task-Embedded Attention (EmBERT) [2]. PALs is an early paper with a similar structure and goal to this one. The authors were attempting to achieve state-of-the art performance while minimizing the parameters used. Of the task-specific functions, they found that projected attention layers performed the best, hence the name of the paper. With their implementation, the model was able to perform well on 8 different tasks with only 1.13x the parameters of the original model. On top of this, they introduced a sampling scheme called annealed sampling which is geared specifically towards addressing common pitfalls in multi-task training. Every model in this experiment makes use of this sampling method, which has an adjusted distribution based on the current epoch. They also experimented with different insertion locations, with canddiates including the top of the model and at the end of each BERT layer. Both options are explored here.

EmBERT is a later paper that expands on PALs by introducing an even more parameter-efficient technique. The namesake of EmBERT is task-embedded attention, which involves inserting a task-specific vector into each head of multi-head attention. This experimentally achieved comparable results to PALs, but is more efficient int terms of parameters. The improvements over the shared-parameter model were small, but significant in the context of the size of the model.

Finally, there is a paper that I did not reference for this work, but presents an interesting application of multi-task training. This is the Liu paper on multi-tasked deep neural networks for semantic classification [3]. In this paper, researchers were able to use multi-task training to address the issue of data limitations in supervised training. Supervised training can be especially effective due to the human approval, but can be laborious to collect data for. Researchers were able to take human-annotated data for a range of similar tasks with the intent of applying it to a smaller range of tasks. They were able to demonstrate that the combination of tasks actually improved the performance of each task.

4 Approach

The starting point of the experiment is the most basic form of a multitask learning model. This is also my baseline, similar to its use in the EmBERT paper [2]. This model shares all parameters between tasks, with classification heads as the only difference between the tasks. Any successful technique should have a better performance than this model, otherwise the additional parameters would be a waste of storage and computation resources. The EmBERT authors also compared to separately trained models for each task. Similarly, I will use the model trained for sentiment classification only as a point of comparison.

The next step is to add projected attention layers as described in the paper BERT and PALs [1]. This method involves inserting task-specific, miniature BERT layers between the last base BERT layer and the classification heads. To reduce the parameters required, the PAL projects the input down to a lower-dimensional space before calculating attention. The result is then projected back to the original dimension to be classified. This transformation is given by

$$\mathrm{TS}(\mathbf{h}) = V^D g(V^E \mathbf{h}), \tag{1}$$

where V^E and V^D are the encoder and decoder projection matrices. g can be any function (in this case, it is multi-head-attention). The projected size can be arbitrarily small, so I will use the size recommended by the authors (204). There is an alternate use of these PALS presented in which they are inserted into the final layer normalization of each BERT layer. This allows the final matrix multiply to be left out of each PAL, saving parameters. On top of this, I add one last extension in the form of a learned scalar to multiply the PAL output by before adding it to the FFN output of the layer. This moderates the strength of the PAL compared to the base model.

For a third point of comparison, I will use task-embedded attention as in EmBERT [2]. This approach uses even fewer parameters than PALs, adding a single task-specific vector the each of the multi-head attention heads. This altered head is given by

$$EmAtt(Q, K, V) = Softmax\left(\frac{\hat{Q}^T \hat{K}}{\sqrt{d_k}}\right)\hat{V} \tag{2}$$

$$\hat{Q} = Q + Q^{(i)}_{emb}, \quad \hat{K} = K + K^{(i)}_{emb}, \quad \hat{V} = V + V^{(i)}_{emb}.$$

The terms added to the query, key, and value are the additional embeddings for task i. By using vectors instead of matrices, EmBERT has a tiny parameter cost, allowing it to be an effective option when a large number of tasks are required.

All of these approaches use annealed sampling for training. Batches are randomly sampled from all of the datasets according to a distribution dependent on the size of each set. The motivation of annealed sampling is that early epochs should avoid overfitting to small sets, whereas later epochs should make sure that all tasks are sufficiently represented. To this end, the probability of a sample being drawn from dataset D_i with size N_i is proportional to N_i^α, where α is given by

$$\alpha = 1 - 0.8\frac{e-1}{E-1}, \tag{3}$$

where e is the current epoch and E is the total number of epochs. As α gets smaller, the distribution approaches uniformity, giving the desired sampling behavior. Experimentally, the success of this method was seen when the accuracies of tasks associated with smaller datasets improved significantly in later epochs. For the scope and goals of this experiment it was not necessary to test multiple sampling schemes.

All code other than the provided BERT starter code is original. I will use hyperparameters and equations from the papers referenced, but the code itself will be my own. There are also a few unique aspects of my code that differ from the setups described in the source papers. First, the PALs paper shares the projection linear layers across all BERT layers, while my version uses separately trained projection layers for each PAL. Additionally, the trained scalar multiplier was an original contribution which was effective for this setup.

5 Experiments

Data: Every model except for the single-task model uses SST, CFIMDB, Quora, and SemEval datasets, while the single task uses only SST and CFIMDB. Future experiments could benefit from additional data, and incorporating more sets would likely result in a performance improvement. However, due to the comparative nature of the experiment, state-of-the art performance is not necessary as long as techniques can be compared against the baseline. Also, because this experiment uses fixed-length epochs, the incorporation of additional data would lead to compromises in training time or in representation of all sets.

Evaluation: The sentiment classification and paraphrase detection tasks use accuracy, which is the percentage of correct classifications. The STS task uses Pearson correlation, which takes a value from -1 to 1. All of these metrics will be compared to the base model. To produce a combined final score, the Pearson correlation is normalized, then the three scores are averaged.

Experimental Details: All variations of the model will use the same hyperparameters and training scheme to keep the environment as controlled as possible. Every test uses full-model fine-tuning with a learning rate of 1e-5, and 10 epochs with 400 batches per epoch. Annealed sampling is used across all tests due to its observed effectiviness across several studies. The time to run each test was consistenly around 1 hour due to the similar parameter counts for each trial.

The multitask configurations to be compared are as follows. **Single-Task** is a model fine-tuned only for sentiment classification. **Multi-Task** is a model that shares all parameters between tasks, so the classification heads are the only task-specific components. **EmBERT** uses task-embedded attention, meaning that task-specific vectors are added to each MHA head in every BERT layer. **PALs Top** inserts projected attention layers for each task before the classification head. **PALs Parallel** adds the output of a PAL before the final layer normalization of each BERT layer. **PALs Parallel Weighted** uses a single learned weight scalar for each layer that determines the contribution of the PAL. **Top + Parallel (Same)** uses a single PAL for each task, but inserts it both in parallel and on top. **Top + Parallel(Different)** uses two PALs for each task, one parallel and one on top. Finally, **Combined** uses task-embedded attention and one PAL for each layer added both in parallel and on top.

Results: For baselines, I used a single-task model fine-tuned only for sentiment classification, and a multitask model that shares all parameters between tasks. As expected, the performance of the multitask model suffered slightly on sentiment classification compared to the single-task model. Next, I tested several variations of task-specific functions and the locations in which they are inserted. Of these, only PALs top outperformed the baseline model. Finally, I combined groups of the more successful multitasking methods. The Top + Parallel (Same) model achieved the best results out of all of the tests. I also

	Single Task	Multitask
Sentiment (Accuracy)	0.522	0.501
Paraphrase (Accuracy)	N/A	0.709
STS (Pearson Correlation)	N/A	0.097

Table 1: Baselines (All Parameters Shared)

	Multitask	EmBERT	PALs Top	PALs Parallel	PALs Parallel Weighted
Sentiment	0.501	0.467	0.483	0.482	**0.503**
Paraphrase	**0.709**	0.550	0.674	0.606	0.550
STS	0.097	0.125	**0.302**	0.069	0.147
Overall	0.586	0.527	0.602	0.541	0.567

Table 2: Comparing Individual Multitask Methods

	Top + Parallel (Same)	Top + Parallel (Different)	Combined
Sentiment	0.496	0.464	0.449
Paraphrase	0.680	0.626	0.686
STS	0.293	0.263	0.249
Overall	**0.608**	0.573	

Table 3: Comparing Combined Multitask Methods

evaluated the best-performing method, PALs Top + Parallel (Same) on the test set. It received scores of 0.505, 0.676, and 0.315 for each task respectively, for an overall score of 0.615, even better than the score on the dev set. Unexpectedly, many of the methods tested performed worse than the baselines. A few configurations were much better than the baseline, but the majority were worse. In particular, a the methods that embed task-specific parameters into every BERT layer generally perform worse than the methods that use stand-alone layers, which I will discuss next in my analysis.

Analysis: Before implementing the learned weight scalars for each layer, I tested a few weight values manually for a single epoch to compare their effectiveness. Lower weights generally seemed more effective, which provides insight to the mechanism causing the reduced performance of certain configurations. It is likely that the task-specific parameters were over-represented in the final output, causing the model to effectively make use of a much smaller parameter count. On top of the learned scalar, it may also be useful to penalize large gradient updates to the task-specific functions to prevent them from dominating the encoding.

Additionally, differing gradient directions cause the improvement of one task to reduce the performance of another. Notably, the baseline model had the best sentiment classification score, but the second worse semantic textual similarity score. In contrast, the majority of the techniques used demonstrated more consistent accuracies across the board, even if the overall score was worse. This suggests that these techniques can address the the issue of "forgetting" tasks, but are not as effective for gradient alignment. This is because the task-specific parameters cannot be altered by other tasks, so each task will have some guaranteed representation in the model. This effect is most dramatic in the STS task, which has a relatively small dataset. The best model tripled the correlation of the baseline, showing that the baseline model may not have retained enough information about this task. However, qualitatively, epochs where one task was especially accurate tended to have worse scores in the other tasks, which is a symptom of gradient alignment issues. As such, it may be worth supplementing multi-task training with alignment techniques such as gradient surgery. Overall, with several configurations outperforming the baseline, this experiment successfully demonstrates the effectiveness of various multi-task techniques and identifies areas for improvement.

6 Conclusion

Based on the experimental results, the most robust multi-task methods involve stand-alone layers that are inserted between existing layers (in this case, before the classification heads). However, the integration of task-specific functions throughout the model can provide even better performance, provided that they are added in moderation. This serves as a successful proof-of-concept for the combination of multiple task-specific integrations, as the best-performing model was used PALs both as stand-alone layers and as supplemental additions within each BERT layer. I was unable to find literature in which a similar technique was used, so this is likely a novel idea.

However, there are some limitations in the result. First, the sub-par performance of some methods suggests that this implementation of each technique is un-optimized, which limits the significance of the results. As an exploratory study, this makes the results presented a starting point for potential further research, but not a conclusive endorsement of one technique over others. Additionally, the baseline itself was relatively low compared to other studies. This is not necessarily an issue because the purpose of the experiment is comparison, not state-of-the-art performance. However, the techniques presented have not yet been proven to be state-of-the-art viable. Finally, the amount of data and training time used is limited due to resource availability.

An avenue for further research would be a more thoroughly optimized and fine-tuned version of new combined techniques presented. This could reinforce the legitimacy of these methods and compare them to a more universal baseline. In particular, addressing the gradient alignment issue further would potentially allow for large performance improvements.

7 Ethical Considerations

When engaging with multi-task learning, one significant ethical concern is seen in the biases and unforseen side-effects that can be produced by the use of disparate datasets. Even in single-task applications, the model is effectively extrapolating from the training information, which is already logically unsound. In multi-task training, the model makes decisions on each task partially based on data from the other tasks, which has the potential to increase the risk of biases or unintended behaviors. This factor makes evaluating and testing each model for biases and potentially harmful results vitally important. For this experiment, the use of annealed sampling addresses the issue by ensuring that all tasks are fairly represented. This reduces the risk that one task skews the results of another task, therefore also reducing the risk of side-effects. If similar techniques were used for public or commercial use, however, it would be necessary to engage in further testing (such as with the AI Safety Benchmark), as simple accuracy numbers are insufficient to determine biases.

Additionally, there would be ethical concerns if similar models were used for slightly different applications than their original training. Compared to a shared-parameter multi-task model, the injection of task-specific data serves to make the model more specialized. For example, a model trained to recognizes paraphrases in Quora questions may not be well-suited to identify paraphrases in a math or programming forum. This could pose problems when the model is used for automatic moderation, as comparable performance to the original task cannot be guaranteed. As before, specific testing and fine-tuning of the model for the desired context would be the solution. However, to protect against potential misuse of the model, it may be worth being conservative with the addition of task-specific information. In this experiment, this was accomplished by the addition of the scalar multipliers. These could further be normalized to a smaller range of values in order to ensure the added information provides a sufficiently small impact on the final embedding.

References

[1] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning, 2019.

[2] Łukasz Maziarka and Tomasz Danel. Multitask learning using bert with task-embedded attention. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2021.

[3] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. NAACL, May 2015.