ComBERT: Improving Multitask-BERT with Combinations of Scale-invariant Optimization and Parameter-efficient Fine-tuning

Stanford CS224N Default Project

Harper Hua
shuohua@stanford.edu

Ruiquan Gao
ruiquan@stanford.edu

Xinyi(Jojo) Zhao
xyzhao99@stanford.edu

Abstract

In the rapidly evolving field of natural language processing (NLP), pre-trained language models such as BERT have established a benchmark for numerous tasks. However, the intensive computational requirements for fine-tuning these models across multiple tasks pose significant challenges. This paper introduces ComBERT, a novel framework that enhances the efficiency and efficacy of multitask BERT models by integrating pre-training optimization with parameter-efficient fine-tuning methods. Our approach leverages a multitask learning strategy, employing transfer learning and a shared loss fine-tuning mechanism. We incorporate tasks like sentiment analysis, semantic textual similarity, and paraphrase detection, focusing on their synergistic potential to foster robust generalization. Furthermore, we utilize advanced optimization techniques, including cosine similarity adjustments and Multiple Negatives Ranking Loss, alongside the innovative Low-Rank Adaptation (LoRA) and Weight-Decomposed Low-Rank Adaptation (DoRA) techniques to enhance model adaptability while minimizing resource demands. Experimental evaluations across standard datasets validate the superiority of ComBERT, demonstrating significant improvements in model performance across diverse NLP tasks while maintaining resource efficiency. This study sets a precedent for future research in developing more scalable and versatile language models.

Key Information

- TA mentor: Johnny Chang
- External collaborators / External mentor / Sharing project: No

1 Introduction

The field of natural language processing (NLP) has witnessed significant advancements in recent years, particularly with the advent of pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) [1]. These models have demonstrated remarkable performance across a wide range of NLP tasks, owing to their ability to learn rich contextual representations from vast amounts of unlabeled text data. However, the success of these models often relies on computationally intensive fine-tuning processes, which can be prohibitively expensive and time-consuming, especially when dealing with multiple tasks simultaneously.

To address these challenges, we propose ComBERT, a novel approach that combines pre-training optimization techniques with parameter-efficient fine-tuning methods to enhance the performance and efficiency of multitask BERT models. Our framework builds upon the standard BERT architecture and introduces several key innovations to improve its adaptability and scalability across diverse NLP tasks.

First, we initiate our approach by implementing and training BERT on sentiment analysis using the SST and CFIMDB datasets for single-task classification. Building upon this foundation, we explore multitask learning [9] as a key component of our framework to enable the model to learn shared representations beneficial across multiple NLP tasks. Our approach involves two principal techniques: transfer learning, which utilizes knowledge from a related task to enhance performance on a target task, and shared loss fine-tuning, which trains tasks simultaneously by integrating their loss functions. We investigate the potential of paraphrase detection as a primary task for multitask fine-tuning due to its ability to capture deep semantic similarities and transferable linguistic structures. By incorporating paraphrase detection alongside tasks such as sentiment analysis and semantic textual similarity, we aim to develop a versatile model that can generalize effectively and adapt to new challenges with minimal fine-tuning.

Second, we investigate the impact of dataset selection and augmentation on pre-training effectiveness. In addition to the commonly used datasets for paraphrase detection, we introduce the MultiNLI dataset [16], which encompasses both sentiment and paraphrase detection tasks. By generating separate datasets for each task from MultiNLI, we seek to enhance the model's understanding of diverse linguistic phenomena and improve its generalization capabilities.

Third, we explore various optimization techniques to refine the model's performance on specific tasks. For similarity prediction, we employ cosine similarity fine-tuning [14], which adjusts the model to assess paraphrase and similarity predictions based on the cosine similarity between the predicted output vector and the target vector in the embedding space. Additionally, we investigate the use of Multiple Negatives Ranking Loss Learning for the paraphrase task, which aims to capture the relative relationships between positive and negative examples more effectively.

Finally, we incorporate parameter-efficient fine-tuning (PEFT) methods, specifically focusing on the recently proposed LoRA (Low-Rank Adaptation)) [6] and DoRA (Weight-Decomposed Low-Rank Adaptation) techniques [8]. PEFT addresses the limitations of traditional fine-tuning approaches by leveraging weight decomposition to split pre-trained weights into low-rank matrices, enabling efficient adaptation while maintaining a learning capacity similar to full fine-tuning. By integrating LoRA and DoRA into our framework, we aim to strike a balance between performance and parameter efficiency, facilitating the deployment of ComBERT in resource-constrained environments.

Through extensive experiments on benchmark datasets, we demonstrate the effectiveness of ComBERT in improving the performance of multitask BERT models across three downstream tasks, . Our results highlight the synergistic benefits of combining pre-training optimization techniques with parameter-efficient fine-tuning methods, paving the way for more efficient and adaptable language models in the future.

2 Related Work

Pre-trained Language Models The development of pre-trained language models, such as BERT [1], has revolutionized the field of natural language processing (NLP). In addition to BERT, other notable pre-trained language models have been proposed. RoBERTa [10] builds upon BERT by modifying key hyperparameters and training on larger datasets, resulting in improved performance on various NLP tasks. XLNet [17] introduces a novel permutation language modeling objective, which captures bidirectional context while avoiding the limitations of traditional autoregressive models.

Multitask Learning Multitask learning [9] has emerged as a promising approach to improve the efficiency and generalization capabilities of NLP models. Raffel et al. [13] introduced T5 (Text-to-Text Transfer Transformer), a unified framework for multitask learning that treats every task as a text-to-text problem. T5 achieves state-of-the-art results on a wide range of NLP tasks, demonstrating the effectiveness of this approach. Sun et al. [15] proposed ERNIE 2.0, a continual pre-training framework that incrementally learns multiple tasks in a continual learning setting, effectively leveraging knowledge transfer across tasks.

Dataset Augmentation Gururangan et al. [3] investigated the impact of domain-adaptive pre-training, where models are further pre-trained on domain-specific datasets before fine-tuning. They found that this approach can significantly improve performance on downstream tasks in the target domain. Feng et al. [2] provided a comprehensive survey of data augmentation techniques for NLP, highlighting their effectiveness in improving model robustness and generalization.

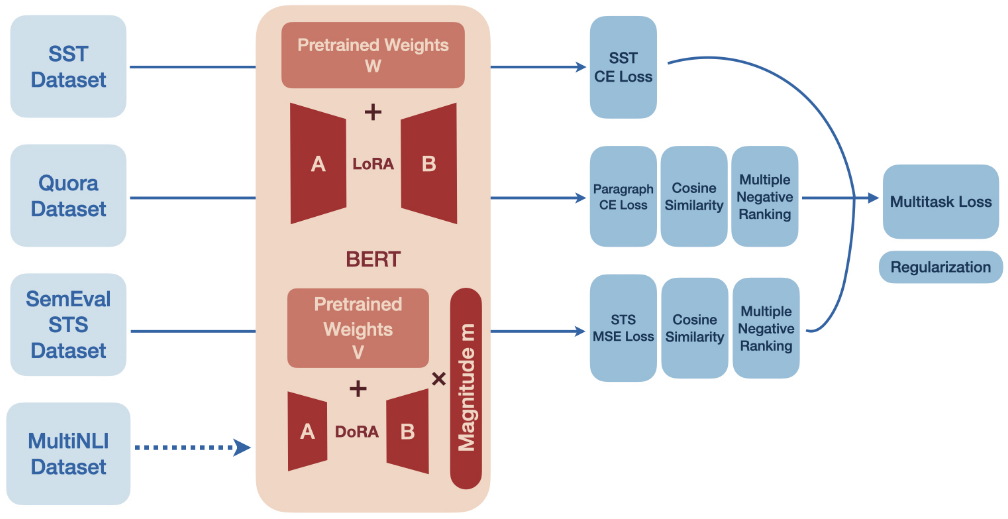

Figure 1: Architecture Overview

Optimization Techniques Mao et al. [12] proposed UniPELT, a unified framework for parameter-efficient language model tuning that combines different optimization techniques, such as adapters and prefix tuning. UniPELT achieves competitive performance while significantly reducing the number of trainable parameters. Zhang et al. [18] revisited the effectiveness of adversarial training for NLP models and proposed a new adversarial training scheme that improves model robustness and generalization.

Parameter-Efficient Fine-Tuning Hu et al. [6] proposed LoRA (Low-Rank Adaptation), which adapts pre-trained weights using low-rank matrices, enabling efficient fine-tuning while maintaining performance comparable to full fine-tuning. Liu et al. [8] further extended this idea with DoRA (Weight-Decomposed Low-Rank Adaptation), which leverages weight decomposition to split pre-trained weights into magnitude and directional components, allowing for more effective adaptation. Additionally, He et al. [4] introduced a unified view of parameter-efficient tuning methods, providing a common framework for understanding and comparing different approaches. They also proposed a novel method called UniPERT, which combines the strengths of different PEFT techniques to achieve better performance and efficiency. Mahabadi et al. [11] proposed Compacter, a compact and efficient fine-tuning method that learns a small number of task-specific parameters while keeping the pre-trained model fixed, achieving performance comparable to full fine-tuning with significantly fewer parameters.

3 Approach

3.1 Baseline Approach

We first implement MiniBert with the given code. We take it as our baseline where each task is trained using its specific training set and evaluated on a development set. This provides a foundational performance benchmark for subsequent enhancements. For our baseline, we employ distinct loss functions tailored to each specific task: cross-entropy loss for sentiment prediction, paraphrase detection loss for paraphrase prediction, and mean squared error (MSE) for similarity prediction. The basic method for computing similarity involves the dot product of two vectors, providing a straightforward yet effective baseline for comparison against more complex approaches.

3.2 Multitask Fine-tuning

Multitask fine-tuning in our framework involves two principal methods: transfer learning and shared loss fine-tuning.

Transfer learning uses knowledge from a related task (task A) to enhance performance on a target task (task B), thereby reducing the dependency on extensive labeled data for the target domain. Both tasks share the same model architecture except for the final layer. Initially, we train the model on task A to obtain a pretrained model M. Subsequently, we transfer all weights from M (except the last layer) to task B and fine-tune using dataset B. During this fine-tuning, the weights are adjusted——not frozen——to tailor the model to the specific requirements of task B. This process allows the model to adapt the broad features learned from task A to the nuances of task B, optimizing performance on the target task.

Shared loss fine-tuning involves training tasks simultaneously while integrating their loss functions. This method encourages the model to find a representation beneficial across all tasks, which is articulated mathematically as:

$$L = \alpha L_1 + \beta L_2 + \gamma L_3$$

where L represents the total loss, L_1, L_2, L_3 are the loss functions for each task, and α, β, γ are weights determining the contribution of each task's loss to the total loss.

3.3 Pretrained Dataset Exploration and Augmentation

In this study, we propose using paraphrase detection as the primary pretraining task for BERT due to its robust feature learning capabilities. This task demands deep semantic analysis, enabling the model to discern subtle linguistic nuances and develop complex semantic representations. Such sophisticated feature learning is invaluable, as it enriches the model's understanding of language, which is beneficial for a variety of subsequent NLP tasks.

Moreover, the skills acquired from paraphrase detection—such as recognizing semantic similarities and interpreting different expressions of similar ideas—are highly transferable to other NLP tasks like sentiment analysis and semantic textual similarity. By pretraining on paraphrase detection, the model not only gains a deep understanding of high-level semantic content but also ensures these capabilities are broadly applicable across diverse NLP scenarios, enhancing both its efficiency and effectiveness.

Besides this, we also find MultiNLI dataset for pretraining. It is a dataset with both sentimental and paraphrase detection. We generate two datasets from it, one is paraphrase and another is sentiment analysis. We use this to replace the pretraining section of paraphrase detection dataset(quora), to increase the understanding of the model.

3.4 Similarity Optimization

Cosine Similarity. In Cosine-Similarity Fine-Tuning[14], we adjust the model to assess the paraphrase prediction and the similarity prediction by the cosine similarity. This approach is particularly effective for tasks involving similarity assessments or matching scenarios. The cosine similarity is computed as follows:

$$\mathsf{CosineSimilarity}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\langle \mathbf{y}, \hat{\mathbf{y}} \rangle}{\|\mathbf{y}\| \cdot \|\hat{\mathbf{y}}\|} ,$$

where $\mathbf{y}, \hat{\mathbf{y}}$ are the two embedding vectors of the two sentences we want to compare. Then, we can process the cosine similarity and define the appropriate loss functions according to the tasks.

3.5 Loss Function and Regularized Optimization

Multiple Negatives Ranking Loss Learning. We use Multiple Negatives Ranking Loss Learning [5] for the paraphrase task, where we need to decide whether two sentences are paraphrases to each other. Suppose S is the similarity score we use. For a batch $(x_1, y_1, l_1), \ldots, (x_b, y_b, l_b)$, where $l_i =$ yes if x_i and y_i are paraphrases and $l_i =$ no otherwise, we compute the loss by the average of

$$-1(l_i = \text{yes}) \cdot S(x_i, y_i) + \log\left(1 + \sum_{j=1}^{b} e^{S(x_i, y_j)}\right)$$

This loss can be viewed as the cross-entropy loss on a multiple choice task – we want to select the most similar sentence of a given sentence, or claim no choice is similar.

Bregman Proximal Point Optimization. We use Bregman Proximal Point Optimization (BPPO) [7] as part of our regularization method. Consider a classification task. Suppose $f(\theta, x)$ is the output distribution of our model θ on input x, we want to update at round t by

$$\theta_{t+1} = \arg\min_{\theta} \sum_{i=1}^{b} \mathsf{Cross\text{-}Entropy}(f(\theta, x_i), y_i) + \mu \cdot \ell_s\left(f(\theta, x_i), f(\theta_t, x_i)\right) \ , \tag{1}$$

where $\ell_s(\mu, \upsilon)$ denote the symmetric KL-divergence $\mathcal{D}_{\mathrm{KL}}(\mu \| \upsilon) + \mathcal{D}_{\mathrm{KL}}(\upsilon \| \mu)$ and $\mu > 0$ is tunable parameter. The implementation of Eq. (1) is via running the optimization on mini-batches.

3.6 Parameter-efficient fine-tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) methods have been introduced to fine-tune pre-trained models with minimal additional parameters, addressing the issue of computational cost and storage requirements associated with full fine-tuning (FT). Among PEFT methods, LoRA (Low-Rank Adaptation) has gained popularity due to its simplicity and effectiveness. However, there exists a performance gap between LoRA and FT, which is often attributed to the limited number of trainable parameters in LoRA.

To bridge this gap, a novel method called DoRA (Weight-Decomposed Low-Rank Adaptation) has been proposed. DoRA leverages weight decomposition, inspired by Weight Normalization, to split the pre-trained weights into magnitude and directional components. By fine-tuning both components, DoRA aims to achieve a learning capacity similar to FT while maintaining parameter efficiency. The directional component, which contains a substantial number of parameters, is adapted using LoRA to enable efficient fine-tuning. Compared to LoRA, DoRA has demonstrated consistent performance improvements across various tasks, including natural language processing, vision-language tasks, and large language models, without compromising inference efficiency.

LoRA. LoRA (Low-Rank Adaptation) addresses the computational and memory limitations associated with updating all model weights during training. In regular fine-tuning, the weight update is defined as $W_{updated} = W + \Delta W$, where W is the original weight matrix and ΔW is the learned weight update matrix. However, computing ΔW can be expensive for large models. LoRA proposes an efficient alternative by approximating the weight update matrix ΔW as the product of two smaller matrices, A and B, such that $\Delta W \approx AB$. The updated weight matrix in LoRA is then given by $W_{updated} = W + A \cdot B$, where A and B are low-rank matrices with dimensions $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times d}$, and r is the rank of the approximation. By using low-rank matrices, LoRA significantly reduces the number of trainable parameters compared to full fine-tuning.

In the forward pass, the input x is multiplied by the low-rank matrices A and B, and the result is scaled by a hyperparameter α. The LoRA layer is then added to the original model weights, allowing for efficient fine-tuning while preserving the knowledge captured during pre-training.

DoRA. DoRA (Weight-Decomposed Low-Rank Adaptation) is an extension of the LoRA (Low-Rank Adaptation) method that aims to improve the performance of parameter-efficient fine-tuning for large language models. DoRA builds upon LoRA by decomposing the pretrained weight matrix W_0 into a magnitude vector m and a directional matrix V. The decomposition is inspired by the mathematical principle that any vector can be represented as the product of its magnitude (a scalar value indicating its length) and its direction (a unit vector indicating its orientation in space). In DoRA, this decomposition is applied to the entire pretrained weight matrix W_0, where each column vector represents the weights connecting all inputs to a particular output neuron. The DoRA process can be summarized in two steps:

1) Decompose the pretrained weight matrix W_0 into a magnitude vector m and a directional matrix V: $W_0 = m\frac{V}{|V|_c}$ where $|V|_c$ is the vector-wise norm of V.

2) Apply LoRA to the directional matrix V and learn the magnitude vector m during training: $W' = m\frac{V+\Delta V}{|V+\Delta V|_c} = m\frac{W_0+BA}{|W_0+BA|_c}$ where ΔV is the update to the directional matrix V, and BA represents the LoRA weight update.

In the forward pass, the LoRA weight update is computed and added to the pretrained weight matrix W_0. The resulting numerator is then normalized by its vector-wise L2 norm to obtain the updated directional component. Finally, the new weight matrix is obtained by multiplying the magnitude vector m with the updated directional component, and the linear transformation is applied using the new weight matrix. By decomposing the pretrained weights and applying LoRA to the directional component, DoRA aims to achieve better performance compared to standard LoRA while still maintaining parameter efficiency.

4 Experiments

4.1 Data

For Bert training part, we will use the data in the folder data. For sentiment analysis tasks, we are going to use the provided Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset. We only use the train set for training and dev set for validation.

For the fine-tune part, we use the train set and dev set of the provided SST set, Quora Dataset and SemEval STS Benchmark Dataset. The SST set has $8,544$ training data points with $1,101$ validation data points. The Quora set has $283,003$ training data points with $40,429$ validation data points. The SemEval STS Benchmark Dataset has $6,040$ training data points with 863 validation data points.

Besides, we also tried to enlarge our pretrained datasets with MultiNLI[16]. The MultiNLI dataset, also known as the Multi-Genre Natural Language Inference corpus, is a comprehensive resource designed for training and evaluating machine learning models on the task of natural language inference (NLI). Developed as an extension of the Stanford NLI (SNLI) corpus, MultiNLI introduces a diverse range of genres of spoken and written text, which were not included in SNLI. The corpus encompasses a variety of genres such as fiction, letters, telephone speech, and government reports, providing a broad spectrum of linguistic contexts.

We generate two datasets out of MultiNLI. One is sentiment analysis dataset, taking the godel_label as its label with 3 labels marked. Another is paraphrase analysis dataset, where we use all the pairs as positive samples. Then for each sentence, we find 10 random sentences as negative labels. In all we have 392,702 training data points for sentiment analysis and 3,927,020 training data points for paraphrase.

4.2 Evaluations

We will evaluate our results on dev set of the datasets and report the result of Test LeaderBoard. Specifically, For SST an Quora Dataset, the metric is accuracy. For STS benchmark Dataset, the metric is correlation. In order to explore the effects of different optimization method, we did several independent experiments for different optimization methods. Because of time limitation, we run fewer epochs to test the efficiency of different methods and combine them in our final experiments.

4.3 Results and Analysis

General Results. We utilized all the approaches mentioned in Section 3. The dev results and test results are:

	SST accuracy	PARA accuracy	STS correlation
Dev	0.529	0.891	0.871
Test	0.551	0.888	0.871

Multitask Finetuning. We present our preliminary multitask finetuning results in Table 1, where we only use $\sim 5\%$ of the huge Quora dataset. It suggests that: 1) single-task fine-tuning on each task has positive effects on the other two tasks; 2) multitask fine-tuning (almost) dominates every single-task fine-tuning – it has comparable performance on the task single-task fine-tuning uses and has significantly better performance on the other two tasks, indicating that incorporating additional related tasks during fine-tuning can effectively leverage the model's capacity to learn from diverse data, resulting in improved generalization and transfer learning capabilities; 3) while full fine-tuning is less time-efficient than last-linear-layer fine-tuning, it achieves significantly better performance. Despite the increased computational cost, fine-tuning the entire model architecture enables more comprehensive parameter optimization and adaptation, resulting in superior performance gains.

Method	α	β	γ	**SST Accuracy**	**PARA Accuracy**	**STS Correlation**
Multitask (full-model)	$1/3$	$1/3$	$1/3$	0.51	0.792	0.857
Single-task (full-model)	1	0	0	0.506	0.577	0.449
Single-task (full-model)	0	1	0	0.446	0.806	0.635
Single-task (full-model)	0	0	1	0.480	0.741	0.861
Last-linear-layer	-	-	-	0.461	0.380	0.260

Table 1: Our results on *dev* sets. We obtain the SST accuracy for the 3rd and 4th lines by running the "last-linear-layer" fine-tuning after the "full-model" fine-tuning. Note that we don't introduce new parameters in the last layer for the PARA and STS tasks.

Pretrained Dataset Selection and Augmentation. We present our pretrained-selection results in Table 2. Our analysis suggests that the PARA Dataset consistently performs well across all tasks, demonstrating strong generalization capabilities. In contrast, other datasets such as SST and STS show optimal performance only on tasks directly related to their respective datasets. Due to its broad applicability, the PARA Dataset is an excellent choice for initial pretraining stages. MultiNLI Dataset can slightly increase the results, but not as good as PARA Dataset.

Pretraining Dataset	**SST acc**	**PARA acc**	**STS corr**
SST Dataset	0.513	0.629	0.861
PARA Dataset	0.502	0.803	0.863
STS Dataset	0.495	0.623	0.851
MultiNLI Dataset	0.515	0.557	0.864

Table 2: Our results on *dev* sets. We train the model on the pretrained dataset for around 3 epochs and use multitask training for the 3 downstream tasks.

Loss Function Optimization. We show the results for loss function optimization in Table 3. Multiple Negative Ranking loss learning will improve the model's ability to distinguish the similarity amongst sentences, which makes it a better objective for similarity-related tasks on PARA and STS datasets. With additional fine-tuning, the PARA accuracy increases 0.002 while the STS correlation increases 0.01 under multiple negative ranking loss. This marginal improvement could be caused by the severe overfitting issue of the pretrained model. Furthermore, we observed that introducing regularization for accuracy evaluation tasks on SST and PARA datasets could boost the performance to some extent, which reinforces our aforementioned explanation of the overfitting problem caused by the pretrained model.

Method	**SST acc**	**PARA acc**	**STS corr**
Loss Function Benchmark	-	0.864	0.856
Multiple Negative ranking loss learning	-	0.866	0.866
No regularization	0.514	0.864	-
Regularized Optimization	0.524	0.880	-

Table 3: The results of Loss Optimization and Regularized Optimization (on the *dev* set) with additional pre-training. We did not know how to apply multiple negatives ranking loss function to the SST task, while we did not implement BPPO method to the STS task.

Cosine Similarity Optimization. We implemented both simple inner dot product similarity and cosine similarity. Comparing the results we find that cosine similarity outperforms inner dot product similarity because the latter fails to take relative similarity values of the vectors into consideration. Moreover, cosine similarity measures the cosine value of the angle between two vectors, offering a better representation for relative distances by measuring magnitude and orientations of vectors simultaneously.

PEFT with LoRA and DoRA. As shown in Table 4, we performed two sets of experiments for LoRA and DoRA respectively. First, we replaced linear layers for Bert model with $rank = 4$ LoRA and DoRA layers. We loaded different baseline checkpoints (pre-trained with single task) and used multitask learning (MTL) loss as the fine-tuning objective. Compared to Table 1 where we used the exact same dataset, we observed that DoRA significantly outperforms LoRA and has comparable performance as the multitask fine-tuning full-model. This resonates withthe novelty of DoRA to boost the fine-tuning perfomance while maintaining the advantage of parameter efficiency by introducing the additional weight decomposition vectors.

Second, we compared the results under different tunable parameter settings. It is worth metioning that we used MTL loss for fine-tuning and loaded checkpoints pre-trained with MTL loss as well. Summarized in Table 4, LoRA and DoRA have similar results on three different tasks in general while DoRA marginally outperforms LoRA, which could be caused by the severe overfitting issues of the pretrained checkpoint with MTL loss, limiting the space for further improvement.

PEFT w/ MTL loss & loaded checkpoints	**#parameters**	**SST acc**	**PARA acc**	**STS corr**
LoRA + SST-base	3,849	0.527	0.524	0.436
LoRA + cfimdb-base	3,849	0.423	0.530	0.309
DoRA + SST-base	757,286	0.496	0.748	0.845
DoRA + cfimdb-base	757,286	0.496	0.753	0.853
Tunable parameters	**#parameters**	**SST acc**	**PARA acc**	**STS corr**
LoRA only	3,849	0.515	0.804	0.853
all paras	109,489,938	0.509	0.818	0.857
DoRA only	757,286	0.514	0.817	0.855
all paras	110,243,375	0.511	0.817	0.856

Table 4: Results of PEFT performance comparison. We set rank as 4 for both LoRA and DoRA linear layer replacement. First, we loaded different baseline checkpoints and tuned on LoRA/DoRA layers only. Second, we loaded MTL full fine-tuned checkpoints with and experimented on different frozen weights.

5 Conclusion

In this project, we introduced ComBERT, a novel framework that combines pre-training optimization techniques with parameter-efficient fine-tuning methods for enhanced multitask BERT models. Through extensive experiments, we demonstrated the effectiveness of ComBERT across three downstream tasks: sentiment analysis, semantic textual similarity, and paraphrase detection. Our results showcased the synergistic benefits of integrating multitask learning, transfer learning, dataset augmentation strategies, loss function optimization, similarity function optimization, regularized optimization, and parameter-efficient fine-tuning methods like LoRA and DoRA.

Notably, ComBERT has achieved the best performance on the combined results of these tasks, outperforming traditional fine-tuning approaches. By leveraging shared representations and efficient adaptation, ComBERT exhibits improved generalization capabilities while maintaining computational efficiency, making it well-suited for resource-constrained environments. These findings pave the way for more efficient and versatile language models in the field of natural language processing.

6 Ethical Consideration

Deploying our model in real-world scenarios presents several potential challenges. Firstly, the model has been trained and tested only on specific datasets, which may not accurately represent the broader real-world data distribution. This discrepancy can lead to inaccurate predictions. Additionally, the training datasets may not include sufficient representation of diverse data points, potentially leading to biased outcomes.

Secondly, the theoretical explanation of the BERT model remain inadequately explained. As a result, providing a precise confidence level for the model's predictions is challenging. This lack of certainty could lead to the generation of unreliable or incorrect data, which might adversely affect decision-making processes.

To mitigate these issues, it is crucial to continuously update and evaluate the model against a more varied and comprehensive dataset that better mirrors real-world conditions. Moreover, efforts should be made to enhance the transparency and interpretability of the model, thereby providing clearer insights into how decisions are made and fostering greater trust in its applications.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. 2021.

[3] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. 2018.

[4] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. 2022.

[5] Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652, 2017.

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[7] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 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, ACL 2020, Online, July 5-10, 2020*, pages 2177–2190. Association for Computational Linguistics, 2020.

[8] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

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

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019.

[11] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *arXiv preprint arXiv:2106.04647*, 2021.

[12] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. 2022.

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 2023.

[14] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[15] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. 2019.

[16] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[17] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. 2020.

[18] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. 2021.