Learning with PALs: Enhancing BERT for Multi-Task Learning

Stanford CS224N Default Project

Michael Hoang
Department of Computer Science
Stanford University
hoangm@stanford.edu

Abstract

Exploration in a single large base model to work on multiple tasks, natural language understanding or otherwise, is an effort known as multi-task learning. Creating a large model with many shared weights to perform many tasks prevents the need to have separately fine-tuned models for each task, reducing total memory requirements.

A method to enable multi-task learning is by adding projected attention layers (PALs) to enable multi-task learning for a modified BERT model, allowing for weight sharing in one model rather than updating all pre-trained weights during fine-tuning for each individual downstream task. In this project, we modify a BERT model to add PALs throughout the model, discussing trade odds of of model size against performance.

This work also discusses various sampling techniques that yield better results, and conclude that adding PALs to the upper half of a BERT model with non-round-robin sampling methods achieve the best results.

1 Key Information

- **Mentor**: Annabelle Tingke Wang

2 Introduction

With the recent attention of LLMs in academia and in the media, many large corporations and startups alike have trained their own models with increasing parameter count. Models with modest parameter counts less than a decade ago now have billions of parameters, with trillion parameter models now being trained (Dash et al., 2023). As scaling laws have continued to prove that larger parameter count improves performance, the expenses required to train and fine-tune such models have become astronomical. There is a growing body of work that attempts to train models that achieve the same amount of performance as their counterparts (Hoffmann et al., 2022) at a fraction of the parameter count, leading to an entire field of machine learning model creation known as efficiency (Wan et al., 2023), with many sub-fields that focus on a variety of tasks such as quantization, parameter-efficient fine-tuning, and sparsity, to name a few.

Reducing parameter count and saving memory is an incredibly important task, especially in memory constrained environments such as mobile devices or IoT-enabled micro-controller applications, in addition to training large models on servers. By reducing model size without sacrificing significant accuracy, smaller and cheaper memory can be purchased, making hardware solutions more affordable and thus more attractive. More memory can be allocated to learned parameters in other layers or parts of the model, allowing for exploration of other architectures. Computation and cost is also saved

by reducing parameter count. According to Belevich et al. (2022), training a 300B model with 512 NVIDIA A100s costs more than $4 million dollars.

In addition to reducing parameter count using the methods mentioned above, having the same model perform multiple tasks, removing the need to fine-tune a model for each individual task, is another active method for reducing total parameter count and memory requirements. This is called multi-task learning. In this work we implement Projected Attention Layers (PALs), first introduced in Stickland and Murray (2019) as an architectural means to learn multiple objectives in a model. Alternatives to round-robin sampling are also introduced that improve performance by enabling dataset-count aware sampling methodologies called square root sampling and annealed sampling.

3 Related Work

3.1 Multi-task Learning

In addition to Projected Attention Layers in multi-task learning, other related works explore multi-tasking in models using various methodologies. While PAL explores adding projected attention layers to various points in the model, methods such as gradient surgery (Yu et al., 2020) exist to improve accuracy during loss calculation. Instead of an architectural change, gradient surgery projects a task's gradient onto the normal plane of another task's gradient that is "in conflict" or causing interference. This operation in the training algorithm ensures that model updates during training for one task don't hinder performance of another.

In addition to LLMs, there have been many publications exploring methods of enabling models to multi-task in various modalities. Wang et al. (2021) explores creating a unified convolution-based vision model using a single representation across multiple vision tasks. It explores "explicit" and "implicit" knowledge representation in brain anatomy to reconcile learning on multiple tasks, such as object recognition and segmentation. Tang et al. (2020) employs multi-task learning in a recommendation system using a mechanism called progressive layered extraction (PLE). Sharing all weights and parameters for all tasks, PLE extracts and passes relevant information between specific shared and task-specific layers in an attempt to be a hybrid between an architecture like PALs and fully fine-tuned models.

3.2 Baseline

This work's implementation of PALs builds on top of established work involving the BERT model (Devlin et al., 2018). The BERT model uses 12 layers of encoders consisting of Transformers introduced in Vaswani et al. (2017).

4 Approach

As the baseline section introduced, the BERT model was used as the inspiration for a minimalist BERT model implementation that was modified to include the additional Projected Attention Layers. A skeleton of the minBERT model was provided and portions of the model were implemented in the work. The base BERT model has 12 hidden layers, 12 attention heads, and a hidden dimension size of 768.

4.1 Multi-headed Attention

The multi-head attention layer (Vaswani et al., 2017) is the component of the transformer architecture that projects the input sequence to each head as a projected hidden state. The attention mechanism computes a scaled-dot product by computing the dot product of a query with all keys, dividing by the square root of the dimension of the keys, and applying a softmax to obtain attention values, which are a weighted sum of the values.

$$\text{Attention}_i(h_j) = \sum_t \text{softmax}\left(\frac{W_i^q h_j \cdot W_i^k h_t}{\sqrt{d/n}}\right) W_i^v h_t \tag{1}$$

As seen in Equation 1, the self attention performed on each head h_j is dependent on the number of heads n. The number of heads determines the scaling factor $\sqrt{d/n}$ as well as the dimensions of W_i^q, W_i^k, and W_i^v as size $d/n * d$. The notation of h_t denotes every sequence element.

The multi-head portion of multi-head attention allows the model to simultaneously attend to different features in different dimensional-sub-spaces to obtain richer informational representation. Afterwards, each individual head is concatenated together before being applied to a dxd linear transformation with W^O, as seen in Equation 2 (we drop the j in h_j).

$$MultiHead(h) = Concat(head_1, ..., head_j)W^O \tag{2}$$

For completeness, in addition to the multi-headed attention layer, other components of a full BERT layers also consist of a feed-forward layer (FFN) and layer normalization (LN):

$$SelfAttention(h) = FFN(LN(h + MultiHead(h))) \tag{3}$$

With the complete BERT layer being:

$$BertLayer(h) = LN(h + SelfAttention(h))) \tag{4}$$

4.2 PAL Implementation

Our approach for implementing PALs will follow a similar approach as in Stickland and Murray (2019).

Instead of adding parameters "at the top" or after the final layers of the BERT base model, our work introduces and modifies parameters within the BERT layers themselves. In order to capture the different downstream tasks in a separate dimensional subspace, multi-headed attention layers are introduced within each BERT layer. In order to reduce parameter count, a linear layer is introduced that transforms the BERT hidden dimension to a lower PAL hidden dimension before computing multi-headed attention:

$$PAL(h) = W^{d_{hidden}} MultiHead(W^{d_{PAL}} h) \tag{5}$$

The original BERT hidden dimension of $d_{hidden} = 768$ is used, whereas a PAL hidden dimension $d_{PAL} = 204$ is used. Where the original Stickland and Murray (2019) paper uses a shared linear transformation for all of the PALs, our work deviates by adding a transformation layer in every PAL, adding some parameters for performance. Originally, all 12 BERT layers were fitted with PALs but we found that using fewer PAL-enhanced BERT layers such as 6 did not hurt performance, while also reducing model size.

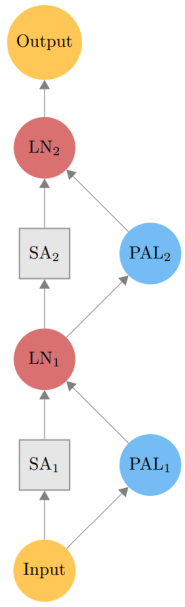

Figure 1: Projected Attention Layers between BERT layers

Similar to the reference work, inspiration is also taken from He et al. (2015) to introduce PALs "in parallel" to each of the BERT layers, much like residual connections, shown in Figure 2. The goal is to introduce task-specific parameters to each layer, l, of the BERT model in addition to improving gradient flow from training on the each task.

$$h^{l+1} = LN(h + SelfAttention(h) + PAL(h)) \tag{6}$$

To reduce parameter count from the addition of PALs, the final $d * d$ linear transformation with W^O described in Section 4.1 is also removed, without affecting performance.

4.3 Sampling Methods

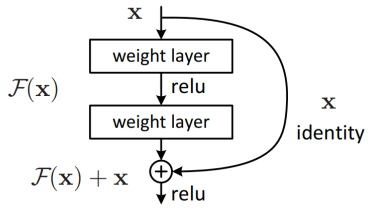

Figure 2: Residual Adapter from He et al. (2015).

An important consideration in multi-task learning is the introduction of different sampling methods for datasets during training. As BERT with PALs requires training data from all downstream tasks in order to improve the accuracy of each task, a method for fairly sampling data where disproportionate data exists between the downstream tasks must be addressed in order to avoid interference of a particular task being trained on for many steps to the detriment of other tasks. Different sampling techniques are described.

Round robin sampling is the simplest form of sampling from each dataset, cycling through each dataset in a deterministic fashion. However, due to training sample availability, this means that if a given task has a smaller dataset than another's, all training examples will have been cycled through, resulting in either an imbalanced training schedule or looping through the smaller dataset's training examples again. This disproportionate size imbalance of datasets could lead to either overfitting on tasks with smaller datasets or under-training of tasks with larger datasets.

Square root sampling is introduced to prevent disparity between the probabilities of choosing tasks. It does this by setting the probability of selecting data from a task as proportional to the fractional root of the number of training examples of the task:

$$p_i \propto N_i^{\alpha}$$

This solves the imbalance of dataset size if round robin were used. However, it also introduces under-training of tasks with smaller datasets as problematically, training data from that task's particular dataset will be extremely low if dataset size imbalances are large.

Annealed sampling is introduced to alleviate under-training of tasks with smaller datasets. It changes α above to train tasks more equally towards the end of training where interference has the greatest risk and ensures smaller tasks are adequately trained (E is total number of epochs):

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

5 Experiments

5.1 Data

We use three datasets to evaluate our BERT model. One dataset exists for each of the downstream tasks, each with differing training, evaluation, and test dataset sizes.

- **Sentiment analysis**: the Stanford Sentiment Treebank containing 215,154 unique phrases will be used, which contains a possible sentiment out of 5 sentiments. The training set contains 8,544 examples. The dev set contains 1,101. The test set contains 2,210 examples.
- **Paraphrase detection**: a dataset with labeled question pairs is provided by Quora is used to indicate whether different questions are paraphrases of each other. The training set contains 141,506 examples. The dev set contains 20,215 examples. The test set contains 40,431 examples.
- **Semantic textual similarity**: the SemEval STS Benchmark dataset is used consisting of different sentence pairs of varying similarity on a scale from 0 to 5, with the former having no semantic similarity and the latter having semantic. The training set contains 6,041 examples. The dev set contains 864 examples. The training set contains 1,726 examples.

5.2 Evaluation method

For each downstream task, a test set is used to determine the accuracy of of predictions for our PAL-enhanced model. However, given our objective is to produce a single model that's able to

complete inference on the test set with comparable accuracy to the fine-tuned models, we use the accuracy of the fine-tuned models as our baseline metric.

The baseline fine-tuned models were trained using naive task-specific classification at the output layer and does not optimize for higher accuracy using cosine similarity or other performant methods.

In addition to comparison to baseline performance, model size comparison to the summation of the fine-tuned models for each downstream task is included.

5.3 Experimental details

For all our experiments, 12 encoder layers were used. A mixture of 12 vanilla BERT layers, 12 PAL-enhanced BERT layers, and 6 PAL-enhanced BERT layers out of a 12 layer BERT model were all trained. The Adam optimizer (Kingma and Ba, 2014) was implemented and used during training, with a learning rate of $1E-5$ A dropout probability of 0.3 was used.

Fine-tuned models for our baseline were trained using parameters of 10 epochs a batch size of 8. Training the STS fine-tuned model took approximately 30 minutes. Training the paraphrase fine-tuned model took approximately 8 hours. Training the SST fine-tunred model took approximately 25 minutes.

All PAL-enhanced models trained in batch sizes of 16.Total training time using our annealed method for all downstream tasks took approximately 30 minutes. Given the dataset size differences between the Quora dataset and the SST and SemEval STS, we trained on 6000 epochs using square root sampling, with the last 1000 epochs using annealed sampling.

Open source implementation of PALs and square root and annealed sampling methods were not used or inspected during implementation.

5.4 Results

Given different configurations of PAL-enhanced BERT, Table 1 compares different prediction performance against each of the fine-tuned models trained purely on each downstream task. We found that our most performant model was 6 PAL, 6 BERT and No Pooling Layer Model and normalize it to the fine-tuned models.

Table 1: Performance Comparison on Downstream Tasks

Model Configuration	**STS Acc.**	**Paraphrase Acc.**	**SST Pearson Corr.**
Fine-tuned BERT Models	0.527	0.787	0.361
12 PAL Model	0.496	0.760	0.285
6 PAL, 6 BERT Model	0.503	0.762	0.246
6 PAL, 6 BERT, No Pooling Layer	0.520	0.758	0.321
6 PAL, 6 BERT, No Pooling Layer + Shared	0.400	0.759	0.329
================================			
6 PAL, 6 BERT, No Pooling Layer (Normalized)	0.987	0.963	0.892

In addition to the different model configuration sizes given in Table 2, the fine-tuned BERT model size of 417.MB can also be multiplied by 3 for a total memory requirement of 1253.01MB. If we create a ratio of 1253.01MB to the size of our most performant model of 6 PAL, 6 BERT, with No Pooling Layer, then the ratio is 0.34, an expected 1/3 of the memory requirement from summation of each of the fine-tuned models.

Finally, a comparison of our most performant PAL model of 6 PAL, 6 BERT and No Pooling Layer using square root sampling and annealed sampling is compared against the round robin sampling method. The round robin method used a batch of 2 to prevent re-using data for the STS and SST task datasets.

Our

Table 2: Model Size with Varying PAL Configurations

Model Configuration	Size (MB)	# Ratio compared to Base
Fine-tuned BERT Model (Base)	417.67	1.0
12 PAL Model	437.8	1.048
6 PAL, 6 BERT Model	427.8	1.024
6 PAL, 6 BERT and No Pooling Layer Model	425.48	1.019
6 PAL, 6 BERT and No Pooling Layer + Shared PAL Transform	419.49	1.004

Table 3: Model Performance under Different Sampling Configurations

Sampling Configuration	STS Acc.	Paraphrase Acc.	SST Pearson Corr.
Square Root and Annealed Sampling	0.520	0.758	0.321
Round Robin Sampling	0.503	0.707	0.327

6 Analysis

Our original model was the most performant model with a PAL transformation layer within each PAL instead of a shared transform between layers that was used in the original PAL paper.

This model may have outperformed the one with the shared transform layer in that it was able to capture the downstream tasks in each layer with its increased parameter count. One issue with PALs that is addressed in other publications is that of gradient interference, suggesting that the additional linear transforms within each PAL allows for less interference than with a shared transform.

In Table 3, round robin sampling performed best with the SST Pearson Correlation score, which had the fewest training samples, confirming problems with imbalanced datasets. However, this did not apply to STS, suggesting that the naive method of cycling through the other downstream tasks interfered with the gradients associated with STS accuracy. Our lower paraphase accuracy is expected given how large the dataset is for that particular task.

7 Conclusion

PALs were added to a base BERT model to enable multi-task learning and to examine performance and compare resultant model sizes. We found that our most performant model involved using a PAL-enhanced BERT model of 6 layers out of the 12, and also involved removal of the final linear transformation pooling layer after concatenating multi-head attention. Depending on the application and it's dependence on accurate predictions, the final normalized accuracy results may not be accurate enough for critical applications; however, for applications involving memory and battery constrained on-device ML, the reduced accuracy may be sufficient given the 1/3 lower memory requirement.

We also found that using different sampling methods that was dataset-size aware allowed for more performant models in each of the downstream tasks.

References

Pavel Belevich, Yanli Zhao, Shen Li, Jessica Choi, Rohan Varma, Pritam Damania, Geeta Chauhan, Mahesh Yadav, Pierre-Yves Aquilanti, and Sundar Ranganathan. 2022. Training a 1 trillion parameter model with pytorch fully sharded data parallel on aws. *Medium*. Accessed: March 17, 2024.

Sajal Dash, Isaac Lyngaas, Junqi Yin, Xiao Wang, Romain Egele, Guojing Cong, Feiyi Wang, and Prasanna Balaprakash. 2023. Optimizing distributed training on frontier for large language models.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

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

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. RecSys '20, page 269–278, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2023. Efficient large language models: A survey.

Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. 2021. You only learn one representation: Unified network for multiple tasks.

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