An Exploration of Multi-Task Learning over minBERT

Stanford CS224N Default Project (TA mentor: Josh Singh)

Chunming Peng
cmpeng@stanford.edu

Max Yuan
max9@stanford.edu

Annie Wang
wangj00@stanford.edu

Abstract

This project aims to enhance multi-task performance over minBERT in Sentiment Analysis (SST), Paraphrase Detection (Para), and Semantic Textual Similarity (STS). We explore various strategies: hyperparameter optimization, dataset augmentation, loss function enhancements, contrastive loss, and multiple negatives ranking loss. We also integrate LoRA, and additional attention head layers, seeking to improve minBERT's effectiveness across diverse NLP tasks. Through rigorous experimentation, we demonstrate that our approach significantly improves model generalization and robustness, resulting in notable performance enhancements across all three tasks concurrently. Our method achieves the top 15 on the leaderboard, underscoring its effectiveness and competitiveness in real-world applications.

1 Introduction

The objective of the project is to boost BERT's effectiveness with multi-task learning approaches, and to develop an efficient and stable version of miniBERT. This refined version will be tailored specifically for improving sentiment analysis, paraphrase detection, and semantic textual similarity tasks. We aim to achieve this by exploring six strategies: hyperparameter optimization, dataset augmentation using Llama3, loss function enhancements, contrastive loss, multiple negatives ranking loss, integration with LoRA and additional attention head layers. Furthermore, we conduct comprehensive performance evaluations, comparing our methodology against existing baselines. The key challenge lies in avoiding interference or negative transfer between multiple tasks and at the same time enhancing their overall accuracy and performance, since multi task learner tends to interfere with each other, and improving the performance with one task might degrade that of another.

2 Related Work

The recent advancements in natural language understanding tasks have seen BERT emerge as a leading model, showcasing state-of-the-art performance. However, there remains room for improvement, particularly in text classification tasks. Sun et al. (2019) presents a novel approach to fine-tuning BERT, encompassing three crucial stages: initial pre-training on task-specific or domain-specific data, optional integration of multi-task learning for related tasks, and subsequent fine-tuning tailored to the target task. Additionally, it investigates diverse fine-tuning methodologies for BERT, including strategies for preprocessing lengthy text inputs, identifying optimal layers for text classification, mitigating the catastrophic forgetting phenomenon, and implementing layer-wise learning rate adjustments, as demonstrated in 1 (Sun et al. (2019), Howard and Ruder (2018)). Furthermore, the paper explores scenarios of low-shot learning, wherein the model adapts to limited training data, thus contributing to a comprehensive understanding of BERT's applicability and performance enhancement strategies in various contexts. For extensive literature review, please find the "Related Work" Section in the Appendix.

$$\theta_l^t = \theta_l^{t-1} - \eta^l \cdot \nabla_{\theta_l} J(\theta) \tag{1}$$

3 Approach

Baseline minBert Model The project employed multi-task learning to simultaneously train sentiment analysis, paraphrase detection, and semantic textual similarity tasks. We utilized a joint loss function with linear weights for each task's loss term. For sentiment analysis, minBERT Jiang et al. (2020)'s pooler output, followed by a dropout layer and linear layer, optimized cross-entropy loss. Paraphrase detection involved minBERT processing sentence pairs, concatenating embeddings, and applying a linear layer to optimize binary cross-entropy loss. Semantic textual similarity utilized mean pooling of minBERT's last hidden state output, cosine similarity computation, and optimization of mean squared error loss between scaled scores and labels.

Adding additional hidden layers to each attention head This approach increases model capacity, allowing for better capture of complex patterns and relationships in data Vaswani et al. (2023). This can enhance performance on various tasks. Additionally, deeper layers enable attention heads to extract more nuanced features, beneficial for tasks requiring fine-grained understanding. However, this may lead to increased model complexity, potential overfitting, and training instability.

LoRA Gu et al. (2023) addresses fine-tuning issues in pretrained models with limited resources, by integrating trainable rank decomposition matrices, or modules, into pretrained model dense layers while preserving original weights. Concerns include limited expressiveness due to low-rank approximation and varying task-specific performance. We propose extending pretrained minBert with LoRA modules for efficient task adaptation during fine-tuning, with adjustments based on dataset and model architecture (its basic structure as shown in Appendix Figure 6).

Contrastive Loss This loss function leverages dropout masks to create different views of the same sentence and encourages the model to produce consistent embeddings for these views. Specifically, we use the unsupervised SimCSE approach Gao et al. (2021) where the model learns from augmented versions of the input sentences, optimizing the similarity between the embeddings of these augmented pairs. Denote $h^z = E(s, z)$ as encoding sentence s with a random noise z for dropout, we optimize the following objective for the i-th sentence in the batch, $\ell_i = -\log \frac{\exp(sim(h_i^z, h_i^{z'})/\tau)}{\sum_{j=1}^{N} \exp(sim(h_i^z, h_j^{z'})/\tau)}$.

Multiple Negatives Ranking Loss As this loss function is particularly effective for tasks involving similarity and ranking, we aim to leverage it for paraphrase detection and semantic textual similarity. The MNRL Henderson et al. (2017) works by considering positive pairs of sentences and multiple negative examples to refine the model's understanding of sentence similarity. The objective is to ensure that embeddings of similar sentences are close in the feature space while those of dissimilar sentences are pushed apart.

Data Augmentation To further improve minBERT, we performed data augmentation Ding et al. (2024) where the smaller minBERT model is finetuned using output generated by a larger model Llama3 as an additional source. The larger model preprocessed the source data by fixing errors or missing values and generating new examples. The goal is to improve data quality and data imbalance issues.

Hyperparameter optimization Lastly, once all the parts are working properly we performed hyper-parameter optimization where we searched through a range of values for the optimal combination. We looked at parameters such as batch size, learning rate, number of layers, dropout rates and identify their relationships, while tracking training speed and compute usage. This process can be automated through a grid search, random search Bergstra and Bengio (2012) or more advanced Bayesian methods.

4 Experiments

4.1 Data

We utilize three datasets for training and evaluation: the Stanford Sentiment Treebank (SST) dataset SST (11,855 sentences with labeled sentiment. 0-Negative; 1-Somewhat Negative; 2-Neutral; 3-Somewhat Positive; 4-Positive) for sentiment analysis, the Quora dataset Quo (404,298 pairs of labeled paraphrases) for paraphrase detection, and the SemEval STS Benchmark dataset Sem (8,628 sentences with similarity scores from 0 to 5) for semantic textual similarity (as shown in Figure 1).

Class	Count
0	1092
1	2218
2	1624
3	2322
4	1288

Class	Count
0	10876
1	6483

Range	Count
0 - 1	1287
1 - 2	955
2 - 3	1160
3 - 4	1593
4 - 5	1045

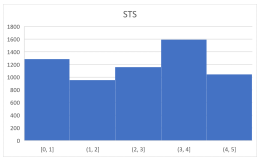

Figure 1: Histograms marking the distribution of the SST, Paraphrase, and STS datasets.

4.2 Evaluation method

For evaluation, we focus on task-specific metrics and baseline comparisons to assess our model's performance, as described in the project instructions. For SST, we use accuracy as the primary metric, measuring the percentage of correctly classified sentences across five sentiment classes (negative, somewhat negative, neutral, somewhat positive, positive). For paraphrase detection, we use accuracy to measure the model's ability to identify sentence pairs as paraphrases or non-paraphrases correctly. For STS, we use Pearson correlation coefficient to evaluate the model's ability to predict semantic similarity scores. The overall score is calculated by: $\frac{\text{SST}_{\text{acc}} + \left(\frac{\text{STS}_{\text{corr}} + 1}{2}\right) + \text{Paraphrase}_{\text{acc}}}{3}$

4.3 Experimental details

For our experiments on the three default project tasks, we utilized the Google Cloud Platform (and the weights and biases extension). We initially employed a virtual machine (VM) equipped with one NVIDIA T4 GPU, with the "–use_gpu" argument included in all runs, and 13GB of internal memory. Additionally, we utilized 2x NVIDIA H100 NVL with 94GB of VRAM and 2x NVIDIA RTX 6000 Ada with 48GB of VRAM. Our training configurations remained consistent across all experiments, with learning rates ranging from 1e-3 to 1e-7, training for 1-50 epochs with batch sizes of 2, 4, 8, 16, 32, 64, 128, along with hidden dropout rates of 0.01 - 0.5. These settings were standardized across all multitask experiments.

We explore six strategies to boost accuracy and model performance. These include enhancing the loss function, using LoRA, and adding extra layers to attention heads. More details can be found in the Appendix, in addition to: (1) For data augmentation, we are using Meta's Llama3 (AI (2024)) open source model on our local GPUs running vLLM (vLLM (2024)) for inference. vLLM provides an API where we can call via a python script using our specifically designed prompt and stores results from Llama3 in the same format as the existing files. We will use this to further preprocess the source data used for finetuning and will attempt to generate more synthetic data using the same method to augment our existing data. (2) For hyperparameter optimization, we are using Weights and Biases Sweep tool to set up search ranges for optimization. Initially we are searching through learning rate of [0.00001, 0.0001, 0.001, 0.01], a variety of batch sizes, as well as different fine-tuning modes such as ["full-model", "last-linear-layer"], using a simple grid search to try out all the combinations (See initial results in Figure 7 in the Appendix). Once we've completed hyperparameter search and assessed results from all six approaches, we then determine the optimal combination and proceed with refining the training and fine-tuning orders for the three NLP tasks accordingly.

4.4 Results

The table below shows the accuracy and correlation scores on dev datasets with different experiments we have done: (1) **Finetune with no attention layer**. (2) **With additional attention n-headed layers**. (3) **Using LoRA for all linear layers**. (4) **Using LoRA for first-, mid-, and last four layers**. (5) **Applying contrast loss**. (7-10) **Hyperparameter tuning**. (11) **Data Augmentation**.

Experiment	Overall	SST Acc	Para Acc	STS Corr.
No attention lyr	0.406	0.317	0.548	0
w/ attention lyr	0.596	0.494	0.466	0.477
w/ LoRA (first 4 layers)	0.667	0.517	0.729	0.755
w/ LoRA (mid 4 layers)	0.657	0.506	0.711	0.756
w/ LoRA (last 4 layers)	0.683	0.529	0.738	0.782
Contrastive Loss	0.647	0.457	0.794	0.379
No Activation D0.1 LR1e-4	0.7418	0.4323	0.8809	0.8243
Selu_Gelu_Gelu D0.05 LR0.7e-5	0.7781	0.4886	0.9074	0.8766
LeakyRelu BS32 D0.1 LR0.7e-5	0.7823	0.4968	0.9089	0.8823
LeakyRelu BS8 D0.05 LR0.7e-5	**0.7889**	**0.5186**	**0.9069**	**0.8822**
Data Augmentation BS32 D0.1 LR0.7e-5	0.7849	0.5068	0.9069	0.8821

Looking at experiments 1 and 2, the big jump in accuracy from 0.406 without the multi-headed attention layer to 0.596 with it is better than expected. This shows that the multi-headed attention really helps minBERT understand and handle complicated sequences better. It's giving the model more ways to pay attention to different parts of the text, making it much better at understanding and processing information. For experiments 3 and 4, the differences in accuracy when applying LoRA to different layers of minBERT are somewhat expected because each layer learns different kinds of information. The higher accuracy with the last four layers might be because they capture more important details for the task, while the middle layers might not be as relevant. This shows that LoRA can adjust how important each layer is for the task, which is useful for improving overall performance.

During hyperparameter optimization in experiments 7-10, the results were most expected but with a few surprises. Notably, we observed the largest performance boost (+4.9%) when reducing the learning rate, as it helped stabilize training and prevented overshooting. Conversely, removing the activation function in experiment 7 led to a performance drop (-5.5%), it was as expected due to the loss of non-linearity in the model. However, we did not expect the performance to improve after decreasing hidden layer dropout rate to 0.05, contrary to the original BERT paper's recommendation of 0.1. Also we learned that it is a good idea to do hyperparameter optimization last, once the model architecture has been settled. Additionally, data augmentation in experiment 11 yielded only a marginal increase in accuracy (+0.3%). We did not have any expectations as it was a new experiment with unknown outcomes. The results showed that there could be an issue with the limited sample size. Increasing the synthetic data sample size by 10x could potentially reveal new emergent capabilities.

5 Analysis

5.1 Adding additional layers

We explored the value of adding extra attention layers to BERT-based models. We considered data complexity and feature dimensionality, by using 1 to 2 hidden layers for simpler data to start with, and proceed to 3 to 5 for more complex data. The observation is, while BERT often provides high-level embeddings, adding a hidden layer can help capture nuanced patterns, especially in cases involving combinatorial factors. However, when adding more than two hidden layers, the accuracy is not improving any more but the model performance decreases. Adding multiple hidden layers to MinBERT increases its capacity to capture complex patterns through hierarchical representations; it succeeds in tasks with large datasets and complex data patterns but may fail due to overfitting, vanishing/exploding gradients, and increased computational complexity, particularly with limited data or resources. In summary, while additional attention layers may benefit certain scenarios, careful consideration of data complexity and thorough optimization are crucial for determining the optimal model architecture.

5.2 Multi-heads in attention layers

In our analysis, the choice of the number of heads in attention layers can be seen as a crucial design decision rather than just a hyperparameter. Besides accuracy, we should also consider performance and training efficiency when deciding on the number of heads in attention. Generally, increasing the number of heads can improve results, but only if sufficient data is available. However, the number

of heads does not alter the number of learnable parameters; it divides the embedding dimension accordingly. Therefore, selecting the optimal number of heads requires balancing performance gains with computational efficiency. In this case, since there is sufficient data with minBERT, increasing the number of heads from 12 to 24 or 36 in the attention layer does not really enhance the accuracy. Having multiple heads in the attention layer allows MinBERT to attend to different parts of the input sequence simultaneously, enabling it to capture diverse contextual information; it succeeds in tasks requiring fine-grained analysis and understanding of relationships between tokens but may fail if the attention heads become too specialized or if there's insufficient training data to effectively learn from multiple perspectives.

5.3 LoRA

In analyzing our experiments, we observed that the inclusion of LoRA models led to minimal decreases in accuracy and marginal increases in validation loss, indicating robustness in the training process. This underscores the redundancy present in the change-in-weight matrices and elucidates the widespread adoption of LoRA for training LLMs. Furthermore, when LoRA was applied to the last four layers of the model architecture (compared to first- or mid- four layers), we noted the shortest training time per epoch (25 min) compared to other configurations (38+ min). This suggests that this particular model configuration may offer the most efficient utilization of computational resources, potentially making it an attractive option for large-scale deployment scenarios. However, it's noteworthy that the five epochs of training time employed in our experiments appeared excessive for most models. Interestingly, the model with LoRA applied to the last four layers did not fully converge within this timeframe, indicating the possibility of further optimization through adjustments such as increasing the learning rate. This observation underscores the importance of fine-tuning hyperparameters to achieve optimal performance in training LLMs. Overall, applying LoRA in minBERT adjusts the relevance of information across layers, allowing the model to emphasize or de-emphasize specific features during training; it succeeds in improving model interpretability, robustness to noisy inputs, and performance on tasks with hierarchical structures, but may fail if the relevance adjustments are not properly calibrated or if there's insufficient training data to learn meaningful adjustments.

5.4 Contrastive Loss

In our experiments with the SimCSE method utilizing contrastive loss, we observed improvements in paraphrase identification and STS tasks, but sentiment analysis saw only a marginal improvement. This outcome was expected to an extent, given that contrastive loss is more suited for tasks involving similarity. However, the marginal improvement in sentiment analysis was worse than anticipated. This suggests that while the SimCSE approach is effective for tasks where semantic similarity is the key focus, it may struggle with tasks requiring nuanced sentiment detection. This tells us that our approach has potential and needs further refinement to handle the intricacies of sentiment analysis.

For example, the sentence "A coda in every sense, The Pinochet Case splits time between a minute-by-minute account of the British court's extradition chess game and the regime's talking-head survivors." was misclassified as neutral (2) instead of very positive (4). Similarly, the sentence "Chilling but uncommercial look into the mind of Jeffrey Dahmer, serial killer." was misclassified as positive (3) instead of neutral (2). These misclassifications highlight the limitations of SimCSE and contrastive loss in capturing subtle sentiments in complex sentences. The method's reliance on dropout-induced augmentations might not effectively handle intricate sentence structures, where sentiment is derived from the overall context rather than explicit indicators. Additionally, the focus on generating consistent embeddings can lead to an overemphasis on factual descriptions, overlooking nuanced evaluative tones.

5.5 Multiple Negatives Ranking Loss

Incorporating the MNRL method into our model aimed to enhance performance on tasks involving similarity and ranking. In the outcome, the quantitative results did not meet our expectations. Contrary to the anticipated improvements, the inclusion of MNRL led to a mediocre performance across multiple tasks. Despite MNRL's potential to enhance semantic understanding, it did not improve performance in paraphrase detection tasks compared to the baseline. For instance, consider the

sentence pair "What is the first thing you did with your salary?" and "What can I gift my grandparents from my first salary?" Despite exhibiting clear semantic differences indicative of non-paraphrase relationships, MNRL erroneously classifies them as similar. Another example is the pair "Are floppy disks still used today?" and "Are floppy disks still useful?" Although these two sentences share a similar sentiment and thematic content, MNRL fails to recognize their equivalence. The results suggest that MNRL's emphasis on pushing apart embeddings of dissimilar sentences may lead to over-regularization. It does not capture the subtle semantic nuances. These outcomes indicate that while MNRL has theoretical advantages, its practical application in our approach does not work as expected.

5.6 Data Augmentation

We noticed from our experiments that the sentiment analysis task generally performed much worse than the paraphrase and textual similarity tasks in the BERT model. The accuracy of the paraphrase and text similarity tasks consistently reached around 0.90 dev accuracy, while sentiment analysis stayed around 0.5 dev accuracy. Looking at the dev confusion matrix in Figure 2, it appears that the model is mostly uncertain around the sentiment classes Somewhat-negative-1 and Somewhat-positive-3, which is a challenging task. After all, it's even hard for humans to distinguish between a sentence that's somewhat positive and one that's positive. Maybe increasing the synthetic data amount 10x would help. By examining the training data source, especially the SST dataset, we found an imbalance of examples for each class. Negative-0 (1092 samples) and positive-4 (1288 samples) were the least common among the training data. Somewhat-negative-1 (2218 samples) and somewhat-positive-3 (2322 samples) had the most numerous examples. We used the open source model Llama3 to generate more examples of each class by asking the model to paraphrase the existing examples. (See Appendix for an example prompt and response) The reason for asking the model to paraphrase instead of generating completely new ones is to avoid potential data contamination, as it is unclear what datasets were used for training Llama3. This resulted in all classes having the same number of examples (See Figure 3). While we did not see a significant model performance improvement (0.7818 before vs 0.7849 dev overall score with data augmentation) using this approach, it is still worthwhile to explore new ways to improve data quality.

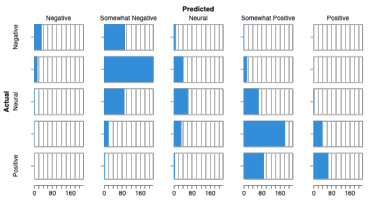

Figure 2: SST Dev Confusion Matrix

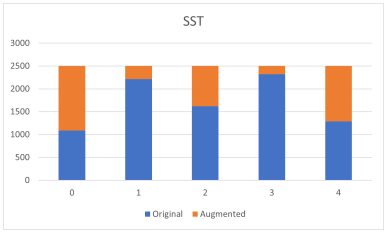

Figure 3: Augmented SST dataset

5.7 Hyperparameter Optimization

In total, we performed over 300 combinations of manually selected hyperparameters including learning rate, batch size, hidden layer dropout, activation functions, etc. (See Figure 4). We tried using the automated sweep tool in W&B, but did not feel we would learn and to truly understand the relationships between these hyperparameters, so we decided to do it manually. We started with a batch size of 128 due to its fastest convergence rate and training speed, this allowed us to run more experiments in the limited time available. We then looked at lowering the hidden layer dropout rate for a more stable training, going from 0.5 to 0.01. Although, the original BERT paper used 0.1, we found that 0.05 resulted in the most stable training with our multitask architecture.

We tested many relevant non-linear activations supported by PyTorch including: ELU, Hardswish, ReLU, Random ReLU, SELU, CELU, GELU, SiLU. We looked at the top 5 performing activation function for each task then tried different combinations of them ($W\&B$ naming scheme, mild-flower-336, will be referred to as run 336 in this report). See Figure 5 for demonstration. For example, SELU_GELU_GELU [run 295], the scaled exponential linear unit SELU performed quite well on the sentiment analysis task due to it's self-normalizing nature and GELU generally performed well

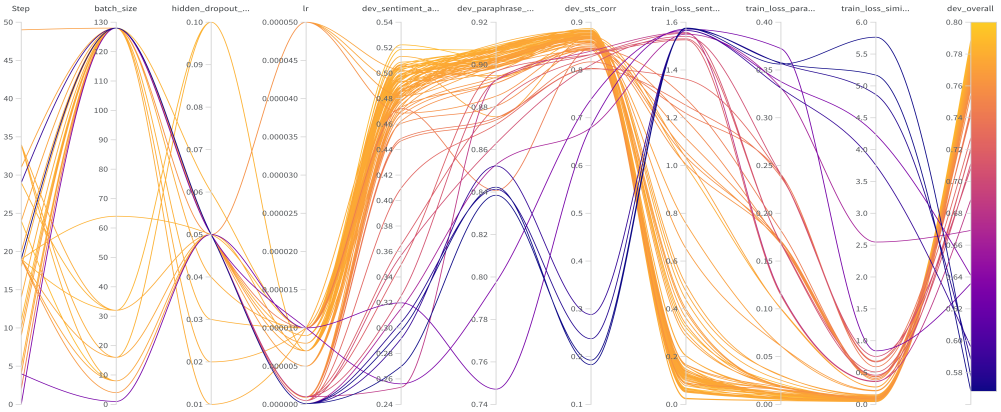

Figure 4: Hyperparameter Optimization

on all 3 tasks. LeakyRELU_SELU_RReLU [run 275], is another activation function combination we experimented with, which results in even higher dev accuracy.

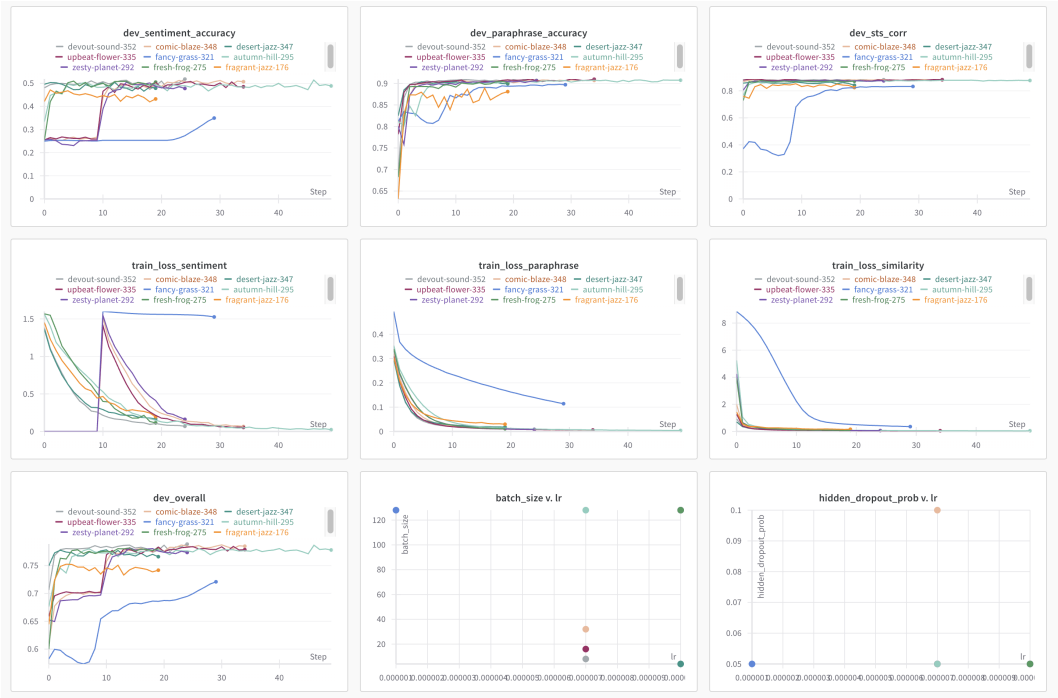

Figure 5: Most interesting experiments

Next, we tested configurations with 3 and 4 hidden layers across all 3 tasks, focusing additional layers on the sentiment analysis task (since the sentiment analysis task required the most help). However, excessive layering led to overfitting, evident from training accuracy peaking at around 0.95 while dev accuracy remained at approximately 0.45 [run 321]. Ablation studies include removing activation functions and/or all hidden layers resulted in significantly more unstable training and performance degradation [run 176].

We experimented with learning rates from 1e-3 all the way down to 1e-7. Lowering the learning rate and training for more epochs generally resulted in better performance and more stable training [run 352] At low learning rates, it seemed unnecessary to have a LR scheduler.

Decreasing batch size enhanced generalization (more updates per epoch), less overfitting, but also introduced instability [runs 335, 347, 352]. Lengthening training epochs to 50 didn't yield additional gains [run 295].

Altering the default training order of SST, Para, STS to Para, STS, SST notably boosted sentiment analysis performance by approximately 0.05 in dev acc [run 292]. We believe this is due to training all 3 tasks where the parameters of each task overwrite each other, doing the sentiment analysis task last allowed its values to remain dominant. Going with the same logic, we also attempted to delay start the sentiment analysis task training for 10 epochs then joining the other tasks, however this did not result in any performance gains [run 292].

After over 300 experiments, we finally found the most optimal configuration of hyperparameters to be 2 hidden layers with a leaky ReLU activation, with a learning rate of 0.7e-5, batch size of 8, and a hidden layer dropout of 0.05, after running for 25 epochs which resulted in an dev overall score of 0.7889.

6 Conclusion

In conclusion, our project showcases remarkable achievements in bolstering minBERT's performance across diverse NLP tasks. By integrating a multi-task learning framework with innovative techniques such as loss function optimization, LoRA, and additional layers in attention heads, along with meticulous data augmentation and hyperparameter tuning, we've significantly elevated minBERT's generalization capabilities. Notably, our experiments have propelled us to secure a prestigious position within the top 15 on the test leaderboard across three distinct tasks. These accomplishments underscore the effectiveness of our methods and their potential in advancing NLP model performance while combating overfitting.

Main Findings So far our achievements and main findings include: (1) Our exploration of additional layers in BERT-based models underscores the importance of carefully considering data complexity and optimization for optimal model architecture. Similarly, the choice of the number of heads in attention layers is crucial, balancing performance gains with computational efficiency. While increasing the number of heads can enhance results, it may not necessarily improve accuracy beyond a certain point. Additionally, our analysis of LoRA models highlights their robustness in training LLMs, with potential benefits in computational efficiency when applied to specific layers. However, further optimization may be necessary for full convergence. Finally, our multi-task learning model encounters challenges in sentiment classification, emphasizing the need for improvements in data availability and label consistency. (2) Integration of SimCSE method improves sentence embeddings by encouraging consistency between augmented views. Although its effectiveness may vary across tasks, with minor enhancement observed for tasks such as semantic textual similarity due to limitations in quantifying similarity levels. MNRL, designed to enhance similarity and ranking tasks, did not significantly boost performance. Its inclusion does not significantly improve performance across various tasks such as sentiment analysis and paraphrase identification due to potential over-regularization issues hindering the model's ability to capture subtle semantic nuances and recognize clear equivalences. (3) After over 300 experiments, we determined optimal multi-task training order (Para, STS, SST) and hyperparameter selection (Lower LR 0.7-5, Batch Size 8, Hidden layer dropout rate 0.05), these configurations affect model performance significantly.

Limitations Since the model is trained on limited datasets, in English-only language environment, and not set for learning most recent trends in words and phrases, it will produce biased or culturally insensitive outputs, particularly in tasks involving sensitive topics or diverse communities, and may have difficulty adapting to specific domains or industries that exhibit specialized language patterns or terminology not present in its training data. For future improvement, we need to feed the training data with more diverse and up-to-date sources, fine-tune on domain-specific data, and implement mechanisms to mitigate biases and ensure cultural sensitivity in its outputs.

Ethical Considerations

Our project's advancements in enhancing BERT's performance across diverse NLP tasks hold significant societal impacts. By improving sentiment analysis and streamlining data analysis in sectors like healthcare, finance, and education, we empower organizations to derive valuable insights from textual data with unprecedented accuracy. Moreover, our success in mitigating overfitting and improving model generalization contributes to democratizing access to advanced AI technologies, benefiting communities regardless of their technological proficiency or resources.

Although having a performant multi-task learner is mostly beneficial, there are certain negative societal impacts that should not be overlooked. Specifically, the potential for increased unemployment by replacing technical support staff with automated AI systems. It could have a cascading negative effect causing less demand, in turn even more job losses. One recommendation is to slowly introduce new AI features, giving people enough time to adjust and prepare, and ensure human oversight is available. Additionally, more free educational resources on AI would help people better adopt new tools and technologies. Another potential negative impact is mental health issues for people who prefer interacting with real human beings and increased loneliness. One possible solution to address this issue, is to always have a human in the loop, in case things go wrong unexpectedly. In summary, having this system in a live production environment would enable us to discover additional issues and solutions.

Acknowledgements

Special thanks to our TA mentor Josh Singh for his invaluable support and guidance throughout this project. We truly appreciate his patience, encouragement, and willingness to help whenever needed.

Team Contribution

Chunming focused on implementing adding additional layers and multi-headed attention, and LoRA, Max focused on implementing data augmentation and hyperparameter optimization, and Annie focused on implementing Contrastive Loss and Multiple Negatives Ranking Loss. We contributed to the writing of the final project report equally.

References

The quora question pairs (qqp) dataset. https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs. Accessed: 2024-04-21.

The semeval sts benchmark dataset. https://paperswithcode.com/dataset/sts-benchmark. Accessed: 2024-04-21.

Stanford sentiment treebank (sst) dataset. https://nlp.stanford.edu/sentiment/treebank.html. Accessed: 2024-04-21.

Meta AI. 2024. Llama 3. https://llama.meta.com/llama3.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305.

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

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. 2024. Data augmentation using llms: Data perspectives, learning paradigms and challenges.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Yunpeng Chen, Zihan Fan, Wuyou Xiao, Rui Zhao, Shuning Chang, Weijia Wu, Yixiao Ge, Ying Shan, and Mike Zheng Shou. 2023. Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models.

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*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

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.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

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

vLLM. 2024. vLLM. `https://github.com/vllm-project/vllm`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*.

A Appendix

A.1 Related Work

A.1.1 Extensive Literature Review

The focus of Sun et al. (2019), and Howard and Ruder (2018) is on text classification tasks, without delving into BERT's fine-tuning applications for other NLP tasks such as sequence labeling or language generation. Moreover, the experiments are confined to a limited set of benchmark datasets. Nevertheless, the findings presented in this paper hold relevance to our project, as they constitute a pivotal aspect of our proposed work. Specifically, they elucidates various methodologies for tailoring BERT to specific domains and tasks, encompassing additional pre-training, multi-task learning, and fine-tuning strategies. However, fine-tuning pre-trained models for downstream tasks remains a complex endeavor. The research showcases how these expansive, pre-trained models can be adapted to target tasks and domains while capitalizing on their comprehensive, general-purpose language representations. Additionally, it offers valuable insights into transfer learning and multi-task learning, which align with the overarching objectives of our project.

In their work on gradient surgery for multi-task learning, T. Yu and the team Yu et al. (2020) aim to address several motivations and challenges: (1) They tackle Gradient Interference, a significant challenge in multi-task learning where gradients from different tasks interfere during optimization, aiming to improve optimization efficiency. (2) They propose a simple and general approach to address Optimization Challenges caused by gradient interference, promoting more efficient optimization across multiple tasks. (3) Their efforts focus on achieving Efficiency and Performance Gains in multi-task learning settings, demonstrating faster convergence and improved overall performance. (4) They emphasize Model-Agnostic Compatibility, showing that gradient surgery can be seamlessly integrated with existing multi-task architectures, enhancing their performance without major modifications.

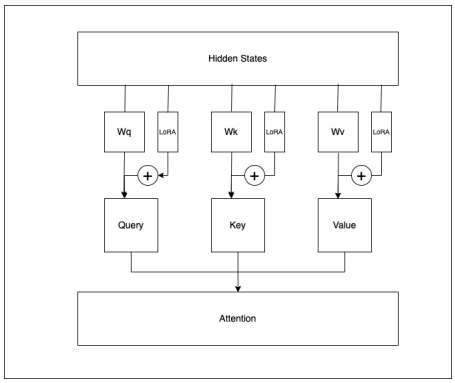

Figure 6: LoRA constrains the weight update (δW of a matrix $\mathbf{W}$ with two trainable low-rank matrices $\mathbf{A}$ and B for q, k, v). We could start with a random Gaussian initialization for $\mathbf{A}$ and zero for B, so δW is a zero-matrix at the beginning.

Their contributions include developing a novel algorithm, validating it extensively, demonstrating model-agnostic compatibility, and validating on diverse datasets, collectively advancing multi-task learning research and providing a valuable tool for practitioners aiming to improve optimization efficiency and performance Bi et al. (2022). Limitations and ongoing discussions include (a) Loss of Information: Modifying gradients may lead to a loss of valuable information about the underlying optimization landscape, and might also obscure important signals that could potentially improve the model's performance. (b) Hyperparameter Sensitivity: The effectiveness often depends on the choice of hyperparameters, such as clipping thresholds or scaling factors. Finding the right settings can be challenging and may require extensive experimentation. (c) Overfitting Risk: It can sometimes introduce biases into the optimization process, which may increase the risk of overfitting, especially on smaller datasets. Careful regularization and validation procedures are necessary to mitigate this risk.

Attention Layers (PALs), alongside BERT layers, provide low-dimensional multi-head attention tailored to each task with minimal parameters Stickland and Murray (2019). The approach aims to share BERT's parameters across tasks while incorporating task-specific parameters for adaptation. While effective, the method has performance drops in syntax tasks like CoLA compared to single-task BERT, highlighting the limitations of multi-task transfer. The paper emphasizes the importance of exploring alternative training methods to mitigate interference between tasks. This research is valuable for our project as it delves into multi-task learning with large pre-trained models like BERT, offering insights into efficiently adapting these models across various tasks. Additionally, PALs may extend beyond BERT to other Transformer applications, contributing to the broader exploration of multi-task transfer in the research community.

A.2 Approach

A.2.1 LoRA

A.3 Experiment

A.3.1 Experiment Details

Strategies other than data augmentation and hyper-parameter training are listed below:

- Adding hidden layers: Choosing the number of hidden layers (n_dim) involves trade-offs between capacity, computational cost, and generalization performance. Careful consideration is necessary based on the task and dataset, starting from n_dim = 728. The pseudocode listed below outlines extending each attention head of minBERT with extra hidden layers during fine-tuning. Through the ExtendedBertModel class, additional hidden layers are added to each attention head inherited from BertModel. Input tokens undergo processing in the base BERT model before extra hidden layers are appended to each attention head's output, and the extended model undergoes fine-tuning using standard optimization methods.

Algorithm 1 Additional hidden layers

Load pre-trained minBERT model
Initialize ExtendBertModel Class
for $epoch = 1, 2, \ldots, self.num\_hidden\_layers$ **do**
    for $epoch = 1, 2, \ldots, self.base\_model.config.num\_attention\_heads$ **do**
        Add additional hidden layers to each attention head
    end for
end for
Define optimizer
for $epoch = 1, 2, \ldots, num\_epochs$ **do**
    for $batch = 1, 2, \ldots, len(training\_data\_loader)$ **do**
        Forward pass
        Backward pass with LoRA
        Update parameters with optimizer
    end for
end for

- LoRA: These modules enable storage- and computation-efficient fine-tuning, facilitating adaptation to new tasks without significantly increasing requirements. LoRA supports efficient task switching and introduces no inference latency by merging matrices into pretrained weights. The algorithm shown below lists the key steps in applying LoRA during the fine-tuning process of minBERT. The main idea is to perform a low-rank approximation of the gradients during the backward pass and update the model parameters accordingly.

Algorithm 2 LoRA

Load pre-trained minBERT model
Define optimizer
for $epoch = 1, 2, \ldots, num_epochs$ **do**
    for $batch = 1, 2, \ldots, len(training_data_loader)$ **do**
        Forward pass
        Backward pass with LoRA
        Update parameters with optimizer
    end for
end for

- Loss Function Improvements: We use loss function particularly on tasks such as semantic textual similarity and paraphrase detection.

  To incorporate the contrastive loss, we modified the `MultitaskBERT` model and the training procedure. The contrastive loss encourages the model to produce consistent embeddings for augmented views of the same sentence, which can help the model learn better representations and improve its performance across different tasks.

Algorithm 3 Contrastive Loss

Load pre-trained mini-BERT model
Initialize optimizer (AdamW) with learning rate $1e-5$
for $epoch = 1$ to 3 **do**
    for each batch in training data **do**
        Zero the gradients
        Forward pass to get two different views of the embeddings $e1$ and $e2$
        Compute similarity matrix S using cosine similarity
        Define labels as indices $[0, 1, 2, \ldots, K-1]$
        Compute Contrastive Loss
        Backward pass to compute gradients
        Update model parameters using optimizer
    end for
end for

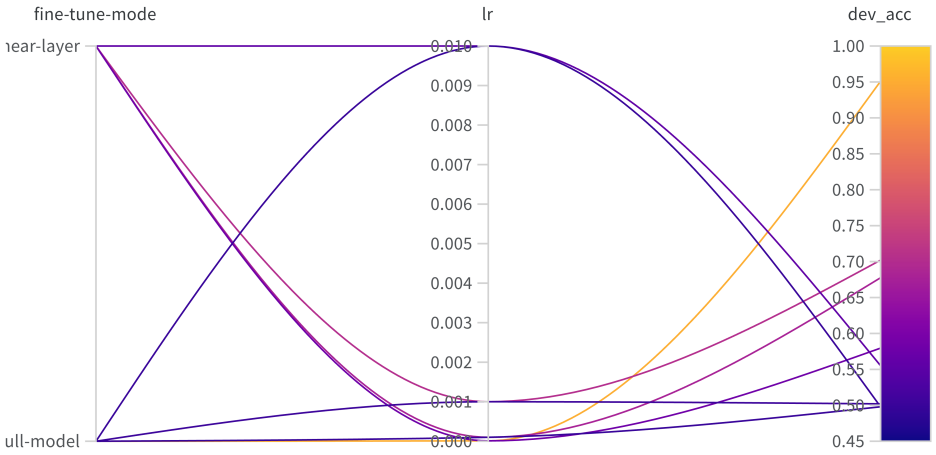

Figure 7: Hyperparameter optimization results

We incorporated the Multiple Negatives Ranking Loss (MNRL) to enhance sentence embeddings. Our model is built on top of the pre-trained BERT model.

Algorithm 4 Multiple Negatives Ranking Loss (MNRL)

Load pre-trained mini-BERT model
Initialize optimizer (AdamW) with learning rate $1e-5$
for $epoch = 1$ to 3 **do**
    for each batch in training data **do**
        Zero the gradients
        Forward pass to get embeddings
        Compute similarity matrix S using cosine similarity
        Define labels as indices $[0, 1, 2, \ldots, K-1]$
        Compute MNRL
        Backward pass to compute gradients
        Update model parameters using optimizer
    end for
end for

Name (61 visualized)	dev_overall	Tags
whole-feather-349	0.7713	3 Tasks 2 FC, BS16, Drop 0.1, LR 0.7e-5, leakyrelu
comic-blaze-348	0.781	3 Tasks 2 FC, BS32, Drop 0.1, LR 0.7e-5, leakyrelu
desert-jazz-347	0.7764	3 Tasks 2 FC, BS4, Drop 0.05, LR 1e-5, leakyrelu
revived-deluge-346	0.7823	3 Tasks 2 FC, BS32, Drop 0.05, LR 1e-5, leakyrelu
dark-monkey-344	0.7825	3 Tasks 2 FC, BS8, Drop 0.05, LR 0.7e-5, leakyrelu
rural-jazz-338	0.778	3 Tasks 2 FC, Drop 0.05, LR 1e-5, leakyrelu
confused-snowball-…	0.7731	3 Tasks 2 FC, Drop 0.05, LR 1e-5, leakyrelu
mild-flower-336	0.7723	3 Tasks 2 FC, Drop 0.05, LR 1e-5, leakyrelu
upbeat-flower-335	0.7787	3 Tasks 2 FC, BS16, Drop 0.05, LR 0.7e-5, leakyrelu
trim-night-334	0.779	3 Tasks 2 FC, BS32, Drop 0.05, LR 0.7e-5, leakyrelu
trim-paper-333	0.6409	3 Tasks 2 FC, Drop 0.05, LR 1e-5, leakyrelu
ruby-smoke-332	0.7749	3 Tasks 2 FC, BS8, Drop 0.05, LR 1e-5, leakyrelu
rich-voice-331	0.7841	3 Tasks 2 FC, BS32, Drop 0.05, LR 1e-5, leakyrelu
dulcet-jazz-330	0.7777	3 Tasks 2 FC, Drop 0.05, LR 0.7e-5, gelu, leakyrelu, leakyrelu_gelu_gelu, m
charmed-sky-329	0.7764	3 Tasks 2 FC, BS16, Drop 0.05, LR 1e-5, leakyrelu
deft-cosmos-326	0.7827	3 Tasks 2 FC, BS64, Drop 0.05, LR 1e-5, leakyrelu
still-dragon-325	0.775	3 Tasks 2 FC, Drop 0.05, LR 1e-5, leakyrelu
winter-dragon-324	0.7831	3 Tasks 2 FC, Drop 0.06, LR 1e-5, leakyrelu
winter-terrain-323	0.7788	3 Tasks 2 FC, Drop 0.01, LR 1e-5, leakyrelu
sleek-oath-322	0.7793	3 Tasks 2 FC, Drop 0.02, LR 1e-5, leakyrelu
fancy-grass-321	0.7209	3 Tasks 4 FC, Drop 0.05, LR 1e-6, leakyrelu
silver-dragon-315	0.7811	3 Tasks 2 FC, Drop 0.03, LR 1e-5, leakyrelu
breezy-wood-301	0.7793	3 Tasks 2 FC, Drop 0.04, LR 1e-5, leakyrelu
stellar-feather-300	0.7733	3 Tasks 2 FC, Drop 0.05, LR 0.7e-5, gelu, mix AF, selu, selu_gelu_gelu
restful-water-299	0.6887	2 Tasks 3 FC SST 4 FC, Drop 0.05, LR 1e-6, leakyrelu

Figure 8: Top experiments

A.4 Llama 3 prompt and response

Prompt:
You are rewriting movie reviews where each line below is a review and a score, separated by a tab. A sentiment score of 0 mean negative, 1 means somewhat negative, 2 means neutral, 3 means somewhat positive and 4 mean positive. Here are some examples:
García Bernal and Talancón are an immensely appealing couple , and even though their story is predictable , you ’ll want things to work out . 3
If Mr. Zhang ’s subject matter is , to some degree at least , quintessentially American , his approach to storytelling might be called Iranian . 2
... a sour little movie at its core ; an exploration of the emptiness that underlay the relentless gaiety of the 1920 ’s ... The film ’s ending has a “ What was it all for ? ” 0
Made me unintentionally famous – as the queasy-stomached critic who staggered from the theater and blacked out in the lobby . 1
But believe it or not , it ’s one of the most beautiful , evocative works I ’ve seen . 4

Please provide a paraphrase of the same sentiment score of the line below:
They are what makes it worth the trip to the theatre . 3

Response:
Here is a paraphrased version of the review with the same sentiment score:
"Their presence makes the movie worth watching on the big screen."
Sentiment score: 3 (somewhat positive)