SlapBERT: Shared Layers and Projected Attention For Enhancing Multitask Learning with minBERT

Stanford CS224N Default Project

Deven Pandya, Allison Jia, and Alex Zhai
Department of Computer Science
Stanford University
devenp14@stanford.edu, ajia@stanford.edu, alexzhai@stanford.edu

Abstract

In this project, we implement multi-task learning for sentiment analysis, paraphrase detection, and similarity assessment by building a model upon the minBERT framework. By employing architectural improvements and training techniques, including dynamic loss adjustment, learning rate scheduling, SMART regularization, Projected Attention Layers (PALs), annealed sampling, and gradient surgery, our model demonstrates substantial improvements over the baseline. Furthermore, with an original shared transformer encoder layer, strategic freezing of BERT layers, and annealed sampling, we achieve a maximum development accuracy of 0.694, significantly surpassing our baseline methods.

1 Key Information to include

Our mentor is Josh Singh. We have no external collaborators nor are we sharing projects.

2 Introduction

In this project, we tackle the problem of multi-task learning in Natural Language Processing (NLP), which poses a significant challenge due to the complexities inherent in the English language, including diverse syntactic structures, pragmatic contexts, and semantic understanding. Each of the three tasks we aim to tackle — sentiment analysis, paraphrase detection, and similarity assessment — present challenges that require our model to capture different aspects of language learning and understanding.

Traditional single-model training for multiple tasks relies on transfer learning from large pre-trained or task-specific models, requiring extensive fine-tuning that can result in inefficiency and redundancy.

The current state of the art in multi-task learning (MTL) is characterized by innovative approaches that leverage shared representations and task-specific adaptations to improve performance across multiple tasks. Notable contributions include the use of transformer-based models for MTL in NLP, as exemplified by Liu et al. in "Multi-Task Deep Neural Networks for Natural Language Understanding" (2019), which demonstrates the effectiveness of shared architectures. Liu et al. (2019) Additionally, the concept of dynamically balancing task learning, as proposed in Kendall, Gal, and Cipolla's "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics" (2018), showcases advanced techniques in adaptive learning for managing task interference and prioritization effectively. Kendall et al. (2017) We seek to enhance minBERT through a systematic exploration of strategies:

- **Multitask Classification Expansion**: Enhance the model to concurrently train on sentiment analysis, paraphrase detection, and similarity assessment, employing round-robin multi-task training and task-specific loss functions for better generalization.
- **Dynamic Loss Adjustments**: Implement a dynamic loss adjustment strategy to balance each tasks' impact to maximize overall performance.

- **Learning Rate Scheduling and Transformer Encoder**: Apply learning rate scheduling to address plateauing loss and incorporate a transformer encoder layer to boost similarity task performance.
- **Interleaved Batches and Gradient Surgery**: Adopt interleaved batch training and gradient surgery to enhance generalization and address gradient conflicts.
- **Shared Transformer Layer**: Use a shared transformer layer with interleaved batches and gradient surgery for regularization and cross-task learning, marking our most **original** innovation.
- **SMART Loss Integration**: Include SMART loss for regularization to curb overfitting.
- **Projected Attention Layers (PALs) and Annealed Sampling**: Implement PALs to reduce interference in task-specific fine-tuning and annealed sampling for strategic task selection during training.
- **Layer Freezing**: Freeze most layers, except those near the output during fine-tuning, to preserve the minBERT model's generalized capabilities.
- **Combined Pretraining and Multi-task Training**: Initially pre-train the model on individual tasks and then proceed with multi-task learning to minimize task-specific interference.

3 Related Work

Our minBERT model incorporates a transformer layer with multi-headed attention to capture data dependencies, enhanced by "Add & Norm" layers for output processing and stabilization. Ashish Vaswani and Polosukhin. (2017) We build off of this baseline model to increase the robustness of our embeddings, fine-tuning BERT through multi-task learning.

In "MTRec: Multi-Task Learning Over BERT for News Recommendation," Bi and Li apply multi-task learning to refine BERT embeddings for improved news recommendation Bi et al. (2022), using Gradient Surgery to resolve learning conflicts across various tasks like recommendation, classification, and recognition, enhancing BERT's task-specific performance. Yu et al. (2020)

Expanding upon the concepts of multi-task learning as explored by Bi and Li, Stickland and Murray (2019) delve into the challenge of over-fitting within multi-task learning frameworks. Stickland and Murray (2019) Their research proposes the introduction of Projected Attention Layers (PALs) alongside task-specific output layers. PALs act as adaptive elements within the shared BERT architecture, enhancing the generation of task-specific embeddings without compromising the shared learning infrastructure crucial for general applicability across tasks. Furthermore, Stickland and Murray introduce annealed sampling as a method to manage the imbalance of data volumes across different tasks, promoting more balanced exposure to each task over successive training epochs. These innovations inform the development of our multi-task learning framework, showcasing their potential to refine and improve the flexibility and efficacy of multi-task learning models.

We also explore SMART Loss proposed by Jiang and He, a smoothness-inducing regularization, as an alternative to reduce overfitting in multi-task finetuning of pre-trained models. Jiang et al. (2019) Our paper's regularization strategies, including SMART Loss, are largely drawn from their methodology.

4 Approach

4.1 Baselines

As a foundational baseline, we establish **RandBERT**, a randomized model that generates arbitrary outputs for sentiment classification, paraphrase detection, and textual similarity (see Table 1). This baseline serves as a reference point for the performance of subsequent architectures across the three tasks against a method that formulates predictions without leveraging learned data features.

To facilitate multi-task learning across sentiment classification, paraphrase detection, and semantic textual similarity tasks, we introduce **BaseBERT**. The pretrained BaseBERT serves as an additional baseline because it is our simplest architecture, consisting of a dropout layer and a linear layer, appplied to all three tasks. Furthermore, it employs a round-robin technique for task iteration to mitigate model overfitting to any task. In the context of this project, we categorize an architecture as pretrained if it does not modify the BERT layers while training.

For our baseline and all subsequent architectures, we consistently employ the same designated loss function for each specific task. In particular, for sentiment classification, we apply CrossEntropy loss due to its effectiveness for categorical tasks where embeddings are classified into one of five sentiment categories. For paraphrase detection, we select BCEWithLogitsLoss, recognizing its suitability for binary classification tasks. For the evaluation of semantic textual similarity, which involves outputting a score on a continuous scale from 0 to 5, we utilize MSELoss. This choice recognizes the continuous nature of similarity and allows the model to perceive subtle distinctions beyond discrete categories. The total loss is calculated as the sum of the individual losses for each task.

4.2 Dynamically Adjusted Loss

As a first attempt to improve BaseBERT, we implement dynamically adjusted loss (DAL) with the goal of learning how to weight tasks in a manner that contributes to the best overall performance of the model. To do so, we implement the following loss adjustment, taking inspiration from Kendall and Gal's homoscedastic task uncertainty from their paper "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". Kendall et al. (2017)

$$\ell_t := e^{-\sigma_t}\ell + \sigma_t \quad . \tag{1}$$

Here, we set the task-specific loss (ℓ_t) to be a scaled version of the originally calculated loss via a transformation that depends upon a task specific learnable parameter σ_t.

4.3 Learning Rate Scheduling and Similarity Transformer Layer

While DAL significantly improves our dev accuracy across all three tasks, we encounter major overfitting problems when adding DAL upon BaseBERT. To account for this, we implement learning rate scheduling which gradually reduces the learning rate when performance plateaus for several epochs. In order to improve similarity, our worst-performing task, we add an **original** transformer encoder layer that attends to minBERT embeddings of the two input sentences in order to capture semantically relevant information. The output of this transformer encoder layer is then fed through the same linear layer with dropout as described in BaseBERT.

4.4 Shared Transformer Encoder Layer

As we expect, the additional shared layer leads to improvements in similarity performance, but with decreased paraphrase and sentiment accuracy. Further, the model still overfits, so we use the similarity task's shared transformer encoder layer across all three tasks to regularize and facilitate cross-task learning. Our new model, **ShareBERT**, stacks paraphrase and similarity embeddings, processes them through the shared layer (with size of $d = 768$), and feeds resultant embeddings to respective task specific linear layers. This approach enables ShareBERT to learn broadly applicable word embeddings for all tasks by using a common attention mechanism. We also implement interleaved batches within our round-robin approach, so that our model cycles through one batch per task instead of iterating through entire datasets at a time. The addition of the shared transformer encoder layer represents our most **novel** contribution and results in our best performance.

4.5 Gradient Surgery

Given the increased frequency in cycling between tasks, we also introduce gradient surgery to reconcile conflicting gradients between different tasks. Yu et al. (2020) Gradient surgery projects the gradient of an i-th task g_i onto another conflicting task gradient g_j's normal plane as follows

$$g_i = g_i - \frac{g_i \cdot g_j}{||g_j||^2} \cdot g_j. \tag{2}$$

In order to implement this, we refer to Tseng's PCGrad implementation. Tseng (2020)

4.6 SMART Loss

We find that ShareBERT leads to a significant jump in performance for paraphrase detection and similarity prediction, but overfitting persists even with the introduction of a shared layer. To address this, we introduce Smoothness Inducing Adversarial Regularization (SMART) as proposed by Jiang and He. Jiang et al. (2019) SMART attempts to minimize the KL divergence between P, the probability distribution of the model's predictions on the original input, and Q, the probability distribution of the model's predictions on a slightly perturbed input. By implementing SMART Loss ($\ell_s(P, Q)$), scaled by a tuned regularization weight, to each of our task losses, we attempt to further regularize our model by making its output consistent and independent of small perturbations in input. We use Jiang's SMART implementation to do this. Jiang et al. (2020)

$$l_s(P, Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P). \tag{3}$$

4.7 Projected Attention Layers

In order to reduce negative interference between tasks in multi-task learning, we add task specific Projected Attention Layers (PALs) as defined by Stickland and Murray on top of our ShareBERT architecture to arrive at our final architecture, **SlapBERT**. Stickland and Murray (2019) (See Figure 6). Denoting the BERT Layer as BL, the self-attention layer as SA, our down projection transform as V_D, our up projection transform as V_E, our layer norm as LN, and our two feed forward layers with GELU as FFN, we calculate the new output of the BERT layer as follows

$$PA(\mathbf{h}) = V_D(SA(V_E(\mathbf{h}))) \tag{4}$$

$$BL(\mathbf{h}) = LN(\mathbf{h}_{att} + FFN(\mathbf{h}_{att}) + PA(\mathbf{h})) \tag{5}$$

In SlapBERT, we implement 12 fine-tuned, task-specific PAL Layers that run in parallel to 12 BERT layers to develop specialized, task-dependent knowledge from shared representations. (See Figure 1)

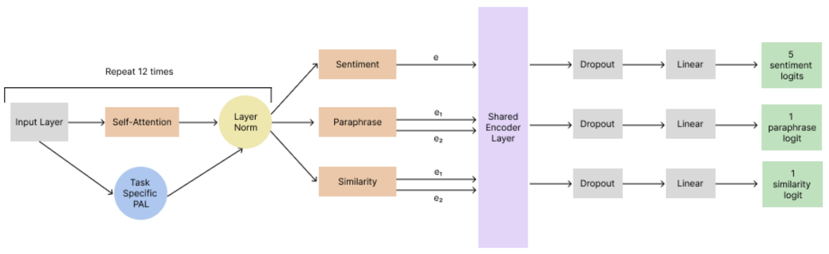

Figure 1: SlapBERT Architecture

4.8 Annealed Sampling

Noting the discrepancy in data between QQP and the other datasets along with our significantly higher performance on QQP, we also introduce annealed sampling which cleverly adjusts the sampling rate of tasks to account for dataset size differences in order to gradually balance training over time and reduce overfitting. Stickland and Murray (2019) Where N_i is the number of data in the i_{th} task and p_i is the probability of sampling the i_{th} task, we sample tasks according to the probability distribution:

$$p_i \propto N_i^{\alpha} \text{ where } \alpha = 1 - 0.8\frac{e-1}{E-1}. \tag{6}$$

Here e is the current epoch and E is the total number of epochs.

4.9 Layer Freezing

While we find that SlapBERT performs comparably to ShareBERT, both models overfit, so we additionally propose freezing the first two layers of BERT to retain some generalizability from the pretrained BERT and learn only the necessary task-specific information in layers closest to the output.

4.10 Individual Pretraining

To mitigate interference and preferentially localize learning within task-specific layers, akin to the strategy of freezing certain BERT layers, we adopt an approach of initial single-task pretraining for

SlapBERT. This step is designed to stabilize the weights of the task-specific PALs prior to multi-task training. This stabilization is expected to strengthen the foundation upon which multi-task learning builds, thereby enhancing overall performance and reducing negative transfer between tasks.

4.11 Other Similarity Prediction Modifications

In addition to the previous methods, we explore architectures with original implementations of both Multiple Negative Ranking Loss Learning and Cosine Similarity fine-tuning for improving similarity prediction, both of which Henderson and Al-Rfou explore in depth. Henderson et al. (2017)

5 Experiments

5.1 Data

For sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset which contains 11,855 sentences from movie reviews. Socher et al. (2013) As a part of this task, our minBERT embeddings are used to predict sentiment classification labels of negative, somewhat negative, neutral, somewhat positive, or positive for movie reviews. We then use the Quora Dataset (QQP) with 400,000 question pairs for paraphrase detection. Shankar Iyer and Csernai (2017) For this task, our minBERT embeddings are used to determine whether words or pairs of phrases convey the same semantic meaning. Finally, we use the SemEval STS Benchmark Dataset with 8,628 sentence pairs for semantic textual similarity. Agirre et al. (2013) For this task, our minBERT embeddings are used to determine how semantically similar two phrases are on a scale of 0 to 5 where 0 signifies no semantic relation. See table 5 in our Appendix for example data from each dataset.

5.2 Evaluation Method

Accuracy metrics are used for sentiment analysis and paraphrase detection due to their discrete outputs, while the Pearson correlation coefficient evaluates semantic textual similarity tasks. To assess overall performance, we adopt a unified metric that averages the normalized Pearson score with accuracy scores from sentiment analysis and paraphrase detection tasks to evaluate the model's overall multi-task performance. The development accuracies for these metrics are detailed below.

5.3 Experimental Details

We originally begin training our models with learning rates of 1×10^{-3} and 1×10^{-4}, but after implementing learning rate scheduling, we start each model experiment with a learning rate of 1×10^{-5}. Our models are trained with batch size 8 for 10 epochs. All of our models use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-7}$, and weight decay of 1×10^{-4}.

We run **BERT** with the default configuration outlined by Devlin et al. (2019), with a hidden layer size of $d_h = 768$. There are a total of 12 BERT layers and 12 attention heads per layer. We use **SMART** as detailed in Jiang et al. (2019) with $\tilde{x} = 1$, $\epsilon = 1 \times 10^{-6}$, $\sigma = 1 \times 10^{-5}$, and $\eta = 1 \times 10^{-3}$. We run **PALs** with the default configuration in Stickland and Murray (2019) with PALs attention layers of size $d_s = 204$ and 12 attention heads for each layer.

5.4 Results

Table 1: Baseline Performance

METHOD	SST	QQP	STS	Overall
BaseBERT (Pretrained Baseline)	0.363	0.652	0.190	**0.537**
RandBERT (Randomized Multi-task Classification)	0.198	0.375	-0.013	**0.356**
SlapBERT (Individual Pre-train)	0.208	0.625	0.446	**0.519**

Our initial baseline performance outcomes provide benchmarks for subsequent modifications, with accuracies reported for SST and QQP and Pearson Correlation for STS. Table 1 showcases the efficacy of our models: BaseBERT, which feeds the pretrained embeddings through linear layers with dropout using round-robin sampling; SlapBERT, which undergoes sequential individual pre-training for each task; and RandBERT. For the individual pretraining on each task using SlapBERT, the non PALs BERT layers are frozen for each task except similarity prediction.

Table 2: Initial Attempt at Multi-task Learning

	SST	**QQP**	**STS**	**Overall**
Dev Accuracy	0.379	0.663	0.321	**0.568**

From our analysis, it is evident that the BaseBERT model, even without fine-tuning, outperforms RandBERT, which generates random predictions without utilizing learned data features. This outcome suggests that BaseBERT successfully captures pertinent data representations across tasks. However, it is noteworthy that SlapBERT, despite being our most sophisticated architecture, falls short of the performance of our pretrained BaseBERT. This discrepancy underscores the efficacy of round-robin sampling as opposed to sequential task-specific training, which informs our decision to fine-tune minBERT using the BaseBERT architecture with round-robin sampling.

Table 3: The Effects of DAL, Gradient Surgery, Shared Layers, and Layer Freezing on Performance

	Setup 1		**Setup 2**		**Setup 3**	
	Dev	Train	Dev	Train	Dev	Train
SST Accuracy	0.511	0.984	0.484	0.923	0.510	0.689
QQP Accuracy	0.767	0.990	0.820	0.986	0.771	0.782
STS Correlation	0.337	0.873	0.553	0.825	0.545	0.773
Overall Accuracy	**0.649**	0.970	**0.694**	0.941	**0.684**	0.786

Setup 1: BaseBERT w/ Dynamically Adjusted Loss
Setup 2: ShareBERT w/ Interleaved Batches and Gradient Surgery
Setup 3: ShareBERT w/ Gradient Surgery, Dynamically Adjusted Loss, & 10 Unfrozen Layers

After fine-tuning BERT in our initial multi-task learning approach, we notice a jump in performance across all the tasks, especially similarity, as shown in table 2. Our models, across three configurations outlined in table 3, display varying degrees of effectiveness. Setup 1 augments BaseBERT with dynamically adjusted loss (DAL), boosting sentiment and paraphrase task performance but inducing overfitting, possibly due to DAL's propensity to underprioritize complex tasks. Setup 2's ShareBERT integrates interleaved batches and gradient surgery and delivers the best overall results, with marked gains in paraphrase and textual similarity tasks. However, overfitting remains a challenge. In response, Setup 3 reintroduces DAL and freezes two BERT layers in ShareBERT, curbing overfitting but at the expense of task-specific performance. This suggests that the learning terrain is sufficiently complex such that enhancing generalizability is challenging to do without overfitting and implies that transfer learning between these three tasks is difficult.

Table 4: BaseBERT, ShareBERT, and SlapBERT w/ SMART Loss and Annealed Sampling

METHOD	**SST**	**QQP**	**STS**	**Overall**
SMART Loss, Interleaved Batches				
ShareBERT (Dropout 0.5)	0.463	0.818	0.558	0.687
SlapBERT (Dropout 0.5)	0.507	0.779	0.342	0.652
SMART Loss, Annealed Sampling				
BaseBERT	0.522	0.764	0.359	**0.655**
SlapBERT	0.523	0.823	0.440	**0.689**
SlapBERT w/ Cosine Similarity	0.503	0.752	**0.545**	0.676
SlapBERT w/ MNRL	0.474	0.724	0.029	0.571

Table 4 compares the performance of our models with SMART loss and annealed sampling. We observe that SlapBERT performs better than BaseBERT and performs comparably to ShareBERT while still reducing overfitting. We found that implementing SMART loss does not significantly reduce overfitting (See Table 6). However, SMART loss in conjunction with annealed sampling improves the performance of our BaseBERT and SlapBERT models while slightly reducing overfitting. Cosine Similarity with SlapBERT produces the highest dev performance for the similarity task but at the expense of overall performance.

We experiment with different dropout probabilities, learning rates, weight decay weights, regularization weights, and number of frozen layers; however, we only present on our most significant findings and omit the other results for brevity. Our highest performing architecture achieves a final test performance of 0.695, with a breakdown of 0.505 for SST, 0.819 for QQP, and 0.521 for STS.

6 Analysis

To analyze our model, we reference the t-SNE method for embedding visualization from van der Maaten and Hinton (2008) on our own sentence embeddings, projecting them onto three dimensions. With the following visualizations, we can identify how the embeddings separate or cluster.

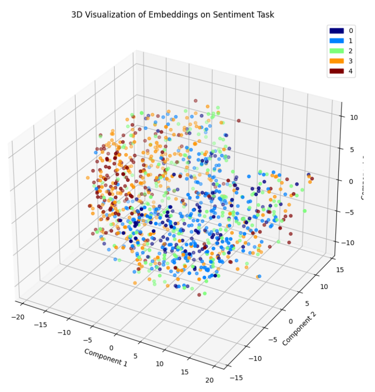

Figure 2: Sentiment Labels

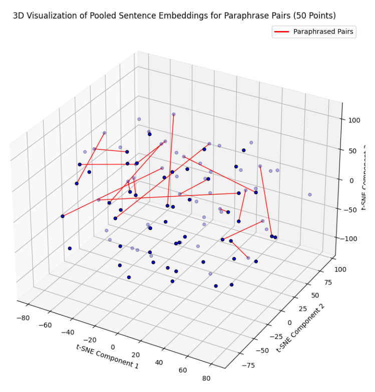

Figure 3: Paraphrase Pairs

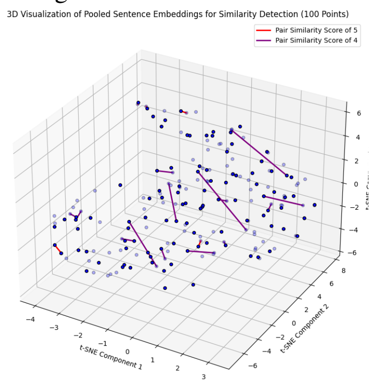

Figure 4: Pairs with Similarity of 4 & 5

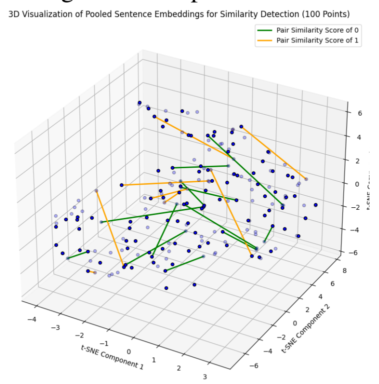

Figure 5: Pairs with Similarity of 0 & 1

In Figure 2, we visualize 1,101 embeddings, distinguishing their sentiment labels by color. We see that the most extreme sentiments — either negative or positive sentences with scores of 0 or 4 — cluster together with each other, with the dark blue and red clusters most distinct. On the other hand, embeddings for more neutral sentiments are more difficult to parse and therefore scatter more evenly throughout the embeddings, as evidenced by the green nodes (for sentiment label 2) that are almost perfectly spread throughout the figure. From this visualization, it becomes apparent that the model struggles to precisely differentiate between mildly positive sentences and those that are clearly positive, as well as between mildly negative sentences and unequivocally negative ones. This is depicted through the diffuse clustering of light blue nodes (1) in proximity to the dark blue nodes (0), with the former being more loosely distributed. A similar pattern is observed with the orange nodes (3), which gather near the red nodes (4) but scatter more broadly.

To visualize our model's paraphrase performance, we create pairs by connecting paraphrased embeddings. In Figure 3, connections between paraphrases are significantly shorter than connections between non-paraphrase pairs (see Figure 7 in the Appendix), which highlights our model's ability to identify semantically similar sentences and cluster them closely in embedding space. We observe that there is still variety in the spread between two ends of a paraphrase connection, even though we perform the best on the paraphrase task. This could be due to the fact that there may be many similar elements between a phrase and its paraphrase, but the two could still be syntactically different. Since paraphrases can include wide ranges of vocabularies and structures, the allowed variance between paraphrases is greater than the allowed variance in similarity scores.

Next, to visualize our model's similarity performance, we connect embeddings for sentence pairs with high similarity scores (4 & 5) in Figure 4 and pairs with low scores (0 & 1) in Figure 5. In comparing the two, we can see that sentence pairs with high similarity scores are closer together compared to the sentence pairs with low scores. This reflects the model's ability to not only effectively capture semantic similarity but also distinguish between sentences with little or no semantic similarity.

To further support our analysis, we look at mismatches between our model's predictions and the actual labels of different dev task data. By far, most of the sentiment classification mismatches fall within 1 value of the true label, suggesting that our model generally predicts within the vicinity of the correct sentiment but struggles to learn the decision boundary between adjacent sentiment classes. A more extreme mismatch example is evidenced by the model incorrectly classifying the input "I have no way of knowing exactly how much is exaggeration, but I've got a creepy feeling that the film is closer to the mark than I want to believe" as negative (4) instead of neutral (2). We hypothesize that our model fixates on keywords such as "creepy" that are often associated with negative sentiments and fails to learn the phrase's broader context that deems it neutral.

Similarly, we hypothesize that our model fixates on the presence of certain keywords to predict paraphrase detection. For example, our model incorrectly predicts the inputs "How does rain affect flights and flying?" and "How do planes fly when it is raining?" as a paraphrase pair, perhaps due to the presence of words such as "how", "do", "rain" and "fly" in both inputs.

Finally, in textual similarity, our model incorrectly assigns "The bulk of India then was not controlled by Porus, but by the Nanda dynasty, centered at Pataliputra" and "I believe Alexander lost heart after his horse, Bucephelus, died from a wound received in the battle against Porus" with a similarity score of 3.5 instead of the true label of 0.6. Again, we theorize the model is focusing on the unique keyword "Porus" in both inputs, since it may be uncommon in the corpus and therefore weighted more heavily in the similarity task. This suggests that our model has not learned enough generalizable parameters to capture the semantic meaning of the text as a whole. Instead, it appears to be overly reliant on specific keywords or phrases, which can be indicative of a superficial understanding that does not fully grasp the nuanced interplay of words in context.

7 Conclusion

In our project, we build upon shared representation and task-specific adaptation techniques, central to cutting-edge multi-task learning research. We incorporate task-specific Projected Attention Layers (PALs), annealed sampling for intelligent task selection, and a shared transformer encoder layer to mitigate inter-task interference, leading to the development of two novel architectures: ShareBERT and SlapBERT. These architectures surpass our established baselines and underscore the sophisticated strategies required to enable minBERT to effectively generalize across tasks.

Despite this progress, our top-performing models fall short of surpassing state-of-the-art multi-task learning systems. Aggressive regularization methods, such as layer freezing and task-specific pretraining, introduce complexities that impede the effective transfer of learning between tasks. This is particularly evident in tasks requiring fine-grained discrimination, such as differentiating "somewhat positive" from "positive" sentiment classifications or calculating precise continuous similarity scores, which become challenging when cross-task learning interference is present.

Future work should investigate methods that balance the reduced interference of task-specific learning with maintaining the generalization advantages of shared layers. As the discipline evolves, it is crucial to conduct research with mindful consideration of ethical aspects, such as the ramifications of data representation and potential biases within models. It is vital to ensure that multi-task learning models are developed and applied in ways that are fair, transparent, and inclusive, facilitating their responsible adoption in diverse real-world settings.

8 Team Contributions

We split the work of writing the proposal, milestone, and report evenly. The code was also split evenly. Allison explored MNRL, Cosine Similarity, and developing baselines. Deven worked on the shared transformer encoder layer, PALs, and annealed sampling. Alex developed gradient surgery, interleaved batches, and layer freezing.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. Sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (*SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Łukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998—6008, Online. Association for Computational Linguistics.

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Mtrec: Multi-task learning over bert for news recommendation. In *Findings*.

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

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *ACL*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Nikhil Dandekar Shankar Iyer and Kornél Csernai. 2017. First quora dataset release: Question pairs. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Accessed: 02/29/24.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the International Conference on Machine Learning (or the correct conference name)*, page the range of pages where the paper appears.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

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

A Appendix

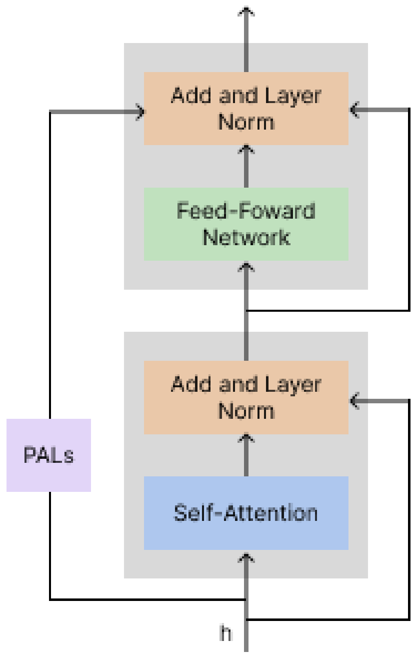

Figure 6: PALs Architecture

Table 5: Sample Data from Datasets

Dataset	Input 1	Input 2	Label
SST	"This film fails to deliver on its promising premise"	N/A	1 (Somewhat Neg.)
QQP	"I am a Capricorn Sun Cap moon and cap rising...what does that say about me?"	"I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?"	1.0
STS	"The bird is bathing in the sink"	"Birdie is washing itself in the water basin"	5

For MNRL, we used the equation provided in the default project spec:

$$\begin{aligned} \mathcal{J}(x,y,\theta) &= -\frac{1}{K}\sum_{i=1}^{K}\log P_{\text{approx}}(y_i|x_i) \\ &= -\frac{1}{K}\sum_{i=1}^{K}\left[S(x_i,y_i) - \log\sum_{j=1}^{K}e^{S(x_i,y_j)}\right] \end{aligned} \tag{7}$$

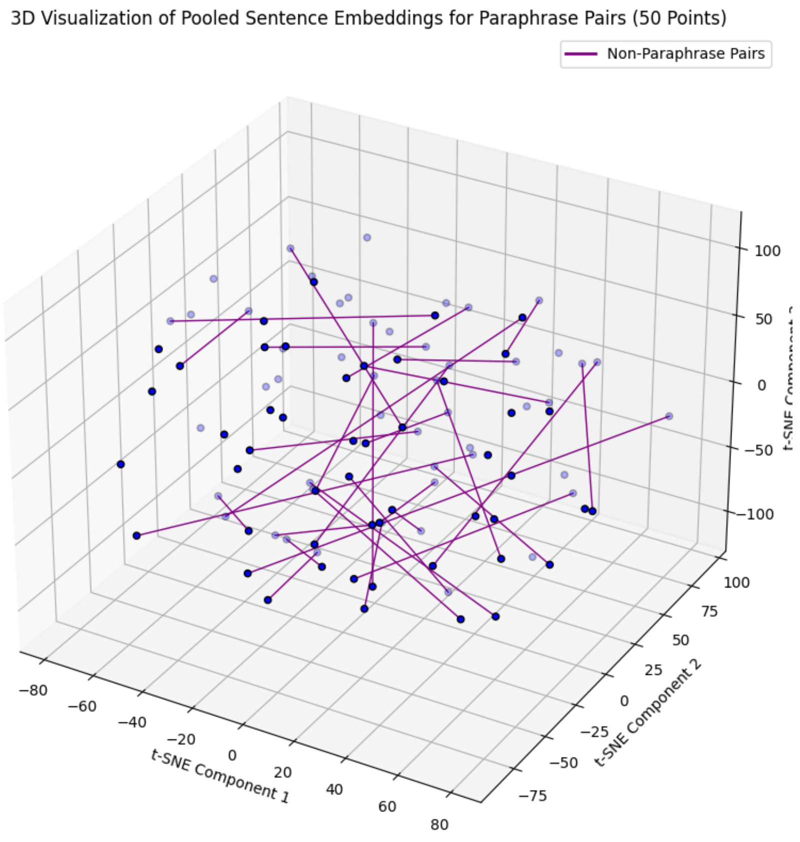

Figure 7: Non-Paraphrase Pairs

Table 6: TRAIN accuracies

METHOD	**SST**	**QQP**	**STS**	**Overall**
SMART Loss, Interleaved Batches				
ShareBERT (Dropout 0.5)	0.900	0.995	0.864	0.973
SlapBERT (Dropout 0.5)	0.975	0.995	0.852	0.965
SMART Loss, Annealed Sampling				
BaseBERT	0.978	0.855	0.857	0.920
SlapBERT	0.956	0.842	0.845	0.907
SlapBERT w/ Cosine Similarity	0.601	0.760	0.789	0.752
SlapBERT w/ MNRL	0.524	0.731	0.085	0.599