Choose Your PALs Wisely

Stanford CS224N Default Project

Zachary Rotzal
Department of Computer Science
Stanford University
`zprotzal@stanford.edu`

Abstract

In this project, a BERT model is built with projected attention layers (PALs) and trained with annealed sampling in order to perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks. The aim of the project was to create the simplest model that performs well on this multi-task problem. The results show that a model without PALs performs best. This result is due to a combination of task interference and the finding that PALs only boosts the performance of certain tasks.

1 Key Information to include

- Mentor: Heidi Zhang
- External Collaborators: None
- Sharing project: No

2 Introduction

The first part of this project implements the standard Bidirectional Encoder Representations from Transformers (BERT) model, whereas the second part, and main focus of this report, implements projected attention layers (PALs) in parallel with the BERT layers. This project is motivated by the goal of creating a simple model that is robust and generalizable to multiple sentence-level natural language tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. As outlined in Stickland and Murray (2019), PALs contains task specific parameters so that the base BERT model can have shared weights between all tasks. As such, PALs is a promising concept to achieve the goal of creating a simple model. An analysis of various BERT and PALs architectures in this project reveals that having more PALs is not always better.

3 Related Work

There are multiple research endeavors that explore the challenge of creating robust models that perform well on multiple tasks. This area of study is generally known as multi-task learning. For this project of creating robust BERT embeddings, the concept of Projected Attention Layers (PALs), as introduced in Stickland and Murray (2019), particularly stood out as a promising endeavor because of its superb performance on the GLUE benchmark, a benchmark composed of various natural language tasks (Wang et al., 2019). PALs also stood out because it used a reduced number of task specific parameters in order to increase the number of shared BERT parameters among tasks. Specifically, PALs adds task specific parameters in parallel to the layers in the original BERT model. Stickland and Murray found that the PALs and BERT implementation did not perform better on all individual GLUE task measures, but rather achieved state-of-the-art performance across the average of GLUE tasks. As such, PALs is argued to be a helpful tool to make BERT more generalizable and robust.

4 Approach

This project's approach to creating a strong multi-task BERT model is to implement projected attention layers in parallel with the BERT layers and use annealed sampling to reduce the effects of unequal training dataset sizes. Both of these concepts were introduced by Stickland and Murray (2019).

4.1 Model Overview

The base model used in this project builds off of the original BERT model and pretrained weights from Devlin et al. (2019). On top of the BERT model are separate, task specific layers, which are discussed in detail below. This structure is considered the "base" model or "zero/no PALs" model for this project. As an extension to this base model, projected attention layers are either added to the top six layers or to the entire twelve layers of the original BERT model. This paper will refer to these two models as the "half" and "full" PALs models, respectively. The PALs architecture is described below.

4.2 Top Layer

Each task specific layer takes in the BERT [CLS] token, applies dropout regularization, and processes it through a linear layer. In the sentiment analysis task, the linear layer produces five values, one for each of the five possible classes in this task. For the paraphrase detection task, the linear layer produces one output, corresponding to a binary "yes/no" to indicate if the two sentences are paraphrases of each other. In the semantic textual similarity task, the linear layer produces a single value, corresponding to the task's continuous output range between zero and five.

4.3 PALs in Parallel

Stickland and Murray (2019) found that implementing PALs in parallel to a BERT layer led to better downstream task performance than implementing PALs in serial. Following their guidance, this paper only implements PALs in parallel to BERT layers.

Each BERT layer can be defined as:

$$\mathbf{h}^{l+1} = LN(SA(\mathbf{h}^l) + FFN(SA(\mathbf{h}^l)))$$

where LN is a layer-norm, SA is the self-attention function, FFN is a feed forward network, $\mathbf{h}^l$ is the hidden state output of layer l, and $\mathbf{h}^{l+1}$ is the hidden state of the next BERT layer. The self-attention function can be written as

$$\text{SA}(\mathbf{h}^l) = LN(\mathbf{h}^l + MH(\mathbf{h}^l))$$

where MH is multi-head attention.

The PALs approach introduces task specific parameters that are used to compute a task specific function, updating the equation for a BERT layer to

$$\mathbf{h}^{l+1} = LN(SA(\mathbf{h}^l) + FFN(SA(\mathbf{h}^l)) + TS(\mathbf{h}^l))$$

Per Stickland and Murray (2019), the task specific function, TS, for PALs is

$$\text{TS}(\mathbf{h}^l) = V^D \; MH(V^E \mathbf{h}^l)$$

where V^D is a 'decoder' matrix of size $d_m \times d_s$ and V^E is an 'encoder' matrix of size $d_s \times d_m$ where $d_s < d_m$. For this project, $d_m = 768$, which is the size of the hidden state in the BERT model. Additionally, $d_s = 204$ for the full PALs model and $d_s = 276$ for the half PALs model because Stickland and Murray (2019) used these numbers in their full and half PALs implementation. Thus, a PAL simply projects the hidden state to a lower dimension, computes multi-head attention on that representation, and returns the re-scaled attention output.

A visual of the projected attention layer in parallel with a BERT layer can be found in Figure 1.

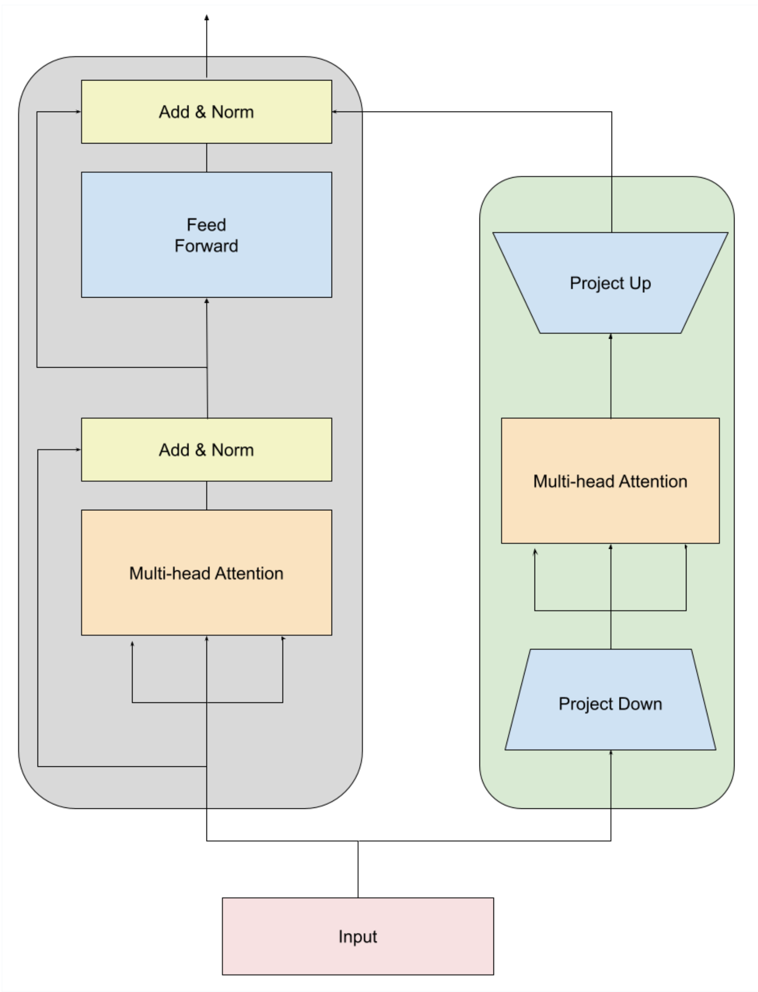

Figure 1: Schematic of a single layer. The large grey box represents the original BERT layer and the large green box represents the parallel projected attention layer.

4.4 Annealed Sampling

This project also implements the process of annealed sampling, as introduced in Stickland and Murray (2019). As discussed in their paper, annealed sampling helps to reduce overfitting on smaller datasets by choosing the next training batch according to a probability distribution defined by the size of the datasets. Specifically, the probability of choosing a dataset i is proportional to its size raised to a power:

$$\mathrm{p}_i \propto N_i^{\alpha}$$

The variable α is

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

where e is the current epoch and E is the total number of epochs. In the initial epochs, training favors the larger datasets; however, towards the end of training each dataset has a likely chance of being chosen. In this manner, annealed sampling helps reduce overfitting on smaller datasets.

5 Experiments

5.1 Data

The datasets used in the project are listed below.

- The Stanford Sentiment Treebank (SST) Dataset [1] is composed of 11,855 examples. Each example is a one sentence movie review that is labeled as "negative," "somewhat negative," "neutral," "somewhat positive," or "positive." The dataset is broken up into 8,544 training examples, 1,101 dev set examples, and 2,210 test examples.
- The CFIMDB Dataset [2] consists of 2,434 movie reviews that is labeled as either "positive" or "negative." The dataset is divided into 1,701 training examples, 245 validation examples, and 488 test examples.
- In the Quora Dataset [3] each example is a pair of sentences with a binary label indicating if the two sentences are paraphrases of each other. The dataset is broken into 141,498 training examples, 20,212 dev set examples, and 40,431 test examples.
- The SemEval STS Benchmark Dataset [4] contains 8,628 examples. Each example consists of a pair of sentences and a label from zero to five, indicating the level of similarity. A label of zero means that the two sentences are not related while a label of five indicates the two sentences share the same meaning. The dataset is divided into 6,040 training examples, 863 validation examples, and 1,725 test examples.

5.2 Evaluation method

The sentiment analysis and paraphrase detection tasks are evaluated based on accuracy. The Pearson correlation coefficient is used to evaluate the performance of the semantic textual similarity task. To compute an average across tasks, the Pearson correlation coefficient is normalized, producing a number between zero and one, and then all tasks are summed and divided over the total number of tasks. This process produces a result between zero and one that is used to assess the overall performance of the multi-task model. Values closer to one correspond to better performance.

5.3 Experimental details

The first part of this final project was to implement the base BERT model. This base implementation was trained and evaluated on the SST and CFIMDB datasets. It was trained in the "pretrain" fashion, meaning that the pretrained BERT weights were held constant while the top layer task parameters were adjusted. A learning rate of 1×10^{-3} was used. The base model was also trained in the "finetune" fashion, where all weights in the model were adjusted during training. A learning rate of 1×10^{-5} was used for the finetune mode. Both the pretrain and finetune training methods used a batch size of 8, a dropout probability of 0.3, and completed 10 epochs over the training data.

The second part of this final project, and the main focus of this report, is the implementation of a BERT and PALs model. This model used the SST, Quora, and STS datasets to train and evaluate on the sentiment analysis, paraphrase detection, and semantic textual similarity tasks, respectively. Cross-entropy loss was used in the sentiment analysis task while binary cross-entropy loss with logits was used for the paraphrase detection task. Mean squared error (MSE) was used for the semantic textual similarity task. As introduced in section 4.1 of this paper, there were three main types of models in this part of the project: zero PALs, half PALs, and full PALs. All of these models were trained the in "finetune" fashion with a learning rate of 1×10^{-5}, a batch size of 16, a dropout probability of 0.3, and a training time of 10 epochs over an annealed sample of the training data. These models took approximately 4 hours apiece to train and evaluate on a NVIDIA T4 GPU.

[1]https://nlp.stanford.edu/sentiment/treebank.html

[2]This project used a subset of the data found at https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

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

[4]https://aclanthology.org/S13-1004.pdf

5.4 Results

Table 1 shows the results for the first part of this project. Unsurprisingly, the results reveal that training under the "finetune" mode, adjusting all weights in the model, leads to higher accuracy.

Dataset	Training Mode	Accuracy
SST	Pretrain	0.390
SST	Finetune	0.520
CFIMDB	Pretrain	0.780
CFIMDB	Finetune	0.963

Table 1: Results for Part 1 of project - implementing baseline BERT model.

Table 2 contains the dev set results for the zero, half, and full PALs models. Interestingly, adding projected attention layers reduced average performance. This result was not expected. Due to this performance on the dev set, the zero PALs model was submitted as the best model for this project and it's results on the test set are shown in Table 3.

Model	**SST Dev Accuracy**	**Paraphrase Dev Accuracy**	**STS Dev Correlation (normalized)**	**Average**
Zero PALs	0.523	0.779	0.377	0.664
Half PALs	0.520	0.772	0.375	0.660
Full PALs	0.505	0.784	0.366	0.657

Table 2: Dev Set Results for the three main models of this paper reveals decreasing average accuracy with the addition of more PALs.

Model	**SST Test Accuracy**	**Paraphrase Test Accuracy**	**STS Test Correlation (normalized)**	**Average**
Zero PALs	0.530	0.785	0.331	0.660

Table 3: Test set results for the zero PALs model.

Due to the unexpected results of half and full PALs models, a full analysis of the tasks and PALs configuration was completed and the results are shown in Table 4. A complete discussion of this table and its meaning can be found in section 6 of this paper.

PALs	**Model Tasks**	**SST Dev Accuracy**	**Paraphrase Dev Accuracy**	**STS Dev Correlation (normalized)**	**Average**
Zero	SST	0.521	-	-	0.521
Zero	Paraphrase	-	0.784	-	0.784
Zero	STS	-	-	0.361	0.361
Zero	SST-Paraphrase	0.531	0.768	-	0.650
Zero	SST-STS	0.517	-	0.384	0.451
Zero	Paraphrase-STS	-	0.776	0.366	0.571
Half	SST-Paraphrase	0.514	0.778	-	0.646
Half	SST-STS	0.513	-	0.379	0.446
Half	Paraphrase-STS	-	0.781	0.364	0.573
Full	SST-Paraphrase	0.509	0.781	-	0.645
Full	SST-STS	0.521	-	0.349	0.435
Full	Paraphrase-STS	-	0.780	0.347	0.564

Table 4: Results for various BERT and PALs model structures performing on different tasks. The maximum score in each task column in underlined. The average score is taken based on the number of tasks the model was trained on, not the total number of tasks in the project (3). A "-" indicates the model was not trained on that task.

The top section of Table 4 shows how a zero PALs model performs on a single task. Since PALs is a multi-task method, projected attention layers were not used in the single task case. These single task models act as a guidepost for task performance. The bottom three sections of Table 4 show the performance of all combinations of PALs and two task configurations. Interestingly, the SST and STS tasks achieve better performance over their respective single task model when they are combined with another task.

6 Analysis

Finding the zero PALs model to perform better than the half and full PALs models on all three tasks was surprising. However, upon further inspection, this result should be expected because PALs do not increase the performance of all tasks.

In Stickland and Murray (2019), their BERT and PALs model is evaluated on the GLUE benchmark, which contains the SST, Quora, and STS databases. In the multiple GLUE benchmark tasks, their PALs model achieves various performance: beating, matching, or under-performing on some tasks when compared to the base BERT model. Specifically, their model under-performs on SST, out-performs on Quora, and matches performance on STS. One of the main takeaways from Stickland and Murray (2019) is that PALs can achieve a better average performance across all tasks in the GLUE benchmark. However, if we look at Stickland and Murray's average performance on the SST, Quora, and STS datasets alone, we see a different picture. Using the highest accuracy value under QQP in Table 2 of Stickland and Murray (2019), their base BERT model achieves an average of 89.5% while their PALs (204) model achieves an average of 89.2% on the three tasks. Their "PALs (204)" model is equivalent to what this project has been calling "full PALs." Thus, this project agrees with Stickland and Murray's results, finding that the base BERT model with no projected attention layers performs better than the full PALs model on the sentiment analysis, paraphrase detection, and semantic textual similarity multi-task problem.

This finding creates intrigue, leading to questions about task interference and determining when projected attention layers help performance. Table 4 provides insight into these two questions. First, we notice that the sentiment analysis task accuracy is greatest when trained with the paraphrase detection task, without using PALs. Also, STS correlation is at its greatest when trained with the sentiment analysis task and no PALs. These relationships indicate that there might be some task relationship between SST-Paraphrase and SST-STS. For both of these two task models, when the third task is added and still using zero PALs, the accuracies drop, as seen in Table 2. Therefore, it seems that there is some task interference between the paraphrase detection and semantic textual similarity tasks. This task interference leads to poor performance because progress on one task can cause regressions on the other task.

Looking at the relationship between the number of PALs and task accuracy, Table 4 seems to show that PALs helps improve paraphrase detection accuracy in the two task models. Additionally, in Table 2, the full PALs model achieves a higher paraphrase detection accuracy than the zero PALs model. Stickland and Murray (2019) also found that full PALs increased or maintained paraphrase detection accuracy over the base BERT model. Therefore, it appears PALs improves the paraphrase detection task.

7 Conclusion

This project implemented a BERT model with PALs and annealed sampling in order to complete the sentiment analysis, paraphrase detection, and semantic textual similarity tasks. The best model for this multi-task problem did not use PALs. This result led into a discussion of task interference and determining the tasks where PALs increases accuracy. Based on these three tasks alone, there appears to be task interference between the paraphrase detection and semantic textual similarity tasks. Additionally, it appears that PALs is particularly helpful at boosting performance of the paraphrase detection task in the multi-task learning setting.

Interesting directions for future work include determining the tasks in the GLUE benchmark where PALs increases accuracy the most. This work could extend to creating generalized rules for the type of tasks that benefit from PALs. Additionally, one might analyze the performance of adjusting model architecture so that PALs is only used for certain tasks (e.g. use PALs for paraphrase detection, but not on sentiment analysis or semantic textual similarity).

The key takeaway from this project is that PALs does not increase performance of all tasks. One must be judicious in determining what tasks will benefit from the addition of PALs. In short, choose your PALs wisely!

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.