PALs of MTL: Investigating Task Scheduling Algorithms in the Presence of PALs

Stanford CS224N Default Project

Kris Jeong
Department of Computer Science
Stanford University
kjeong@stanford.edu

Pauline Arnoud
Department of Computer Science
Stanford University
parnoud@stanford.edu

Abstract

Multitask learning (MTL) in natural language processing promises enhanced efficiency by leveraging shared representations across tasks, but faces challenges like task interference and resource imbalance. To address these, we explore the integration of Projected Attention Layers (PALs) with task scheduling algorithms (round robin, annealed sampling, and dynamic prioritization) on a multitask BERT model for sentiment analysis, paraphrase detection, and semantic similarity. Through extensive ablation studies, we isolate the impacts of PALs and schedulers, and propose optimizations for less-resourced tasks. Our findings suggest that while PALs offer theoretical benefits, their practical implementation does not consistently outperform baseline models. However, effective task scheduling, particularly annealed sampling and dynamic prioritization, significantly enhances multitask performance. These findings underscore the nuanced trade-offs in MTL and the need for context-specific strategies to balance shared and task-specific learning.

1 Key Information to include

Our mentor is Johnny Chang. We have no external collaborators, and we are not sharing the project with any other course.

2 Introduction

Since their introduction in 2017 by Vaswani et al. (2023), transformers have established themselves as the reference model for natural language processing. BERT's paradigm of pretraining on vast datasets followed by fine-tuning on specific downstream tasks has demonstrated unprecedented performance improvements Devlin et al. (2019). However, the sheer size and computational demands of transformers pose significant challenges when fine-tuning models individually for each task due to the extensive computational resources required and the potential redundancy in training separate models for each task. This leads to a pertinent question: can we efficiently train a single multitask model that leverages the strengths of BERT while mitigating the associated computational overhead?

Multitask learning (MTL) offers a promising avenue to address this challenge by training a model on multiple tasks simultaneously, leveraging commonalities across tasks to improve generalization, performance, and efficiency. However, MTL introduces its own set of challenges, notably task interference and the difficulty of balancing learning across tasks with varying resource availability.

In this paper, we extend the works of Stickland and Murray (2019) and Guo et al. (2018), exploring the interaction of Projected Attention Layers (PALs) with three established task scheduling algorithms: round robin, annealed sampling, and dynamic task scheduling. We conduct rigorous comparisons and ablation studies to identify the optimal combination of PALs and task scheduling strategies for enhancing MTL performance. We evaluate the performance of our multitask BERT classifier on three tasks: sentiment analysis (SST), paraphrase detection (Quora), and semantic textual similarity (STS).

Our primary contributions are as follows: we conduct extensive experiments with various combinations of PALs and three established task schedulers to evaluate their impact on multitask learning performance; we perform ablation studies to isolate the influence of the number of PALs versus task schedulers on task performance and interference; and we propose optimization techniques, including bootstrapping and increasing batch sizes, to enhance the performance of less-resourced tasks, specifically in the context of PALs.

Our experimental results demonstrate that incorporating PALs, while promising in theory, does not always yield better results compared to our baselines. However, employing effective task scheduling strategies significantly improves performance on multitask learning benchmarks. Through detailed analyses, we provide insights into the best practices for implementing MTL models in NLP, highlighting the nuanced trade-offs between task-specific and shared learning in the context of PALs and task scheduling.

3 Related Work

We draw inspiration from two key works for this investigation. The first is Stickland and Murray (2019), who introduced PALs to address the limitations of traditional MTL approaches with BERT. Traditional approaches either involve fine-tuning separate models for each task, which is computationally expensive and inefficient, or sharing all parameters across tasks, which can lead to suboptimal performance due to task interference. PALs offer a more nuanced approach by adding low-dimensional multi-head attention layers specific to each task in parallel to standard BERT layers. This allows for efficient task-specific adaptations while sharing most parameters across tasks. The authors also introduced a novel task scheduling strategy termed "annealed sampling" to mitigate the problem of resource imbalance across tasks. Despite demonstrating promising results, their work identified several limitations that warrant further investigation, particularly the interaction between PALs and established anti-interference methods such as task scheduling algorithms.

The second work we draw from is Guo et al. (2018), who introduced and exhaustively tested a dynamic task prioritization algorithm. Unlike traditional methods that treat all tasks equally or rely on fixed schedules, their approach dynamically adjusts the priority of tasks based on their difficulty and learning progress. By focusing on more challenging tasks, the model can allocate resources more effectively, improving overall performance. Their method uses key performance metrics (KPIs) to assess task difficulty and dynamically adjusts the mixing weights of each task's loss objective. This approach ensures that more difficult tasks receive greater attention during training, mitigating the risk of easy tasks dominating the learning process. Their experiments demonstrated significant performance improvements across multiple computer vision tasks, highlighting the potential of dynamic task prioritization to enhance MTL frameworks.

In our project, we implement simplified versions of Stickland and Murray (2019) and Guo et al. (2018)'s task schedulers alongside a simple round robin scheduler. We then test them in conjunction with various numbers of PAL layers, another area Stickland and Murray (2019) highlighted as necessitating future investigation. By examining these combinations, we aim to address the open questions left by previous work and advance the understanding of effective MTL strategies in NLP.

4 Approach

4.1 Baselines

We adapted a single-task BERT classifier, initially fine-tuned for sentiment analysis in the `minBERT` section of the Project Handout, into a multitask model by adding two linear classification layers for paraphrase detection and semantic textual similarity. We updated the codebase to include task-specific dataloaders, forward passes, loss calculations, and a total loss as the sum of the three task-specific losses. We then generate two baselines from this classifier:

- Sentiment Data Only: This baseline serves as a lower bound, highlighting the model's inherent limitations when trained exclusively on a single task. It helps to illustrate the necessity of exposing the model to a diverse set of tasks for achieving balanced multitask performance.

- Sequential Training: This configuration represents a naive approach to MTL. By training the model on each task sequentially, we can observe the cumulative effect of task learning and potential task interference. It serves as a comparative measure against more sophisticated task scheduling methods, allowing us to assess the incremental benefits of advanced strategies.

4.2 Extension 1. Implementing PALs from Scratch

For our first extension, we follow Stickland and Murray (2019)'s architecture to implement PALs from scratch, embedding them into our codebase. PALs are low-dimensional multi-head attention layers added in parallel to the standard BERT layers, allowing for efficient task-specific adaptations with significantly fewer parameters than duplicating the entire model for each new task. In our implementation, each BERT layer is enhanced with a PAL module that consists of the following components:

- Dimensionality Reduction: We first project the hidden states down to a lower dimension using a linear transformation.
- Self-Attention: The projected hidden states are then passed through a self-attention mechanism, which is configured with the reduced hidden size.
- Dimensionality Restoration: The attended hidden states are projected back up to the original dimension using another linear transformation.
- Activation Function: A GELU activation function is applied to the output.

Let n be the number of PAL layers to activate, specified by the user. The code adds the PAL output to the self-attention output of the last n standard BERT layers, followed by layer normalization and dropout. We set the default value of n to 6, a heuristic suggested by the original paper. This reflects the following formula given by the authors:

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

where l is the index of the layer; $LN(\cdot)$ is layer-norm; $SA(\cdot)$ is the self-attention layer; $TS(\cdot)$ is the task-specific function of the form:

$$TS(\mathbf{h}) = V^D g(V^E \mathbf{h})$$

where V^E is a $d_s \times d_m$ encoder matrix; V_S is a $d_m \times d_s$ decoder matrix with $d_s < d_m$; and $g(\cdot)$ is multi-head attention with shared V^E and V^D across layers, not tasks. This implementation was reported to be the highest performing version by the authors.

4.3 Extension 2. Implementing Task Schedulers

Next, we implement three task scheduling algorithms, an established method in MTL of balancing learning across tasks and mitigating the effects of task interference.

Round Robin Scheduling is a widely used baseline in MTL due to its simplicity and effectiveness. In this approach, the model is trained on each task in a cyclic order, ensuring that all tasks receive equal training time. This method helps in maintaining a balanced learning process across tasks without any explicit prioritization. Although round robin scheduling does not account for task difficulty or resource availability, it often serves as a robust baseline for evaluating more sophisticated scheduling strategies. In our implementation, we created iterators for each task-specific dataloader and cycled through them in a round robin manner. The optimizer was reset, and the model parameters were updated after processing each batch.

Annealed sampling, introduced by Stickland and Murray (2019), addresses the limitations of traditional sampling strategies by dynamically adjusting the sampling probability of tasks based on their resource availability. The goal is to avoid favoring well-resourced tasks or overfitting low-resource tasks. Initially, tasks are sampled with a probability proportional to the square root of the number of training examples they have, defined as $p_i \propto N_i^{0.5}$, where N_i is the number of training examples for task i. However, to mitigate task interference over time, the sampling probability is adjusted with each epoch using the annealed parameter α:

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

where e is the current epoch and E is the total number of epochs. This method ensures that tasks are trained more equally towards the end of the training process. In our implementation, we updated the sampling probabilities at the beginning of each epoch based on the annealed parameter and randomly selected tasks according to these probabilities for each batch.

Dynamic Task Scheduling aims to prioritize more challenging tasks during training, thereby allocating resources more effectively and improving overall performance. In this paper, we implement the algorithm proposed by Guo et al. (2018). We first calculate what the authors term the "KPI" $\overline{\kappa}_t$ for each task, which is an exponential moving average (EMA) of its performance metric:

$$\overline{\kappa}_t^{(\tau)} = \alpha\kappa^{(\tau)} + (1-\alpha)\overline{\kappa}_t^{(\tau-1)}$$

where κ_t is the KPI for task t at iteration τ, and α is the discount factor. The difficulty, and therefore priority, of each task is calculated by a variant of the focal loss function. The total loss for the model uses these priorities as weights to produce a weighted sum of the task-specific losses:

$$L_{\text{Total}}(\cdot) = \sum_{t=1}^{|T|} \text{FL}(\overline{\kappa}_t; \gamma_t) L_t(\cdot) = \sum_{t=1}^{|T|} (1-\overline{\kappa}_t)^{\gamma_t} L_t(\cdot)$$

Tasks with lower KPIs are prioritized, ensuring that more difficult tasks receive greater attention. This approach prevents easy tasks from dominating the learning process and ensures a balanced allocation of resources. In our implementation, we dynamically calculated task priorities using the EMA of their performance metrics and adjusted the task selection probabilities and loss weights accordingly during training.

To our knowledge, applying dynamic task scheduling in conjunction with PALs has not been done before and we thus consider it our **original contribution**. We also implement everything in the extensions sections from scratch and provide comprehensive ablation studies in the context of PALs, which we consider to be our primary contributions.

5 Experiments

5.1 Data

We used the Stanford Sentiment Treebank (SST) dataset for sentiment analysis, the Quora dataset for paraphrase detection, and the SemEval STS Benchmark dataset for semantic textual similarity (STS).

Stanford Sentiment Treebank (SST) Dataset: 11,855 single sentences from movie reviews, each annotated with one of five sentiment categories: negative, somewhat negative, neutral, somewhat positive, or positive. It's divided into training (8,544 examples), dev (1,101 examples), and test (2,210 examples) sets. BERT embeddings are used for sentiment classification.

Quora Dataset: 404,298 question pairs labeled as paraphrases or not. It's partitioned into train (283,010 examples), dev (40,429 examples), and test (80,859 examples) sets.

SemEval STS Benchmark Dataset: 8,628 sentence pairs with similarity scores between 0 (unrelated) and 5 (equivalent meaning). It's split into train (6,040 examples), dev (863 examples), and test (1,725 examples) sets.

5.2 Evaluation method

We use accuracy for the sentiment classification and paraphrasing tasks, measuring the proportion of correctly predicted sentiment labels and correctly identified paraphrases respectively. We use the Pearson correlation coefficient for the semantic similarity task. After each epoch, we evaluate on all tasks' development sets, saving the model with the highest total score (sum of accuracies and adjusted STS correlation). Finally, this best model is tested on the test set, with predictions saved for submission.

5.3 Experimental details

We implemented the original BERT model using the "bert-base-uncased" model as per the default project instructions. We then enhanced it in the ways outlined in the Approach section. Each task has a dedicated classifier head: a linear layer mapping from BERT's 768-dimensional hidden state to 5 classes for sentiment, and to a single value for paraphrase detection and STS, with the latter two concatenating the sentence pair embeddings. For each iteration, we fine-tuned the entire BERT model, updating all parameters alongside the additional task-specific layers. We trained using the AdamW optimizer with a default learning rate of $1e-5$ along with a linear warmup and decay schedule, a batch size of 8 (chosen to fit within memory constraints) and 10 epochs. All of these hyperparameters can be set via command-line arguments to facilitate experimentation. These processes were executed separately on an NVIDIA T4 GPU. For reproducibility, we set a fixed random seed (11711).

6 Results and Analysis

We present our results on the development set for various models and scheduling strategies in the table below:

METHOD	**SST** (dev)	**Quora** (dev)	**STS** (dev)	**Score** (dev)
SST-ONLY BASELINE	0.406	0.369	-0.009	0.424
SEQUENTIAL BASELINE	0.396	0.368	0.268	0.466
(0 PALs) ROUND ROBIN	0.511	0.737	0.368	0.644
(0 PALs) ANNEALED	0.499	0.752	0.357	0.643
(0 PALs) DYNAMIC	0.519	0.714	0.362	0.638
(6 PALs) ROUND-ROBIN	0.463	0.737	0.377	0.630
(6 PALs) ANNEALED	0.423	0.743	0.357	0.615
(6 PALs) DYNAMIC	0.459	0.691	0.332	0.605
(3 PALs) DYNAMIC	0.500	0.708	0.322	0.623
(12 PALs) DYNAMIC	0.496	0.716	0.357	0.630
(0 PALs) ANNEALED, MB*	0.504	0.791	0.343	0.656
(6 PALs) ANNEALED, MB*	0.475	0.789	0.362	0.648
(0 PALs) ANNEALED, BOOT*, MB*	0.507	0.792	0.366	0.661
(0 PALs) ANNEALED, BOOT*, MB*, B16*	0.504	0.800	0.390	0.667

Table 1: Development set results for various models and scheduling strategies. *BOOT stands for Bootstrapping, MB stands for More Batches, and B16 stands for batch size 16. We detail what these mean in Section 6.3.

6.1 Isolating the Impacts of Task Scheduling Algorithms

To properly compare the impact of the schedulers and PALs, we performed ablation studies. First, we grouped the dev accuracy on each task by task scheduler, keeping the number of PALs to 0 to isolate the impact of schedulers. The results are summarized in the bar graph and heat maps in Figure 1.

Our baseline of having no real scheduler (sequential training) resulted in the lowest performance across all tasks, highlighting the inefficiencies and significant task interference that arise when tasks are learned in isolation. Despite sequential training eventually processing all training examples, the order of task presentation likely leads to overfitting on the first task, reducing the model's capacity to learn subsequent tasks effectively. This sequential overfitting hinders the model's generalization across tasks, emphasizing the necessity of interleaved training for effective multitask learning.

The round robin scheduler significantly improved performance compared to sequential training, achieving balanced learning across tasks. This method ensures equal training time for each task, providing a robust baseline despite not accounting for task difficulty or dataset size differences. Its balanced approach prevents any single task from dominating the training process.

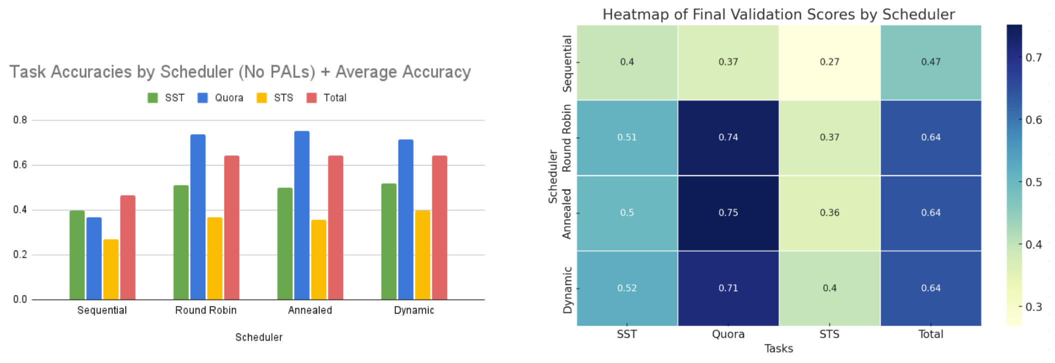


Figure 1: Ablation Study: Dev Accuracies by Task Scheduler.

Annealed sampling, which dynamically adjusts sampling probabilities based on resource availability, performed better on the paraphrase task (+0.015) but slightly worse on SST (-0.012) and STS (-0.011) compared to round robin. This result is likely due to the substantial size disparity between the datasets: as annealed sampling still incorporates a degree of proportionality to dataset size (mitigated slightly by adjusted sampling probabilities), the Quora dataset, being over 33 and 46 times larger than the SST and STS datasets respectively, dominates training. This dominance is less pronounced in round robin scheduling, where each task is sampled equally, preventing the larger dataset from overshadowing the smaller ones.

Dynamic task scheduling performed better on SST and STS, showing an increase of 0.008 and 0.03 from round robin, but these gains seem to come at the expense of the paraphrasing task, showing a decrease of 0.023. This outcome aligns with our expectations, as dynamic task scheduling prioritizes tasks based on their performance metrics. The relatively lower performance of SST and STS in our baselines indicates that these tasks are more challenging, thus receiving higher priority under the dynamic scheduler. This adaptive prioritization ensures that more difficult tasks receive greater attention, improving their performance while slightly reducing the focus on the easier paraphrasing task.

6.2 Isolating the Impacts of PALs

Next, we conducted another ablation study, this time to investigate the optimal number of PALs with the dynamic task scheduler, as this was a new task scheduling algorithm we brought to Stickland and Murray (2019)'s work.

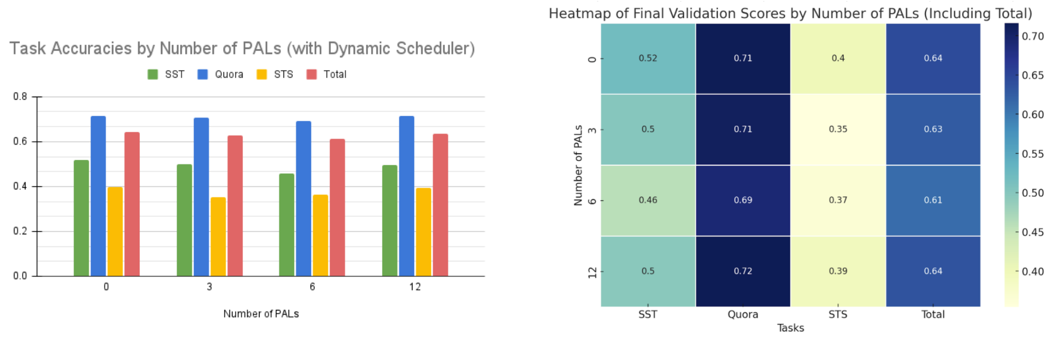


Figure 2: Ablation Study: Dev Accuracies by Number of PALs.

In this study, we observe two key findings that diverge from the original results presented by Stickland and Murray (2019). Firstly, our best performance was consistently achieved with zero PALs across all scheduling algorithms tested, as shown in Table 1.

There are several possible reasons for this discrepancy. The original paper utilized a broader range of tasks (8 out of 9 GLUE tasks), likely providing more opportunities for shared learning and reducing task interference through diverse task-specific signals. In contrast, our comparatively smaller task set of three may not have generated enough diversity in task-specific adaptations to justify the additional complexity introduced by PALs. Moreover, the smaller datasets in our study could have exacerbated overfitting due to the additional parameters introduced by PALs, thereby reducing overall performance. Implementation differences or variations in hyperparameter tuning may also account for this divergence.

Secondly, while Stickland and Murray (2019) suggested using 6 PAL layers as a heuristic for the original BERT architecture with 12 BERT layers, our findings indicate that enhancing all 12 layers with PALs yielded the best performance, followed by 3 PALs, and then 6 PALs. This suggests that the increased capacity for task-specific learning with 12 PALs outweighed the risks of overfitting, particularly in the context of our tasks and datasets. Conversely, having fewer PALs (3) provided some task-specific adaptations without introducing too many additional parameters, which helped prevent overfitting with smaller datasets.

These findings underscore that the optimal number of PALs is highly context-dependent. The authors of the original paper did not provide a definitive rationale for the number of PALs, suggesting it was based on empirical performance rather than a theoretical underpinning. Our results indicate that in our specific context, the balance between shared and task-specific learning benefits more from either no PALs or a larger number of PALs than the previously suggested heuristic of six. This discrepancy in our results highlights the need for further research to systematically explore the impact of PALs across different multitask learning scenarios, considering factors such as the number and diversity of tasks, dataset sizes, and the interplay between task scheduling strategies and task-specific adaptations.

6.3 Further Ameliorating Tasks' Resource Imbalance

The observed performance variations in our experiments highlighted the critical role of dataset size and resource availability across tasks. To address the imbalance and further enhance the performance of under-resourced tasks, we explored several optimization techniques.

METHOD	**SST** (dev)	**Quora** (dev)	**STS** (dev)	**Score** (dev)
(0 PALs) ANNEALED	0.499	0.752	0.357	0.643
(0 PALs) ANNEALED, MB*	0.504	0.791	0.343	0.656
(0 PALs) ANNEALED, BOOT*, MB*	0.507	0.792	0.366	0.661
(0 PALs) ANNEALED, BOOT*, MB*, B16*	0.504	0.800	0.390	0.667
(6 PALs) ANNEALED	0.423	0.743	0.357	0.615
(6 PALs) ANNEALED, MB*	0.475	0.789	0.362	0.648

Table 2: Subset of development set results showing impact of optimization techniques.

Firstly, we experimented with increasing the number of batches processed per iteration by eight times. This approach ensured more frequent updates for each task within an epoch, thereby providing additional learning opportunities for under-resourced tasks. This technique significantly improved performance both with and without the presence of PALs by maintaining a more consistent learning rate and reducing the risk of overfitting on smaller datasets. While this method is not directly targeted at balancing resource disparities among tasks, its general improvement in learning efficiency helped mitigate the overall impact of dataset imbalances.

To specifically target the under-resourced tasks, we employed bootstrapping to double the dataset size for the SST and STS tasks. This involved augmenting the existing data by generating additional samples through resampling with replacement. The increased dataset size provided more training instances, significantly improving the performance of both SST and STS tasks without negatively impacting the Quora dataset. This method effectively mitigated the data scarcity issue, allowing the model to better generalize and learn task-specific patterns for these under-resourced tasks. Additionally, we experimented with increasing the batch size from 8 to 16. This adjustment led to significantly improved performance for the paraphrase and STS tasks, albeit with a slight decrease in performance for SST. Larger batch sizes improve gradient estimation by averaging over more samples, leading to more stable and accurate updates to the model parameters. This stability is particularly beneficial for tasks with smaller datasets, as it helps in reducing the variance in gradient estimates and improves the overall learning process. The slight decrease in SST performance suggests that while larger batch sizes offer general benefits, the specific dynamics of the SST task may require further fine-tuning to optimize batch size effects.

Finally, we increased the number of training epochs to 40, but the improvements observed were minimal. This suggests that while extending training duration can often sometimes enhance model performance, task resource imbalances are the main hindrance to performance in the specific context of MTL.

6.4 Test Set Results

Task	**Test Accuracy/Correlation**	**Change**
SST	0.532	+0.016
Paraphrase (Quora)	0.800	+0.136
STS	0.316	+0.048
Overall Score	**0.663**	**+0.197**

Table 3: Test set results with accuracy, correlation, and change compared to the sequential baseline. The overall score is an aggregate metric of the individual tasks.

7 Conclusion

In this study, we developed a BERT-based multitask classifier, incorporating Projected Attention Layers (PALs) and three task scheduling algorithms: round robin, annealed sampling, and dynamic task scheduling. We conducted extensive experiments to determine the most effective strategies for multitask learning and employed additional optimization techniques to mitigate trends we were seeing in our results, namely resource imbalance and domination by one task.

Our experiments revealed that while task scheduling algorithms like dynamic scheduling and annealed sampling significantly improve multitask learning performance, their interaction with PALs needs careful management. The additional complexity introduced by PALs necessitates a stable and consistent training process, which may not align well with the adaptive nature of advanced task scheduling algorithms. This was evident in our results, where configurations without PALs consistently outperformed those with PALs, highlighting the potential for inefficiencies when these elements are not optimally integrated.

Key insights from our study include the critical importance of addressing dataset size disparities in multitask learning scenarios. Targeted optimization strategies, such as bootstrapping and increasing batch sizes, proved effective in mitigating the challenges posed by resource imbalances. These techniques enhanced the performance of under-resourced tasks, demonstrating the value of tailored interventions in achieving balanced multitask learning.

To better harness the potential of PALs in conjunction with task scheduling, future research should explore more sophisticated integration strategies that harmonize task-specific adaptations with dynamic learning priorities. Potential approaches could involve developing hybrid methods that dynamically adjust the number of active PALs based on real-time task performance or incorporating mechanisms to ensure more synchronized learning signals across tasks. Additionally, investigating the impact of different hyperparameter settings and more diverse task sets could provide further insights into optimizing MTL frameworks.

8 Contributions and Acknowledgments

Both authors worked together on all aspects of the paper: finding the topic, literature review, co-writing the codebase, running the models, data analysis, and writing the paper.

9 Ethics Statement

Given what we know about MTL and the transfer of knowledge across tasks, we must consider the potential for bias in not just one dataset, but all of them. A model can learn biases from one task and propagate them across all tasks, leading to biased or prejudiced predictions in applications where bias can have significant negative impacts. For example, while bias in a simple sentiment classification task might have limited harm, bias in a news paraphrase application could have far-reaching negative consequences, such as the dissemination of misinformation or unfair representation of certain groups. To mitigate this risk, we need to carefully curate our datasets, remaining critical of what we choose to use as benchmarks. Each use of a dataset in academic research makes it more credible, so it is crucial to scrutinize and actively intervene to mitigate these biases. This involves implementing bias detection techniques, employing debiasing strategies, and continuously evaluating the fairness of our models.

Another ethical concern is the overfitting of our model to dominant tasks, which can result in the neglect of less represented but equally important tasks, such as those involving minority languages or marginalized communities. This issue can further entrench existing inequalities in technology access and efficacy. In our investigation, we observed the Quora dataset overpowering the smaller SST and STS datasets despite our efforts to balance them. This imbalance can lead to a model that performs well on tasks with abundant data but poorly on tasks with less representation, exacerbating disparities in technology effectiveness. We found in our study that it is possible to combat these discrepancies to some extent using techniques like dynamic task scheduling and bootstrapping, preventing the model from overfitting to dominant tasks and promoting a more balanced performance across all tasks.

The complexity of multitask learning models, especially when incorporating advanced techniques like PALs and dynamic task scheduling, can make them difficult to interpret. A lack of transparency in model decisions can hinder trust and accountability, particularly when these models are used in high-stakes applications such as medical diagnoses. In our research, we conducted multiple ablation studies to isolate the impacts of different factors, highlighting the challenges in understanding and explaining model behavior. Techniques such as attention visualization, probe studies, ablation studies, and comprehensive documentation of model architecture and training processes can enhance transparency. By providing clear insights into how the model makes decisions, we can build trust and ensure accountability in its applications.

References

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

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. Dynamic task prioritization for multitask learning. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI*, page 282–299, Berlin, Heidelberg. Springer-Verlag.

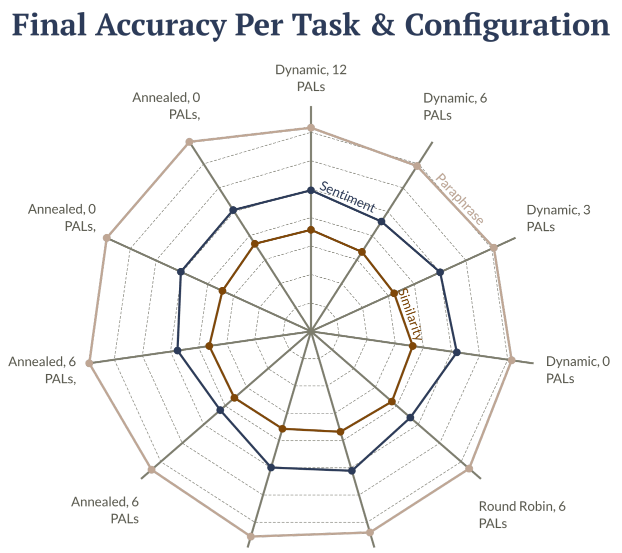
Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning. volume abs/1902.02671, Online. CoRR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

A Appendix (optional)

A.1 Radar Graph of Final Accuracies

Below is a cool radar graph illustrating the final accuracies per task and configuration. This visual representation helps compare the performance across different models and scheduling strategies.



A.2 GitHub Repository

All code and resources for this project are available in our GitHub Repository. The repository includes the implementation of PALs, task scheduling algorithms, and data preprocessing scripts.