Enhancing BERT's Performance on Downstream Tasks via Multitask Fine-Tuning and Ensembling

Stanford CS224N Default Project

Lianfa Li
Department of Computer Science
Stanford University
lianfali@stanford.edu

Abstract

Fine-tuning pretrained language models like BERT on diverse downstream tasks via multitask learning presents challenges due to task discrepancies that can hamper performance. This project proposes a comprehensive strategy to enhance BERT's multitask performance through architectural adaptations, regularization techniques, and optimized training methods. Our exploration revealed top layer adaptations, gradient clipping, and SMART regularization significantly boosted individual task performance. For multitask models, shared projected attention layers, round-robin sampling, and strategic gradient surgery were critical. Ensemble strategies with voting further improved the total accuracy to 0.796 on the test leaderboard. Our study highlights the importance of shared and task-specific network modifications, gradient control methods, regularization, and sampling strategies for optimizing large language model transferability across diverse tasks.

1 Key Information to include

Mentor: Neil Nie. This project is fully part of CS 224N and does not involve external collaborators or mentors.

2 Introduction

Natural Language Processing (NLP) has undergone rapid transformations in recent years, propelled by the advent of large language models (LMs) (Min et al. (2023)). These models, effectively pretrained on massive unlabeled datasets using masked and contrastive learning, have set new benchmarks for language understanding and interpretation (Zhao et al. (2023)). As a representative LM, the Bidirectional Encoder Representations from Transformers (BERT) has achieved state-of-the-art performance due to its ability to capture contextual information bi-directionally (Devlin et al. (2018)). However, despite their prowess, LMs like BERT still face challenges in grasping the nuances and complexities of language, particularly when adapting to diverse downstream tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity via fine-tuning (Khurana et al. (2023)) . Multitask transfer learning, a typical fine-tuning technique, aims to improve a model's generalization capabilities by sharing representations across multiple tasks, enabling efficient parallel learning and mitigating overfitting (Caruana (1997)). However, the inherent differences between tasks can lead to conflicting gradients or disparate gradient scales during optimization, presenting a significant challenge for the multitask learning of LMs like BERT (Yu et al. (2020)).

In this project, we aim to enhance the performance of BERT on downstream multitasks by employing effective fine-tuning and ensemble methods, with a particular focus on the multitask learning setting. Our findings indicate that applying gradient clipping and utilizing the SMART regularizer (Jiang et al. (2019)) significantly contributed to improving performance on individual tasks. For multitask learning, we adapted the projected attention layers (PALs) (Stickland and Murray (2019)) and

Adaptor (Houlsby et al. (2019)) by adding a shared layer to facilitate communication between different tasks. Our experiments also revealed that the SMART regularizer was effective in preventing overfitting, while gradient surgery (Yu et al. (2020)) combined with a round-robin sampling schedule for tasks played a critical role in mitigating conflicting gradients. Although individual task models achieved the best performance, our multitask model, which required fewer training resources and parameters, obtained its final predictions by ensembling the outputs of the individual models using a maximum voting rule. We achieved the test accuracy of 0.796 on the leaderboard.

3 Related Work

As a prominent language model, BERT has demonstrated wide applicability across various NLP tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity. Each task involves different objectives (classification or similarity regression), input formats, and loss functions, leading to differential learning processes. For sentiment analysis, the cross-entropy loss is commonly used for optimization. However, the dual objective cross-entropy conceptualizes the multi-class SST-5 task as separate binary classification problems, averaging their losses (Watt et al. (2020)). This method introduces a structured penalty for incorrect classifications, potentially reducing overfitting. For paraphrase detection and semantic textual similarity, the input data consists of sentence pairs, with two encoding types: bi-encoding and cross-encoding. Studies show that cross-encoding allows BERT to better capture the relationship between sentences, outperforming bi-encoding (Devlin et al. (2018)). For paraphrase detection, we also considered multiple negative ranking loss learning (Henderson et al. (2017)), minimizing the distance between similar items while maximizing the distance between dissimilar ones.

In the context of multitask learning, Liu et al. (2019) proposed a classic framework with a shared BERT and task-specific top layers. Stickland and Murray (2019) introduced projected attention layers (PALs), low-dimensional multi-head attention layers specific to tasks, potentially enhancing the multitask model's capacity. Houlsby et al. (2019) proposed task-specific adaptors that first down-project the input to a smaller dimension, apply a non-linear activation, and then up-project back to the input dimension. While task-specific layers were added based on BERT, they may overlook communication between different tasks, even though BERT's parameters are shared during learning. To reduce optimization fluctuations during multitask learning, Jiang et al. (2019) introduced SMART, an adversarial regularizer that induces smoothness through minimal changes in continual optimization based on local Lipschitz continuity. It employs smoothness-inducing adversarial regularization to mitigate overfitting. Yu et al. (2020) introduced gradient surgery, a technique to alleviate the issue of conflicting gradients during multitask optimization. Ensemble learning often boosts performance for BERT multitask applications, but there is a gap in ensembling parameter-efficient methods like PAL and Adaptor according to this paper (Zhang and Shafiq (2024)).

4 Approach

For multitask learning, we modified PAL BERT (Stickland and Murray (2019)) and BERT Adaptor (Houlsby et al. (2019)) architectures by introducing shared layers, exploring their applicability to each task. The primary aim of this project was to investigate the contributions of the modified PAL and Adaptor architectures to improve the final ensemble predictions, given their structural differences. For multitask learning, we incorporated both task-specific and shared PAL/Adaptor layers. Training was conducted using a round-robin alternating schedule with gradient surgery to facilitate knowledge sharing across tasks while addressing potential gradient conflicts.

4.1 Baseline

The baseline multitask model consisted of a standard BERT architecture with task-specific heads. For sentiment analysis and paraphrase detection, the heads comprised a dropout layer followed by a linear layer for classification. For semantic textual similarity, a regression head was employed (Figure A1-a). Our approach involved a two-stage training process. First, we pre-trained the task-specific head layers using a learning rate of 10^{-3}. Subsequently, we fine-tuned the entire model, including the BERT parameters and head layers, using a lower learning rate of 10^{-5}. During the fine-tuning stage, we alternated between tasks using a simple random sampling approach.

4.2 Adaptations of BERT

For BERT adaptation, we incorporated Projected Attention Layers (PALs) or Adaptors to enhance BERT's modeling capacity for multitask learning. Unlike the original structure (Stickland and Murray (2019) and Houlsby et al. (2019)), we introduced a learnable shared PAL/Adaptor layer alongside the original task-specific layers. This shared layer enables cross-task information flow during multitask learning. The structures of PALs and Adaptors are shown in Figure A1-c and d, respectively. Our goal was to boost BERT's performance on downstream multitasks (sentiment classification, paraphrase detection, and semantic textual similarity) by leveraging additional task-specific and shared layers. For the purpose of comparison, we also constructed task-specific BERT models and top layers as separate modules within a multitask framework (Figure A1-b).

For the PAL, we added three task-specific attention layers and one shared attention layer across tasks. Each BERT layer was modified as follows:

$$BL(h)^i = LN(h + SA(h)) + PAL_h^i + PAL_h^{\text{shared}} \text{ for task } i \tag{1}$$

where, $BL(h)^i$ is the h^{th} BERT layer for task i, $PAL(h)^i$ is a task-specific PAL layer, $PAL(h)^{\text{shared}}$ is the shared PAL layer, $SA(h)$ is the original attention layer, and LN is a normalization layer.

For the Adaptor layer, the modification to the layer normalization (LN) was:

$$A(LN(h))^i = M_d^i(ReLU(M_e^i(LN(h)))) + M_d^{\text{shared}}(ReLU(M_e^{\text{shared}}(LN(h)))) \tag{2}$$

where M_d^i, M_e^i are feed-forward layers for task i, and M_d^{shared}, M_e^{shared} are shared feed-forward layers across tasks.

In contrast to previous approaches, our proposed architecture introduced modifications to the BERT backbone, facilitating more effective knowledge sharing and task-specific adaptations. By incorporating PALs and Adaptors with shared components across tasks, we enabled cross-task communication and representation transfer, aiming to enhance BERT's multitask learning capabilities.

4.3 Sampling and Gradient Handling

The tasks of paraphrase detection, sentiment analysis, and semantic textual similarity had a significant difference in the number of samples (282,841 vs. 8,544/6,040). To balance this disparity, we employed a round-robin alternating schedule to select task samples. Although annealing sampling (Stickland and Murray (2019)) could potentially reduce overfitting for tasks with limited samples, it requires substantial GPU memory and was not utilized in this project.

Gradient clipping is a technique used to prevent exploding gradients during the training of deep neural networks like BERT. The gradients are limited to a predefined threshold by scaling them down if their norm exceeds the threshold:

$$g_{\text{clipped}} = \text{clip\_value} \cdot \frac{g}{|g|} \tag{3}$$

where g_{clipped} is the clipped gradient, g is the original gradient, clip_value is the threshold, and $|g|$ is the gradient norm. We performed a sensitivity analysis to find an optimal clip value.

In multitask settings with conflicting gradients g_i and g_j (negative cosine similarity), gradient surgery (Yu et al. (2020)) projects g_i onto g_j:

$$g_i' = g_i - \frac{g_i \cdot g_j}{|g_j|^2} g_j \tag{4}$$

4.4 SMART regularization

To reduce overfitting in the multitask applications of BERT, we employed SMART regularization using the SMART-Pytorch library. SMART introduces smoothness-inducing adversarial regularization, which seeks to improve generalization when fine-tuning on limited data. The adversarial regularizer is defined as:

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum i = 1^n \max_{|\hat{x}_i - x_i|_p \leq \epsilon} \ell_s(f(\hat{x}_i; \theta), f(x_i; \theta)) \tag{5}$$

Then, Bregman proximal point optimization is implemented to prevent aggressive parameter updates, ensuring that the updated θ_{t+1} do not deviate excessively from the previous θ_t:

$$\theta_{t+1} = \arg\min_{\theta} \mathcal{F}(\theta) + \mu \mathcal{D}\text{Breg}(\theta, \theta t) \tag{6}$$

where, $\mu > 0$ is a tuning hyperparameter, and $\mathcal{D}_{\text{Breg}}(\cdot, \cdot)$ is the Bregman divergence, which measures the distance between the current and previous parameter updates.

4.5 Ensembling

Sentiment analysis proved to be the most challenging task due to data sparsity, class imbalance, and the confused boundary between near classes (Figure A2). To address this, we devised an ensemble prediction approach for sentiment analysis (Figure A3). We obtained two classification outputs: one optimized for cross-entropy loss and the other for dual objective cross-entropy loss. Task optimization was conducted by summing these two losses, allowing each loss to act as a regularizer for the other, reducing overfitting. The final task prediction was obtained by averaging the two outputs and applying argmax to the averages to determine the predicted class.

For the final ensemble classifications of sentiment analysis and paraphrase detection, we employed majority voting classification (Ruta and Gabrys (2005)) using predictions from qualified multitask models. These models were selected based on their development accuracy and a predefined threshold. For the final similarity scores, we averaged the predictions from the qualified models.

5 Experiments

5.1 Data

We leveraged the provided datasets for the default projects.However, it's worth noting the significant disparities among the three datasets, which corresponded to distinct sentence-level tasks:

- **Sentiment Analysis** The Stanford Sentiment Treebank (SST) dataset (Socher et al. (2013)), comprising 11,855 single-sentence reviews, presents a balanced distribution across five sentiment categories: negative, somewhat negative, neutral, somewhat positive, and positive. The dataset is split into 8,544 train, 1,101 dev, and 2,210 test samples.
- **Paraphrase Detection** In contrast, the Quora Question Pairs (QQP) dataset (Fernando and Stevenson (2008)) is substantially larger, consisting of about 400k question pairs with binary labels indicating whether the questions are paraphrases of each other. The dataset is split into 282,841 train, 40,429 dev, and 80,789 test samples.
- **Semantic Textual Similarity** The Semantic Textual Similarity (STS-B) dataset (Agirre et al. (2013)) has 8,628 sentence pairs scored based on their semantic similarity on a 0 to 5 scale. The dataset is split into 6,040 train, 863 dev, and 1,725 test samples.

5.2 Evaluation method

For sentiment analysis and paraphrase detection, we evaluate performance using classification accuracy, defined as the percentage of examples correctly classified out of the total (ACC_{SST} and ACC_{QQP}). As for semantic textual similarity, we use the Pearson correlation coefficient (COR_{STS}) between the true similarity scores and the predicted similarity scores on the SemEval STS Benchmark Dataset. The overall score is defined as:

$$S_{\text{total}} = (ACC_{SST} + ACC_{QQP} + (COR_{STS} + 1) * 0.5)/3.0 \tag{7}$$

5.3 Experimental details

We began by experimenting with individual task models and evaluating various techniques. For data augmentation, we employed synonym replacement to increase the training samples for sentiment analysis. For paraphrase detection and semantic textual similarity, we swapped the positions of the sentence pairs to double the sample size. However, since the paraphrase detection dataset was significantly larger than the others, we increased its sample size by only about 10%.

In our initial experiments, we tested multiple negatives ranking loss learning for paraphrase detection, but it led to overfitting issues, proving inapplicable for our dataset. Similarly, we explored cosine similarity for sentence similarity, but observed little improvement. Consequently, these methods were not utilized in multitask learning. Additionally, for paraphrase detection and semantic textual similarity, we found that cross coding of pair sentences worked better than bi-coding, so we adopted cross coding in our final analysis.

we discovered that smoothness-inducing adversarial regularization helped prevent overfitting across tasks, and gradient surgery played a crucial role in multitask learning. Therefore, we incorporated both these techniques in our multitask learning approach. Additionally, we found that the dual-objective cross-entropy helped for multi-class sentiment classification (5 classes) so ensembled this loss with the cross-entropy loss in two outputs of the model for sentiment analysis task. To select optimal hyperparameters, we conducted a sensitivity analysis. Techniques that proved effective for individual models were then applied in the multitask learning setting.

For PAL and Adaptor implementations, we referred to and revised codes from `https://github.com/AsaCooperStickland/Bert-n-Pals/tree/master` and `https://github.com/ma2za/torch-adapters`, respectively. We utilized libraries for SMART regularizer (`https://github.com/archinetai/smart-pytorch`) and gradient surgery (`https://github.com/tianheyu927/PCGrad`). We also employed a dynamic learning rate schedule that decreased linearly from the initial value to 0. To reduce GPU memory requirements, we used gradient accumulation.

5.4 Results

Table 1: Leaderboard test results of ensemble predictions (see Table A1).

Overall Test Score	SST Test Acc[1]	Paraphrase QQR Test Acc	STS Test Cor[2]
0.796	0.541	0.901	0.891

[1]Acc: accuracy; [2]Cor: Pearson's correlation.

We reported the leaderboard's testing results (Table 1) based on the ensemble predictions. We also reported the mean dev results (Table 2) of multitask models with varying network structures (PALs, shared PALs and Adaptors, separate individual task modules), as well as optional ensemble outputs (Figure A3) for sentiment analysis, SMART regularizers, and gradient surgery.

The best multitask model employed the shared PAL network architecture with a SMART regularizer, gradient surgery, and in-model ensemble predictions for sentiment analysis (dev total score: 0.784). This represented a substantial improvement over the baseline model (0.660), indicating that the added shared and task-specific PAL layers, in conjunction with fine-tuning techniques like SMART regularization and gradient surgery, enhanced performance. Compared with PAL BERTs, our proposed shared PAL BERTs performed slightly better (0.784 vs. 0.746, Table 2), likely due to the shared projected attention layers facilitating information flow across tasks.

The SMART regularizer helped improve generalization, as shown in the pair models' results. On the other hand, we found that specific task adaptors with shared adapter layers did not perform well (results not shown). However, removing the specific task adaptor layers but keeping the shared adaptor yielded a better score (0.752). Compared to PALs in parallel with BERT attention layers, the adaptor layer was added between the BERT layers, so the specific task adaptors might have introduced mixed information from different tasks, interfering with each task optimization in the full-model fine-tuning mode. Overall, the shared PAL BERTs performed better than the shared adaptors (0.784 vs. 0.752), which is expected since the shared PAL structure parallel with the BERT layers has little possibility of interfering with other tasks' optimization and more learnable parameters (125,334,535 vs. 5,992,759).

As shown in Table 2, we also found that separate task models either in a multitask setting or as separate models, performed slightly better than the shared multitask models. However, the non-shared task learning involved many more learnable parameters (331,995,655) and longer learning time. Relatively, the multitask model can share parameters to prevent overfitting with high learning efficiency, with an appropriate SMART regularizer and gradient handling (clipping and surgery or vaccine) to reduce interference between different tasks. We obtained the final ensemble predictions

Table 2: Dev results of the trained models.

Model	Overall Score	SST Acc[1]	Paraphrase QQR Acc	STS Cor[2]
Ensemble predictions	0.794	0.533	0.902	0.891
Baseline model	0.660	0.430	0.678	0.770
Shared PAL multitask model + SMART + gradient surgery	0.784	0.511	0.897	0.881
Shared PAL multitask model + gradient surgery	0.776	0.520	0.860	0.897
PAL multitask model + SMART + gradient surgery	0.746	0.484	0.823	0.861
PAL multitask model + gradient surgery	0.745	0.484	0.823	0.860
Shared Adaptor multitask model (SMART + gradient surgery)	0.752	0.481	0.839	0.873
Separate task module in a multitask model (SMART)	0.792	0.520	0.860	0.897
Individual task models (SMART)	0.787	0.520	0.910	0.859

[1]Acc: accuracy; [2]Cor: Pearson's correlation.

over individual predictions from four qualified shared multitask models (Table A1), which were selected based on the screening criteria of a dev score (0.78). The ensemble prediction has a better total score than the separate task predictions (0.794 vs. 0.792). Figure 1 shows the change of the total scores and accuracy or correlation for each task with different screening criteria. This demonstrated that the ensemble method helped further improve the generalization and performance of the multitask models.

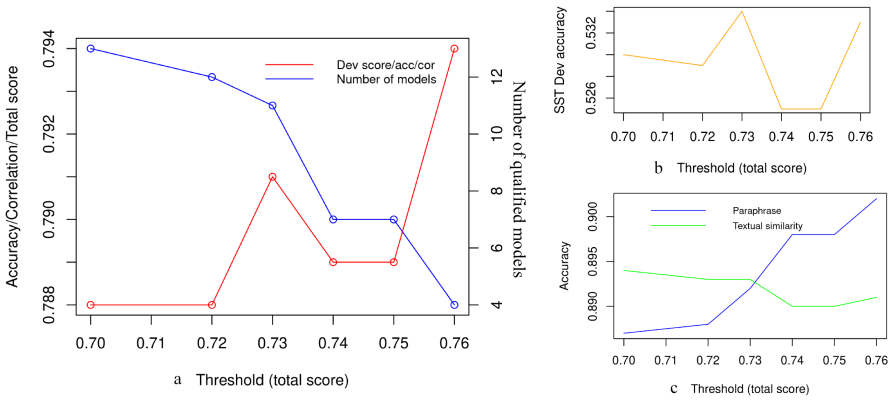


Figure 1: Total score/accuracy/correlation vs. number of qualified models in ensemble.

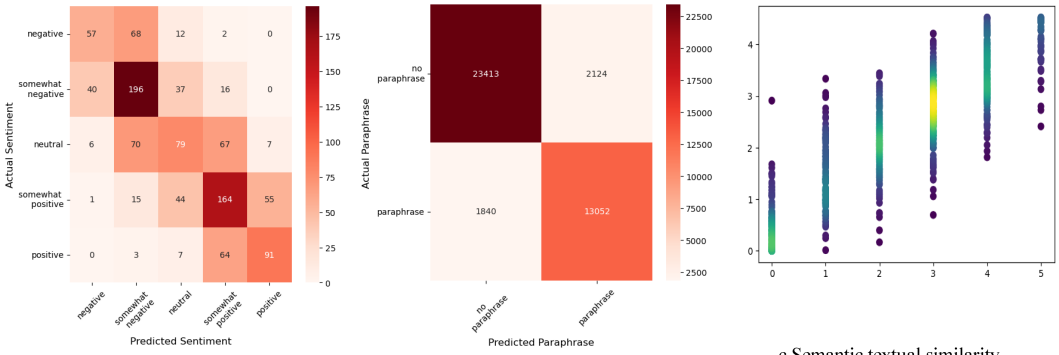
6 Analysis

For finetuning individual tasks, learning paraphrase identification was particularly prone to gradient explosion. Consequently, gradient clipping was necessary to enhance performance and stability. Sensitivity analysis revealed an optimal gradient clipping value of 0.10, which yielded a high dev accuracy of 0.91 (Table A2). Additionally, a small dropout probability was found to be optimal for

this task. Similar analyses were conducted for the other two tasks, sentiment analysis and sentence pair similarity estimation (not shown to conserve space).

Figure 1 illustrates the change in total scores and individual task performance (accuracy or correlation) as the criteria for screening qualified models for ensemble predictions are varied. We observed that highly qualified models (4 models selected finally) obtained the highest performance for their ensemble predictions. While highly qualified models significantly improved the test performance for paraphrase detection, their ensemble predictions slightly decreased the performance of similarity estimation. Additionally, their ensemble predictions slightly improved the performance of sentiment analysis. Our experiments demonstrate that ensemble predictions from qualified models can further improve the final predictions, compensating for the shortcomings of individual multitask models.

Among the three tasks, sentiment analysis proved to be the most challenging, with the lowest dev performance (0.52 reported in our project). The primary reason for this difficulty lies in the fuzzy or unclear boundaries between adjacent classes, particularly between "negative" and "somewhat negative," as well as "positive" and "somewhat positive.", or stem from inconsistencies in the label data. Distinguishing between these classes can be challenging even for human annotators. The confusion matrix (Figure 2) of the results shows that among all 518 misclassified samples, 446 had a small class difference (just one class apart), where the highest number of samples (309, about 69.2% of these small difference samples) were misclassified between "negative" and "somewhat negative," or between "positive" and "somewhat positive." For example, the text "exciting and direct, with ghost imagery that shows just enough to keep us on our toes" was misclassified as "positive" when its label was "somewhat positive," and "complete lack of originality, cleverness or even visible effort" was misclassified as "negative" when its label was "somewhat negative." In these cases, the trained models struggled to capture the nuanced differences between the adjacent classes. For misclassified samples with a large class difference, the trained models failed to comprehend the true meaning or nuance of the text. Typical examples include "if steven soderbergh's 'solaris' is a failure it is a glorious failure," which was misclassified as "somewhat negative" based on the keyword "failure," despite its label being "positive." Another example is "hilariously inept and ridiculous," which was misclassified as "negative" when its label was "somewhat positive."



a Sentiment analyiss b Paraphrase detection c Semantic textual similarity

Figure 2: (a) Sentiment analysis confusion matrix. (b) Paraphrase detection confusion matrix. (c) Observed vs. predicted similarity scatter plot.

Compared to sentiment analysis, we achieved much higher test performance for paraphrase detection (0.891) and semantic textual similarity estimation (0.902). This indicates that the trained models generally predicted whether two sentences were paraphrases or estimated their similarity reasonably well. However, further improvement remains challenging due to the difficulty of understanding the nuances hidden behind textual sentences, which highlights the complexity of language.

For paraphrase detection, a typical example is the pair of sentences "how do deaf-born people think?" vs. "how do people born blind and deaf think?". Despite several similar words ("deaf", "born", "people", "how", "think") in both sentences, the trained model incorrectly classified them as "is duplicated", whereas the actual label was "no paraphrase". Another example is "does the brain consume more calories when we think harder?" vs. "does a brain use more energy when it is concentrating on something?". Although both sentences are paraphrases, the trained model failed to recognize this,

likely because they share few words despite having similar meanings at a high semantic level. Similar challenges exist for semantic textual similarity estimation. For instance, "work into it slowly." vs. "it seems to work ." have different meanings, but the model misclassified them as "somewhat similar" (0 vs. 2.91) due to their shared words. Conversely, "in these days of googling, it's sloppy to not find the source of a quotation." vs. "i agree with kate sherwood, you should be able to attribute most quotes these days by simple fact checking." have a degree of similarity (3), but the model predicted a low similarity score (0.69), failing to capture the semantic relatedness.

These examples highlight the shortcomings of language models like BERT in understanding the nuances and complexities behind language. Compared to the small BERT model used in this study, a more advanced pre-trained model like LLAMA 3 or GPT-4 may help improve performance on these tasks by better capturing semantic nuances.

7 Conclusion

Through this project, we delved into the intricate task of jointly learning three distinct natural language processing tasks sentiment analysis, paraphrase detection, and semantic textual similarity using the BERT language model. Our findings showed the critical importance of gradient handling techniques (clipping and surgery/vaccine), SMART regularization, network structure adaptation, and sampling schedule in mitigating interference between tasks and preventing overfitting. We devised novel architectures, such as shared PALs and adaptors, to facilitate improved information flow between tasks, thereby enhancing the performance of our trained models. Furthermore, we implemented an ensemble strategy that involved selecting qualified models based on a screening criterion, which effectively compensated for the limitations imposed by task interference in a multi-task setting, ultimately boosting the ensemble predictions' accuracy. Through meticulous sensitivity analysis of hyperparameters, including gradient clipping values, SMART weights, dropout rates, learning rates, and schedules, we achieved a remarkable total score of 0.796 on the leaderboard's test set. However, we acknowledge that a more systematic exploration of these hyperparameters and architectural configurations could potentially yield further performance gains.

This intriguing project has piqued our curiosity, motivating us to explore additional fine-tuning strategies, such as early stopping for different tasks and annealing sampling. Moreover, we are keen to investigate the potential benefits of pre-training on context texts and leveraging more sophisticated language models like ChapGPT 4 and LLAMA 3. Of particular interest is understanding how these advanced models perform on challenging samples across the three tasks and how they comprehend and interpret input text compared to their BERT counterparts.

8 Ethics Statement

Our project, which focuses on jointly learning multiple natural language processing tasks using the BERT language model, raises several ethical concerns and potential societal risks. Firstly, the use of large language models like BERT carries the risk of perpetuating societal biases present in their training data. If the text corpora used for pretraining or fine-tuning contain biased language or stereotypes, the models may learn and propagate these biases, leading to unfair or discriminatory judgments in downstream tasks like sentiment analysis or paraphrase detection. Secondly, the ensemble approach employed in our project, which combines predictions from multiple models, might amplify any existing biases or errors in individual models. This could lead to compounded negative effects, especially if the ensemble is used in high-stakes decision-making scenarios.

To mitigate these potential risks, we consider the following strategies: Firstly, conduct rigorous bias testing of the language models and ensemble predictions, particularly for sensitive applications. This includes counterfactual evaluation, disparate impact analysis, or benchmark datasets designed to assess bias. Secondly, implement debiasing techniques during model training, such as adversarial debiasing, or the use of carefully curated, balanced training datasets.

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.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*, pages 45–52.

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*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

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*.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

Dymitr Ruta and Bogdan Gabrys. 2005. Classifier selection for majority voting. *Information fusion*, 6(1):63–81.

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 *International Conference on Machine Learning*, pages 5986–5995. PMLR.

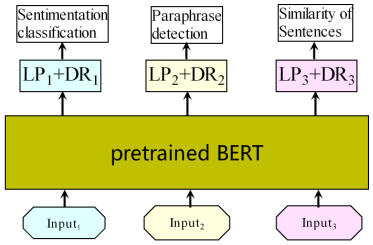
Jeremy Watt, Reza Borhani, and Aggelos K Katsaggelos. 2020. *Machine learning refined: Foundations, algorithms, and applications*. Cambridge University Press.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

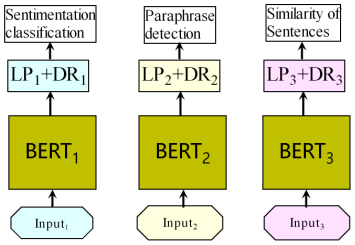
Hongzhi Zhang and M Omair Shafiq. 2024. Survey of transformers and towards ensemble learning using transformers for natural language processing. *Journal of big Data*, 11(1):25.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

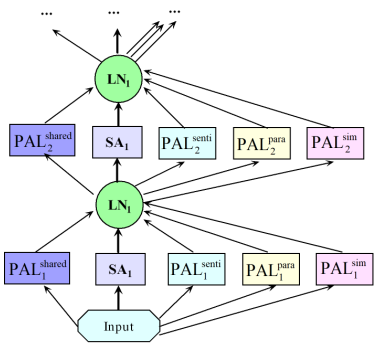
A Appendix



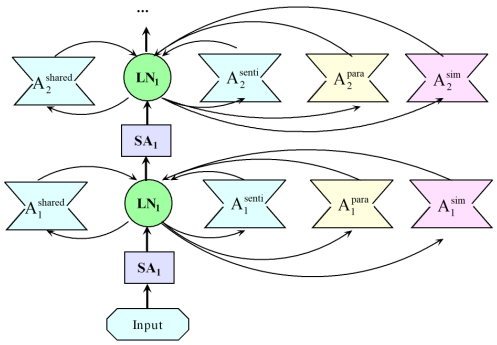
a. Baseline multitask learning



b. Individual separate modules in a multitask model



b. Shared PAL based on the baseline BERT



c. Shared Adaptor based on the baseline BERT

Figure A1: (a) Baseline BERT architecture; (b) Individual separate modules in a multitask model; (c) BERT with added shared projected attention layers (PALs) for multitask learning. (d) BERT with added shared Adaptor layers for multitask learning.

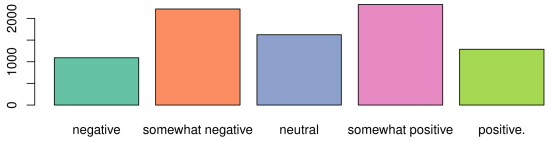


Figure A2: Class distribution for the training dataset of sentiment analysis.

Table A1: Dev Results of the Four Models Selected for Ensemble Predictions.

Model	Overall Score	SST Acc[1]	Paraphrase QQR Acc	STS Cor[2]
Ensemble predictions	0.794	0.533	0.902	0.891
Shared PAL multitask model + SMART + gradient surgery ($\alpha = 1e-5$)	0.781	0.523	0.884	0.875
Shared PAL multitask model + SMART + gradient surgery ($\alpha = 1e-5$)	0.784	0.511	0.897	0.881
PAL multitask model + SMART + gradient surgery ($\alpha = 3e-5$)	0.780	0.510	0.895	0.870
Shared Adaptor multitask model + SMART + gradient surgery ($\alpha = 1e-5$)	0.781	0.507	0.892	0.888

[1]Acc: accuracy; [2]Cor: Pearson's correlation.

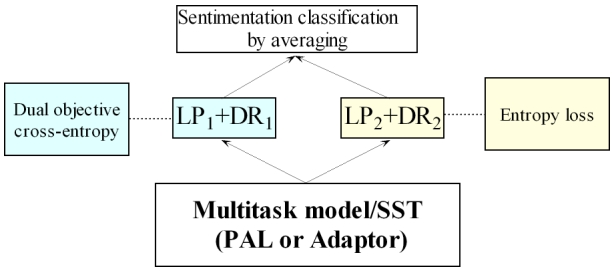


Figure A3: Ensemble prediction combining cross-entropy loss and dual cross-entropy loss objectives.

Table A2: Sensitivity analysis for clip value and dropout probability for paraphrase.

Clip value	Dev accuracy	Dropout probability	Dev accuracy
No clip	0.630	0.01	0.992
0.01	0.905	0.10	0.992
0.10	0.910	0.20	0.991
0.50	0.906	0.50	0.992
1.00	0.907	0.90	0.989