Implementing Projected Attention Layers (PALs) for Joint Multi-Task Learning

Stanford CS224N Default Project

Juanhe (TJ) Tan
Department of Philosophy
Stanford University
`juanhe@stanford.edu`

Abstract

Joint multi-task learning shares operations and parameters across tasks, potentially saving computing time, energy, and storage compared to separately fine-tuning one model for each task. However, this can come at the cost of poorer task-specific performance. We explore ways of overcoming this challenge by adapting a base BERT model for joint learning of sentiment classification, paraphrase detection, and evaluation of semantic similarity. We do this by (i) implementing Projected Attention Layers (PALs), which are small, task-specific, multi-head attention layers that work alongside shared BERT layers, and (ii) trying different rates of sampling across tasks during training. We find that average performance across tasks improves when using PALs along with annealed sampling, a method that initially samples more from tasks with large datasets but samples more equally across tasks as training progresses. We also find that when the tasks we jointly train on are similar, annealed sampling seems to contribute more to performance improvements than PALs. However, if a task is dissimilar to the others and has a relatively small dataset, using either PALs or annealed sampling on their own can worsen performance, and it is important to use both together to dampen this effect. Task-specific performance can also be improved by task-specific pre-training after, but not before, joint multi-task fine-tuning.

1 Introduction

Suppose we want to adapt a large language base model (e.g. the Bidirectional Encoder Representations from Transformers model from Devlin et al. (2018), or BERT for short) for different natural language tasks. We can fine-tune the model *separately* for the different tasks, but this requires learning and storing many parameters – essentially, one full model per task. To reduce the total parameters needed, we can train the base model *jointly* on different tasks, so that we share most parameters across tasks, with only a small number of task-specific parameters. This reduces storage space, and can also save computing time and energy, if we share operations across tasks too. These are useful practical benefits, especially when using machines (e.g. mobile devices) that have tighter hardware constraints.

However, joint multi-task learning can lead to poorer task-specific performance, due to interference across tasks. In this project, we explore how we might overcome this challenge, by adapting a minimal BERT base model for the joint learning of sentiment classification, paraphrase detection, and evaluation of semantic textual similarity (STS). Specifically, we implement a version of the Projected Attention Layers (PALs) architecture, as well as various task sampling methods, proposed by Stickland and Murray (2019). PALs are task-specific multi-head attention layers that work alongside each BERT layer in the base model, while the different task sampling methods determine the probability with which we sample from tasks with different dataset sizes during training. (See Sections 3.2 and 3.3 for details.) In line with the results of Stickland and Murray (2019), we achieved the best performance when we used PALs along with the annealed sampling method, which starts by sampling more from tasks with larger datasets, but samples more equally across tasks as training progresses. One new question we explore in this project is: what is the relative contribution of PALs usage vs. task sampling to task-specific performance? We find that for tasks that are more similar to one another (in our case, paraphrase detection and STS evaluation), the right task sampling method improves performance more than using PALs. However, for tasks that are more dissimilar to the others (in our case, sentiment classification), using either PALs or annealed sampling on their own can worsen task-specific performance, likely due to cross-task interference. For such tasks, it is important to use both together to dampen this effect.

We then tried to improve the BERT with PALs model (with annealed sampling) in various ways: by (i) performing additional pre-training for tasks that the model does relatively worse in, either before or after joint multi-task fine-tuning, (ii) sharing parameters across the more similar tasks of paraphrase detection and STS evaluation, and (iii) adjusting regularization and training hyperparameters. We find that additional pre-training improves task-specific performance slightly, but only if performed after, not before, joint fine-tuning. Average performance can also be improved with a lower dropout probability rate and the use of more training examples per epoch. However, sharing parameters across similar tasks did not improve average performance across tasks.

2 Related Work

There are many ways to adapt a model for multi-task learning. Past attempts include using one neural network with different output layers for each task (e.g. Collobert et al. (2011)), or building one model with sequential layers that each outputs predictions for a different task (e.g. Hashimoto et al. (2016)). Following Stickland and Murray (2019), we can broadly categorize multi-task learning approaches into "hard" or "soft" parameter-sharing varieties. Soft parameter-sharing trains a separate model for each task, but regularizes the distance between the parameters of different models to make them more similar to one another. There is more flexibility in the parameters used for different tasks, but at the cost of using more parameters in total. Regularization can be done with various norms, such as L2 distance (e.g. Duong et al. (2015)) or the trace norm (e.g. Yang and Hospedales (2016)).

In contrast, hard parameter-sharing uses the same hidden layers for all tasks, but a different output layer for each task. This uses fewer total parameters than soft parameter-sharing, but the parameters have to "do more work" across different tasks. The approach in Stickland and Murray (2019), which we adapted for this project, can be considered a version of hard parameter-sharing, where the main BERT layers, which form the bulk of the model's parameters, are shared across tasks, though these are augmented by small task-specific PALs.

The spectrum from soft to hard parameter-sharing thus represents a key trade-off. The harder the parameter-sharing, the fewer total parameters we need, but the greater the risk of negative interference across tasks. How well we manage this trade-off may be influenced not only by our model architecture, but also by the way we train our model across the different tasks. In this project, we look at both of these factors, and explore which might make a bigger difference to the model's performance.

3 Approach

3.1 Adapting BERT for multiple tasks

We first implemented the minBERT model described in the Default Final Project (DFP) Handout (2023). minBERT consists of an embedding layer, followed by a stack of 12 BERT layers with hidden states of dimension d. Each BERT layer l ($l = 0, 1, \dots, 11$) comprises four layers in sequence: (i) a multi-head attention layer, (ii) an add-and-norm layer, (iii) a feed-forward network layer, and (iv) another add-and-norm layer. Using notation from the DFP Handout (2023), we can write:

$$MultiHeadNorm_l(x) = LN_{l,1}(x + MultiHead_l(Q_l(x), K_l(x), V_l(x))) \tag{1}$$
$$BertLayer_l(x) = LN_{l,2}(MultiHeadNorm_l(x) + FFN_l(MultiHeadNorm_l(x))) \tag{2}$$

where each LN_l is a layer normalization, x is the sequence of word piece embeddings for the input sentence, and $Q_l(x), K_l(x), V_l(x)$ are the query, key, and value vectors obtained from x. We also apply dropout with probability 0.3 to $MultiHead_l$ before $LN_{l,1}$, and to FFN_l before $LN_{l,2}$.

After going through these layers, the model outputs an embedding for each word piece of the sentence from the last BERT layer. For our natural language tasks, we use only the output embedding of [CLS], the standard token prepended to every input sentence (since this "pools" information from the whole sentence). For sentiment classification, which takes one sentence as input, we apply a linear layer to the [CLS] output embedding to get a score for each sentiment label, then pick the label with the highest score. For paraphrase detection and STS evaluation, which are two-sentence tasks, we apply a bilinear layer to the [CLS] output embeddings of the two input sentences to get our prediction.

$$pred_{\text{sent}} = \text{argmax}(W_{\text{sent}} h_1 + b_1), \qquad W_{\text{sent}} \in \mathbb{R}^{c\times d}, b_1 \in \mathbb{R}^{c\times 1} \tag{3}$$

$$pred_{\text{para}} = \sigma(h_2^\top W_{\text{para}} h_1 + b_2), \qquad W_{\text{para}} \in \mathbb{R}^{d\times d}, b_2 \in \mathbb{R}^{1\times 1} \tag{4}$$

$$pred_{\text{sts}} = h_2^\top W_{\text{sts}} h_1 + b_3, \qquad W_{\text{sts}} \in \mathbb{R}^{d\times d}, b_3 \in \mathbb{R}^{1\times 1} \tag{5}$$

Here, h_1, h_2 are the [CLS] output embeddings of our input sentences, c is the number of labels for sentiment classification, σ is the sigmoid function, W's are linear layers, and b's are bias terms. We apply dropout with the same probability of 0.3 to h_1, h_2 before prediction.

3.2 Projected Attention Layers (PALs)

We then implemented PALs as described in Stickland and Murray (2019), adapting the code there to suit our BERT implementation. At each BERT layer l, we inserted a PAL for each task t, which (i) uses a linear layer E_t to "encode" the BERT layer's input embeddings x into a smaller dimension $s < d$, (ii) applies multi-head attention to the encoded embeddings, and (iii) "decodes" the result back to dimension d with a linear layer D_t. The output is then put through a non-linearity (GELU, as used by Stickland and Murray (2019)) and added to $MultiHeadNorm(x) + FFN(MultiHeadNorm(x))$ from Equation (2), before the usual layer normalization by BERT, as below (g is the GELU function):

$$PAL_{l,t}(x) = D_t MultiHead_{l,t}(Q_{l,t}(E_t x), K_{l,t}(E_t x), V_{l,t}(E_t x)), \quad D_t \in \mathbb{R}^{d\times s}, E_t \in \mathbb{R}^{s\times d} \tag{6}$$

$$BertPal_{l,t}(x) = LN_{l,2}(MultiHeadNorm_l(x) + FN_l(MultiHeadNorm_l(x)) + g(PAL_{l,t}(x))) \tag{7}$$

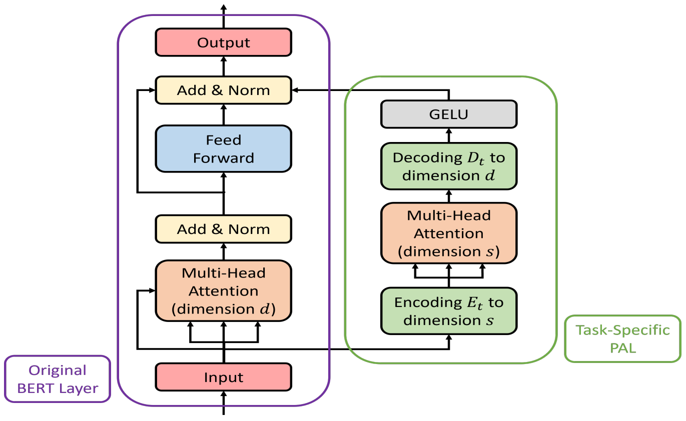
Output
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention (dimension d)
Input
Original BERT Layer
GELU
Decoding D_t to dimension d
Multi-Head Attention (dimension s)
Encoding E_t to dimension s
Task-Specific PAL

Figure 1: BERT layer with one task-specific Projected Attention Layer (PAL)

In essence, each PAL is a lower-dimensional, task-specific, multi-head attention layer that works "in parallel" with the usual multi-head attention and feed forward network in each BERT layer – the outputs from PAL and the usual BERT process are added-and-normed before being passed to the next BERT layer. Note that the "encoding" and "decoding" layers E_t, D_t are the same for each task across all BERT layers, but are different across tasks. We also omitted the matrix W^O (usually multiplied to the concatenated outputs of each attention head) in $MultiHead_{l,s}$, which Stickland and Murray (2019) found to be unnecessary. These moves reduce the number of parameters needed. Specifically, since the $MultiHead_{l,t}$ in Equation (6) works in dimension s rather than d, each $Q_{l,t}, K_{l,t}, V_{l,t}$ has s^2 parameters, while each E_t, D_t has sd parameters. Thus, across 12 layers and omitting bias terms, each task needs a total of $12(3s^2) + 2sd$ parameters for its PALs. In contrast, each BERT layer needs $4d^2$ parameters for the $MultiHead_l$ (d^2 for each of Q_l, K_l, V_l, W^O), and another $2dd_i$ for the FFN_l (where d_i is the size of the intermediate layer in the feed-forward network), for a total of $12(4d^2 + 2dd_i)$ parameters across 12 layers. This is much more than the parameters needed for PALs, when d and d_i are much larger than s (see Section 4.2 for concrete calculations).

3.3 Task sampling

One concern in joint multi-task learning is how to sample training examples across tasks. With a fixed batch size, a "round-robin" approach that cycles through tasks in a certain order when picking the next batch of examples will loop through smaller datasets more times, potentially overfitting on small datasets and underfitting on large ones. To avoid this, at each step, we can instead pick task t with a probability p_t that is proportional to the size of its dataset N_t raised to a power α:

$$p_t \propto N_t^{\alpha}, \quad 0 \leq \alpha \leq 1 \tag{8}$$

$\alpha = 1$ (**"proportional sampling"**) samples each task proportionally to the size of its dataset, which avoids the problems of the round-robin approach. But when dataset sizes are very different, we would end up training on small datasets only very rarely. On the other hand, $\alpha = 0$ (**"round-robin sampling"**) samples equally from every task, which could cause similar problems as the "round-robin" approach. Thus, $\alpha = 0.5$, which Stickland and Murray (2019) call **"square root sampling"**, is a possible compromise. Another approach, which Stickland and Murray (2019) call **"annealed sampling"**, would be to start with a higher α but taper it down over the training period:

$$\alpha = 1 - 0.8 \times \frac{\text{current epoch} - 1}{\text{total epochs} - 1} \tag{9}$$

This means that we sample across tasks more equally toward the end of training, when concerns about cross-task interference might be greatest, as Stickland and Murray (2019) suggest.

3.4 Further extensions

Finally, we experimented with original extensions to the approach of Stickland and Murray (2019) to try to improve our model's overall performance.

Additional pretraining. "Pre-training" means training our model while freezing the base BERT parameters, whereas "fine-tuning" updates the base BERT parameters during training as well. We initially followed Stickland and Murray (2019) in only fine-tuning our models, but noticed that the models did less well on sentiment classification and STS evaluation. Hence, to improve performance on these specific tasks, we did additional pre-training on them, either (i) before jointly fine-tuning the model across all three tasks, or (ii) after joint fine-tuning. For (i), the hope was that joint fine-tuning would get better performance on sentiment classification and STS evaluation if it starts from initial parameters that already do well on those tasks. For (ii), the idea was that after joint fine-tuning, we might get further task-specific gains through more optimization of only the task-specific parameters.

Additional parameter-sharing. Paraphrase detection and STS evaluation are similar tasks as both measure how close two sentences are in their meaning, but our training dataset is much larger for the former than for the latter. We thus tried sharing PALs across the two tasks, such that they shared all the same parameters except their output layers. The hope was that the similarity between the tasks would enable learning transfer from paraphrase detection to STS evaluation.

3.5 Baselines

Our baseline model was as provided in the DFP Handout (2023). It uses only the base minBERT model (i.e. without PALs) and is pre-trained only on the sentiment classification task. Against this baseline model, we compared different models, representing various combinations of (i) whether we used just minBERT, or minBERT with PALs; (ii) different task sampling methods (i.e. round-robin, square root, proportional, or annealed sampling); and (iii) our various proposed extensions.

4 Experiments

4.1 Data and evaluation methods

Our datasets and evaluation methods are as described in the DFP Handout (2023):

- **Sentiment classification**: Stanford Sentiment Treebank (SST). Contains 11,855 examples, split into a training set (8,544 examples), a development set (1,101), and a test set (2,210). Each example is a sentence with one of five labels (negative, somewhat negative, neutral, somewhat positive, positive). Evaluation method is accuracy of predicted label.
- **Paraphrase detection**: Subset of Quora dataset. Contains 202,152 examples (training: 141,506; development: 20,215; test: 40,431); each is a pair of sentences, plus one of two labels (is paraphrase, is not paraphrase). Evaluation method is accuracy of predicted label.
- **STS evaluation**: SemEval STS Benchmark Dataset. Contains 8,628 examples (training: 6,041, development: 864, test: 1,726); each is a pair of sentences, plus a similarity score from 0 to 5 (higher means more similar). Evaluation method is Pearson correlation between predicted similarity score and true similarity score.

4.2 Experimental details

We used the minBERT model as described in the DFP Handout (2023), with $d = 768$. After some initial experiments (see Appendix A.1[1] for details), we settled on the following parameters for our project, to try to optimize our results taking into account our available compute time and resources.

Main set. Using only fine-tuning, we trained eight models, corresponding to the eight combinations of (i) whether we used minBERT only or minBERT with PALs, and (ii) whether we used round-robin, square root, proportional, or annealed sampling. In each training run, we trained a model for 10 epochs, choosing the best set of parameters from the epoch that got the highest average development set score over the three tasks. Per epoch, the number of training examples used was set at $TP = 50\%$ of the total number of training examples available in all datasets (TP here stands for "training proportion"). For our batch size of 16, this works out to 4,877 batches per epoch (rounding down to get an integer). For our optimizer, we used AdamW (as described in the DFP Handout (2023)), with learning rate 2×10^-5 and weight decay of 0.01. For our loss functions, we used cross entropy for sentiment classification, and binary cross entropy for paraphrase detection and STS evaluation. [2]

For each PAL, we used 12 attention heads and a dimension of $s = 324$. Each task thus required $12(3s^2) + 2sd \approx 43$m parameters for its PALs, or only about 5% of the $12(4d^2 + 2dd_i) \approx 85$m parameters for the BERT layers themselves (as $d_i = 3,072$ for our minBERT model). Hence, adding the PALs parameters for the three tasks to the original BERT layer parameters, we used a total of $\approx 1.15\times$ the number of BERT layer parameters for minBERT and PALs, compared to the $3\times$ we would have used if we fine-tuned the whole minBERT model separately for the three tasks.

Extensions. Next, we tested our proposed original extensions to the PALs approach, and did hyperparameter searches. Taking the best-performing model from our main set of experiments, we tested: (i) additional pre-training on sentiment classification and STS evaluation (10 epochs, learning rate 1×10^{-4}, batch size of 16, all training examples used per epoch) *before* vs. *after* joint multi-task fine-tuning, (ii) additional parameter-sharing between paraphrase detection and STS evaluation, and (iii) lower dropout probability of 0.1 and higher TP of 75%.

4.3 Results and discussion

After rescaling to a scale of 100 points, our development set results are given in Tables 1 and 2

[1] This also has the details of the first part of the DFP i.e. the single task sentiment classifier that we built.

[2] For STS evaluation, we divided the similarity score by 5 to get a probability score between 0 and 1 for each example, so that we could compute binary cross entropy between this probability score and the logits predicted by our model's STS evaluation output.

Table 1: Main set, PALs dimension $s = 324$, dropout probability $= 0.3$, $TP = 50\%$

Model	Task Sampling	SST	Quora	STS	Avg.
minBERT only	Fine-tune on SST only	52.6	-	-	-
minBERT only	Pre-train on SST only (Baseline)	40.2	42.2	4.8	29.1
minBERT only	Round-robin ($\alpha = 0$)	50.7	76.4	39.5	55.5
minBERT only	Square root ($\alpha = 0.5$)	50.0	81.1	44.0	58.4
minBERT only	Proportional ($\alpha = 1$)	48.6	80.1	45.7	58.1
minBERT only	Annealed ($\alpha = 1 - 0.8 \times \frac{\text{current epoch}-1}{\text{total epochs}-1}$)	49.0	80.1	47.2	58.7
minBERT with PALs	Round-robin ($\alpha = 0$)	49.3	77.7	41.0	56.0
minBERT with PALs	Square root ($\alpha = 0.5$)	50.7	80.2	47.8	59.6
minBERT with PALs	Proportional ($\alpha = 1$)	51.2	79.3	50.6	60.4
minBERT with PALs	Annealed ($\alpha = 1 - 0.8 \times \frac{\text{current epoch}-1}{\text{total epochs}-1}$)	50.1	80.3	51.2	60.6

Table 2: Extensions for minBERT with PALs and annealed sampling from main set

Extension	SST	Quora	STS	Avg.
Additional pre-training on SST and STS *before* joint fine-tuning	49.5	81.1	45.8	58.8
Additional pre-training on SST and STS *after* joint fine-tuning	51.3	80.3	52.6	61.4
Additional parameter-sharing for Quora and STS	51.0	81.5	47.4	60.0
Dropout probability = 0.1	49.7	81.0	53.6	61.4
Dropout probability = 0.1 and $TP = 75\%$	51.4	80.1	55.9	62.5

Main set. For the same task sampling method, the use of PALs improved average development set performance, by 0.5 to 2.3 points (or about 1% to 4%). This is expected, as the PALs contain extra task-specific parameters, which can encode more information that can then be used to improve performance. Among our task sampling methods, both when we used minBERT only and when we also used PALs, annealed sampling performed the best, while round-robin sampling performed the worst. This vindicates the hypothesis that, when training datasets are of very different sizes, sampling tasks equally may cause overfitting on small datasets and underfitting on large ones. It is better to take a more balanced annealed approach, sampling more from large datasets first and sampling more equally across tasks toward the end of training. As for proportional and square root sampling, both did better than round-robin sampling and worse than annealed sampling, but proportional sampling did better when PALs were used and square root sampling did better without PALs. This is contrary to the results of Stickland and Murray (2019), which found that square root sampling always did better. See Section 5.2 for more detailed analysis of the interaction between PALs and task sampling.

Extensions. Additional pre-training on SST and STS before joint fine-tuning *reduced* the model's performance on those two tasks, especially STS evaluation, while additional pre-training after joint fine-tuning improved average performance by 0.8 points (about 1%). This suggests that any gains made by pre-training before joint fine-tuning might have been wiped out by cross-task interference through the joint fine-tuning process, whereas pre-training after joint fine-tuning could still find localized task-specific improvements, while keeping the BERT parameters fixed. Additional parameter-sharing between paraphrase detection and STS evaluation, on the other hand, worsened STS performance, while improving paraphrase detection performance. This suggests that any transfer learning that took place was insufficient to outweigh the effect of the parameters being skewed more toward paraphrase detection (due to the much larger Quora training dataset).

Hyperparameter search. Reducing our dropout probability from 0.3 to 0.1 improved our average development score by 1.0 point (about 2%). This could be because the number of parameters added to BERT through PALs and the task output layers was not large enough that a higher dropout probability was needed for regularization (considering that the base BERT model had already been robustly trained previously). Increasing the number of batches trained during each epoch also improved performance, by a further 1.1 points (about 2%), suggesting that there was potential to improve performance by subjecting the model to longer training and exposure to more training examples.

Final tested model. We combined the extensions that improved performance to get our final model, as specified in Table 3, together with our final results on both the development and test sets.

Table 3: Final model: minBERT with PALs ($s = 324$), annealed sampling, dropout probability of 0.1, $TP = 75\%$, additional pre-training on SST and STS after joint fine-tuning

	SST	Quora	STS	Avg.
Development set	51.4	80.1	56.8	62.8
Test set	52.2	80.7	56.3	63.1

Our test set results were in line with our expectations from the development set results. Our final model did significantly better on all tasks than our baseline model, which was pre-trained only for sentiment classification. Our final model also outperformed, on all tasks, the basic minBERT only, round-robin sampling model, demonstrating the combined contribution of PALs, annealed sampling, and our various extensions. While our final model did decently on paraphrase detection, it did less well on sentiment classification and STS evaluation. More detailed analysis of this is in section 5.1

5 Analysis

5.1 Task-specific performance of PALs

The main challenge we set out to overcome in multi-task learning was to achieve good task-specific performance without adding too many parameters. We achieved some success on this, but there is much room for improvement. Notably, our models all performed best on paraphrase detection, likely because the Quora dataset was far larger than the others. But our models did less well on sentiment classification (which was our final model's lowest-scoring task). This could be because sentiment classification is the "odd one out" among our three tasks – the other two tasks are more similar to each other as they both measure the closeness in meaning between sentences, rather than their sentiment.[3] Nonetheless, our final model's SST test set score (52.2) was not much worse than the development set score of our minBERT-only model when fine-tuned only on SST (52.6), which suggests that our final model did not do too badly on sentiment classification considering the likely cross-task interference.

As for STS evaluation, our final model performed much better (test set score of 56.3) than our minBERT only model with round-robin sampling (development set score of 39.5). This was the largest improvement among the three tasks, and was likely due to the similarities between the paraphrase detection and STS evaluation tasks, which enabled some learning transfer from the former to the latter. Nonetheless, our final STS test score was still not as high as we might have hoped, e.g. when ranked on the test score leaderboard. The reason for this is unclear. One possibility is that, as STS evaluation is measured with Pearson correlation rather than accuracy, we might need to change our output layer and loss function to better optimize for this metric. To uncover other possible reasons, we qualitatively analyzed the top ten examples where our model made wrong predictions of similarity and of dissimilarity. (See Appendix A.2 for the examples.)

We found that our model tended to wrongly predict that two sentences were dissimilar when (i) the sentences use (near-)synonyms (e.g. "cat" vs. "kitten", "expands" vs. "widens"), or (ii) one of the sentences uses a longer locution (e.g. "A woman and man are riding in a car" vs. "A woman driving a car is talking to the man seated beside her.") or an extra word that is uncommon (e.g. "overwhelmingly", "overhaul"). This suggests that the model may be overweighting the particularity of some words or phrases in differentiating between sentence meanings, perhaps because the base BERT model's representations of these words or phrases are too fine-grained for STS evaluation.

On the other hand, our model tended to wrongly predict that sentences were similar when they shared a similar grammatical structure, but comprised words with different meanings, e.g. "A person is boiling noodles" vs. "A cat is licking a bottle", which both fit "A [noun] is [verb-ing] [noun phrase]". In some cases, one word or phrase completely changes what's being described, e.g. "3 killed, 4 injured in Los Angeles shootings" vs. "Five killed in Saudi Arabia shooting" are both about shootings

[3]This is corroborated by Stickland and Murray (2019), who compared the performance of BERT with PALs on each task to a baseline BERT model singly fine-tuned on that task. There, BERT with PALs fell further short of the baseline BERT model performance on SST, than it did for most of the other tasks.

and casualties, but the different place name means they are about totally separate events, which our model did not clearly recognize. This suggests that the model may rely too much on sentence structure and not enough on "world" knowledge to distinguish sentence meanings, even though this knowledge might exist in the base BERT model (which is trained on many real-world documents).

To correct these errors, one strategy could be to add more layers on top of our model, instead of within the BERT layers, so that the model can use these top layers to better learn which parts of the base model's sentence representations it should look at more closely to distinguish two sentences (e.g. the "world" knowledge of specific terms), and which differences it should ignore (e.g. the use of near-synonyms). The information needed for STS evaluation may already exist in the BERT representations (augmented by PALs), but what we need is perhaps a better way for the model to transform these representations into the form needed for the specific evaluation task.

5.2 Interactions between PALs and task sampling

Another question we had set out to explore was the relative importance of using PALs vs. better task sampling. Since round-robin sampling did worst and annealed sampling did best overall, we took our minBERT only model with round-robin sampling as a baseline, and computed the improvement on each task when we (i) switched to annealed sampling only (without PALs), (ii) used PALs only (without annealed sampling), and (iii) used both annealed sampling and PALs. (See Figure 2.)

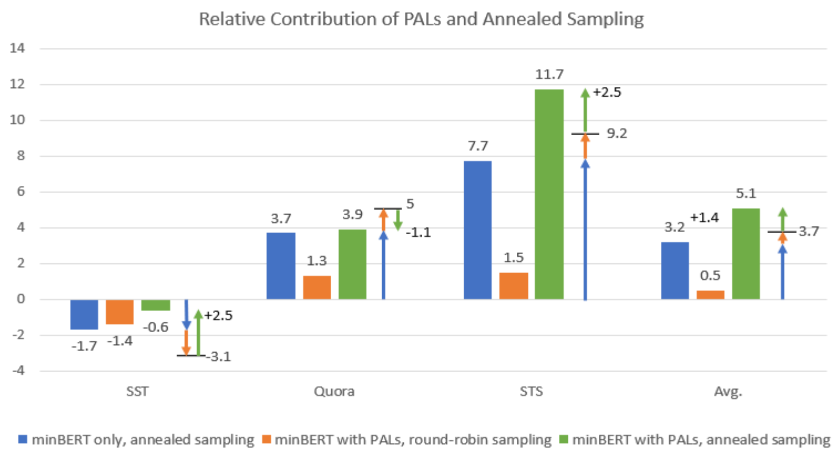

Figure 2: Increase in task performance (measured in points), compared to baseline model (minBERT only, round-robin sampling). Black lines indicate expected performance if we just add contributions from both annealed sampling (blue arrows) and PALs (orange arrows). Green arrows represent interaction effects between annealed sampling and PALs that get us from the black lines to the actual performance improvement from using both annealed sampling and PALs.

For paraphrase detection and STS evaluation, using PALs improved performance much less compared to switching from round-robin to annealed sampling. (The blue bars are much taller than the orange ones.) This suggests that in joint fine-tuning on similar tasks, using the right task sampling method may be more important than adding more task-specific parameters, likely because task-specific parameters are needed less when tasks are more similar. Also, interaction effects between PALs and annealed sampling were positive for STS evaluation but negative for paraphrase detection. This could be because, as the task with the largest training dataset, paraphrase detection would already have benefited significantly from more proportional task sampling, such that there are not much more gains to be had from using PALs too.

In contrast, for sentiment classification, annealed sampling and PALs actually *worsened* performance when used on their own, but using both together dampened the effect. This could be because sentiment classification has a much smaller training dataset than paraphrase detection, and is also quite different from the other two tasks, leaving it more vulnerable to cross-task interference. Since annealed sampling would have trained less on sentiment classification than round-robin sampling, cross-task interference could have increased, and without task-specific PALs to counteract this, performance might have suffered more as a result. This is supported by Table 1, which shows that without PALs, SST performance *decreases* as task sampling becomes more proportional (i.e. as α increases).

As for PALs, using them means that during training, the model has to balance between updating the shared base BERT parameters and the task-specific PALs. That SST performance worsened when PALs were introduced suggests that round-robin sampling does not get this balance right. One reason could be that, since the other two tasks are more similar, joint fine-tuning is likely to skew the shared parameters more towards those tasks, and so the sentiment classification PALs need to "work harder" to do well on SST. If so, then a more proportional task sampling method like annealed sampling could help, because it would train on SST less often, allowing other tasks to update the shared parameters more often in between. Then, when it is time to train on SST again, the AdamW optimizer might push the sentiment classification PALs more decisively (given the infrequency of their updates) toward where they need to be to counteract the skew in the shared parameters. This hypothesis is supported by Table 1, which shows that with PALs, SST performance *increases* as task sampling becomes more proportional. All this suggests that in joint fine-tuning, if there are tasks that are dissimilar to the rest, it is important to use both PALs and annealed sampling together, rather than either in isolation.

6 Conclusion

We conclude that in joint multi-task fine-tuning, the use of both PALs and annealed sampling can improve average task performance. This is in line with the findings of Stickland and Murray (2019). Our results build on their work by suggesting that, when the tasks that we are jointly fine-tuning on are similar to one another, using annealed sampling would likely contribute more to performance improvement than using PALs. But if there is a task that is quite dissimilar to the others, it is important to use both PALs and annealed sampling together. Moreover, we found that after (but not before) joint fine-tuning, task-specific pre-training can help to further improve performance.

One major limitation of this work was that we did not have the compute time and resources to repeat training runs with different seeds, to rule out randomization effects. Our project was also limited to joint fine-tuning on just three tasks. More work would be needed to verify the replicability and generalizability of our results when training on more tasks. Our project also did not achieve as high scores as other teams on the leaderboard, which might suggest more fundamental limitations with our approach. To some extent, this is expected as we had actively constrained ourselves to adding only a small number of parameters. But even with this constraint, there might be significant gains that we missed out on, especially for STS evaluation, which could perhaps be obtained through clever tweaks of our output layer and loss function. Something to think about for next time!

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76):2493–2537.

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.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China. Association for Computational Linguistics.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.01587.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Yongxin Yang and Timothy M. Hospedales. 2016. Trace norm regularised deep multi-task learning. *CoRR*, abs/1606.04038.

A Appendix (optional)

A.1 Initial experiment results

In the first part of this project, we built a classifier only for sentiment classification, and pre-trained or fine-tuned it separately on the SST and CFIMDB datasets. (CFIMDB dataset: 2,434 examples divided into train set of 1,701, development set of 245, and test set of 488; each example is a movie review with a binary sentiment classification of negative or positive. Evaluation method is accuracy.) Our development set results are reported in Table 4

Table 4: minBERT sentiment classifier development set results

Training Method	Accuracy
Pre-trained on SST	39.1
Fine-tuned on SST	52.6
Pre-trained on CFIMDB	77.1
Fine-tuned on CFIMDB	95.9

We then extended our minBERT model for multi-task learning of sentiment classification, paraphrase detection, and STS evaluation. Initially, using only fine-tuning, we trained four models, corresponding to the four combinations of (i) whether we used minBERT only or minBERT with PALs, and (ii) whether we used square root or annealed sampling, since these were the two preferred methods of Stickland and Murray (2019). We set $TP = 50\%$ and used a batch size of 8, which worked out to 9,755 batches per epoch. For our loss functions, we used cross entropy, binary cross entropy, and mean squared error against the correct labels, for sentiment classification, paraphrase detection, and STS evaluation respectively.

For each PAL, we used 12 attention heads and a dimension of $s = 204$ (same as the set-up of Stickland and Murray (2019)). Each task thus required $12(3s^2) + 2sd \approx 1.8$m parameters for its PALS, only about 2% of the ≈ 85m parameters for the BERT layers themselves.

Our results are summarized in Table 5. As the performance was not very good (< 50 for SST and STS), we decided to increase the PAL size to $s = 324$. We also increased batch size to 16 to speed up our training runs. Finally, we also switched our loss function for STS evaluation to binary cross entropy, as this led to better results.

Table 5: Initial set, PALs $s = 204$, STS loss function = mean squared error, batch size = 8, $TP = 50\%$

Model	Task Sampling	SST	Quora	STS	Avg.
minBERT only	Square root sampling	49.2	80.2	39.1	56.2
minBERT only	Annealed sampling	48.7	79.0	44.3	57.3
minBERT with PALs	Square root sampling	48.4	80.4	39.0	56.0
minBERT with PALs	Annealed sampling	49.5	79.4	45.5	58.1

A.2 Qualitative analysis of STS evaluation errors

We ran the sigmoid function on the logits predicted by our final model for STS evaluation, then multiplied by 5 and subtracted the result from the true similarity score to get a difference score. A negative score means the model predicted the sentences to be less close than they really are, while a positive score means the model predicted the sentences to be closer than they really are. The maximum difference score is 5 and the minimum is -5.

Table 6: Top 10 examples where model wrongly predicted that sentences were not similar

Sentence 1	Sentence 2	Difference Score
Microsoft says to cut up to 18,000 jobs	Microsoft to cut 18,000 jobs in overhaul	4.0
A girl is talking to her dad on a cell-phone.	a girl is talking on her phone.	3.8
Stock index futures point to lower start	Stock index futures signal early losses	3.4
German parliament overwhelmingly backs Greek rescue deal	German parliament endorses Greek bailout	3.3
They have actually showed a cooling trend since due to the recent decline.	get a cooling trend from because of the recent decline.	3.3
Obama to announce gun control plans Wednesday	Obama to set out gun control plans	3.3
A woman and man are riding in a car.	A woman driving a car is talking to the man seated beside her.	3.1
A cat is drinking milk.	A kitten is drinking milk.	3.1
A close-up of a sheep with its tongue hanging out.	Close up image of a sheep with it's tongue hanging out.	3.1
Israel expands subsidies to settlements	Israel widens settlement subsidies	3.0

Table 7: Top 10 examples where model wrongly predicted that sentences were similar

Sentence 1	Sentence 2	Difference Score
Work into it slowly.	It seems to work.	-4.0
France cleared to impose controversial 'millionaire tax'	Manila to increase counter-terror team	-3.6
A person is boiling noodles.	A cat is licking a bottle.	-3.6
Chris Froome rides to Tour de France endgame	Chris Brown concerts cancelled new	-3.6
A little girl in an orange striped outfit is airborne whilst bouncing on a bed.	A dog in a red shirt is chasing a squirrel through the glass.	-3.5
Taiwan gang leader nabbed at airport after 17 years	Zambia arrests ex-leader Banda over oil deal	-3.3
3 killed, 4 injured in Los Angeles shootings	Five killed in Saudi Arabia shooting	-3.2
Zoo worker dies after tiger attack	Indian-origin teacher dies after attack in New Zealand	-3.1
A submarine is going through water.	A baby is falling asleep.	-3.1
A woman posing in front of an apartment building in the snow.	White bus parked in grass in front of building.	-3.1