Adapt BERT on Multiple Downstream Tasks

Stanford CS224N Default Project

Ran Li
Department of Computer Science
Stanford University
`ranl3@stanford.edu`

Abstract

In this research, we explore how to adapt BERT on different downstream tasks, including sentiment analysis (sentence classification), paraphrase detection (sentence-pair classification) and semantic textual similarity (sentence-pair regression). We experiment multiple architectures and methodologies to optimize the multi-task performance: multi-task BERT fine-tuning with mixed loss, task-specific heads with MEAN pooling, SMART regularization, and adaptive training for imbalanced datasets of different tasks. We evaluate the model on Stanford Sentiment Treebank (SST) dataset, Quora Dataset and SemEval STS Benchmark Dataset, and show how the combination of these methodologies can improve the model performance.

1 Key Information to include

- Mentor: Josh Singh
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

In recent years, pre-trained language models based on Transformers architecture, such as BERT (Bidirectional Encoder Representations from Transformers, Devlin et al. (2018)), have revolutionized the field of natural language processing by achieving the state-of-art performance on wide range of tasks. BERT's success is largely attributed to the bi-drectional representation with deep networks, which allows it to capture context from both directions in one sentence. Despite of its impressive capabilities, adapting BERT to multiple downstream tasks simultaneously has significant challenges. Each task may have different requirements and characteristics, making it hard to produce a general solution.

Multitask Learning offers a promising approach to leverage both the shared knowledge from the pre-trained model backbone and the task-specific heads. In our research, we explore the adaptation of BERT for three distinct downstream tasks: sentence classification (like sentiment analysis), sentence-pair classification (like paraphrase detection), and sentence-pair regression (like semantic textual similarity). Our approach involves fine-tuning BERT with task-specific heads, cosine-similarity architecture design ((Reimers and Gurevych, 2019)), incorporating pooling mechanisms, and implementing training strategies to handle imbalanced datasets. Additionally, we introduce SMART regularization mechanism ((Jiang et al., 2020)) to enhance model robustness and reduce overfitting.

We demonstrate the effectiveness of our methods through extensive experiments on Stanford Sentiment Treebank (SST), Quora Dataset, and SemEval STS Benchmark Dataset, showcasing improved performance across all tasks.

3 Related Work

Pre-trained language models have evolved different architecture paradigms over the past years. BERT is the encoder-only model, T5 (Raffel et al., 2020) is the encoder-decoder model, and GPT (Radford and Narasimhan, 2018) is the encoder-only model. These architectures offer diverse advantages on different NLP tasks, and all of them demonstrates the power of tranformer-based architectures.

The application of BERT has been widely studied in recent literature. Subsequent researches focus on enhancing BERT capabilities through various extensions and adaptions. RoBERTa (Liu et al., 2019) scales BERT with longer training and more data. ALBERT (Lan et al., 2019) reduces memory consumption and increases training speed. DistilBERT (Sanh et al., 2019) offers a smaller and faster BERT version with distillation techniques.

Multitask Learning has been widely explored as a means to generalize downstream tasks with shared knowledge. Multitask Fine-Tuning has been further refined through many innovative work. Stickland and Murray (2019) introduced Projected Attention Layers (PAL) for different task adaptation. Yu et al. (2020) proposed Gradient Surgery, a technique to project the gradients of tasks and avoid conflicting tasks' gradients.

The Sentence-BERT model (Reimers and Gurevych, 2019) demonstrates the effectiveness using siamese and triplet network for sentence-pair tasks that can be compared using cosine-similarity. Our work builds upon these insights by experimenting with pooling strategies, exploring the siamese and triplet network, and adopting cosine-similarity. Sentence-BERT has been particularity effective on semantic textual similarity tasks.

Regularization techniques, such as SMART (Jiang et al., 2020), have proven effective in mitigating the overfitting and enhancing model robustness. SMART applies adversarial perturbations during the training process to encourage smoother loss.

4 Approach

Multitask learning usually involves a shared backbone network, followed by task-specific heads that make predictions for each task. In our research, we use Google's open source BERT model **bert-base-uncased** (Devlin et al., 2018) with 110M parameters as the backbone, and develop task-specific heads. We experimented different Pooling Mechanisms. We have also created adaptive training cycles to handle imbalanced training data, and adopted SMART regularization to avoid overfitting. The overall Multi-Task BERT Architecture is shown in Figure 1.

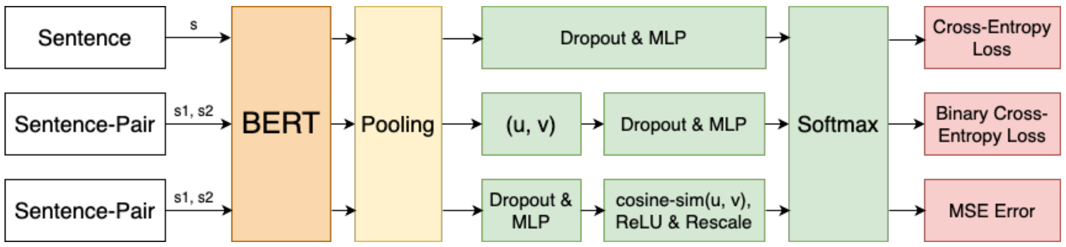


Figure 1: Multi-Task BERT Architecture

4.1 Task-specific Heads, Loss Functions and Pooling

We build different task-specific heads to learn embeddings from BERT models.

Sentence Classification (SST Dataset): We first apply a dropout on BERT embeddings of the input sentence, and then a hidden layer $\mathbb{L}$ with size $\mathbb{R}^{h\times h}$, a ReLU activation function, and another linear layer $\mathbb{L}$ with size $\mathbb{R}^{h\times c}$. c is the number of classes, and for SST Dataset, $c = 3$.

Sentence-Pair Classification (Quota Dataset): For each sentence-pair s_1 and s_2, we first apply a dropout on their BERT embeddings u and v individually. We use siamese network structure and concatenate them into (u, v). Then, use a hidden layer $\mathbb{L}$ with size $\mathbb{R}^{2h\times h}$, a ReLU activation function, and another linear layer $\mathbb{L}$ with size $\mathbb{R}^{h\times 1}$.

Note that, for this task, we have also experimented with the triplet network structures from Reimers and Gurevych (2019), which concatenate the embeddings into $(u, v, |u - v|)$. However, from our experiments, it has worse performance than the siamese structure. So, our model keeps to use (u, v) and diverges from their conclusion.

Sentence-Pair Regression (STS Dataset): For each sentence-pair s_1 and s_2, we predict their BERT embeddings u and v individually. Then, we use a hidden layer $\mathbb{L}$ with size $\mathbb{R}^{h \times h}$ for each of them, and compute the cosine similarity between the output. Because cosine similarity has range[-1, 1], we add another ReLU activation function to convert the range to [0, 1], and then linearly scale it to [0, 5].

Note that, the architecture of this task is inspired by SentenceBERT from Reimers and Gurevych (2019).

4.1.1 Loss Functions

We use Cross-Entropy Loss for Sentence Classification task (SST), Binary Cross-Entropy Loss for binary sentence-pair classification task (Quota), and Mean Squared Error (MSE) for sentence-pair regression task (STS).

4.1.2 Pooling Mechanism

Pooling is crucial to process the embeddings from BERT model. We explored two pooling mechanisms: CLS pooling and Mean Pooling. For all tasks, Mean Pooling has significantly better results than CLS pooling.

- **CLS Pooling**: The embedding of BERT [CLS] token (the first token in the sequence) is often used as a summary representation of the entire sequence.
- **Mean Pooling**: Mean pooling is the average the embeddings of all tokens in the sequence. This approach considers information from all tokens, potentially providing a more comprehensive representation.

4.2 Multi-Task Fine-Tuning

Beyond training the last task-specific layers, we also perform the BERT model full fine-tuning. We use one BERT model as the shared backbone for multiple downstream tasks.

Typically, the combined loss function for multi-task architecture is:

$$\mathcal{L} = \sum_{i=1}^{n} \lambda_i \mathcal{L}_i$$

In our research, we treat each task loss equally with $\lambda_i = 1$ as we found their loss have in similar scales by our tests:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$$

4.3 Training Strategy for Imbalance Data

Our training data for each task is very imbalanced. Quora Dataset has 40 times data as SST Dataset, and 50 times data as STS Dataset. Therefore, we need to use iterative cycles, so that we can aggregate losses from three tasks within each iteration.

One training strategy to mix training data during each iteration is the **Round Robin (RR)** approach. This method involves sequentially sampling a batch of training data from each task in turn, and then sum up the losses of them. However, this approach can lead to overfitting for tasks with significantly smaller datasets (SST, STS) due to excessive training iterations.

An alternative training strategy is the **Adaptive Round Robin (ARR)** approach. In this method, we begin by fine-tuning the model with the Quora dataset for r epochs to leverage its larger size for initial learning. Afterwards, we switch to the round-robin strategy to balance the training and learn across all tasks. This approach aims to mitigate small-dataset overfitting by utilizing dataset size to dynamically decide the training process.

4.4 SMART Regularization

Aggressive fine-tuning can overfit training data. Proposed by Jiang et al. (2020), the SMART (Smoothness-inducing Adversarial Regularization and Bregman Proximal Point Optimization) regularization framework is designed to enhance the robustness and efficiency of fine-tuning pre-trained language models. It is effectively the same as solving the following optimization, where $\mathcal{L}(\theta)$ is the loss function, λ_s is hyperparameter:

$$\min_{\theta} F(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta)$$

And $\mathcal{R}_s(\theta)$ is the smoothness-inducing adversarial regularizer:

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

This optimization can be solved by Bregman Proximal Point Optimization. In our code implementation, we modified the original PyTorch implementation of SMART from Jiang et al. (2020) to adapt to sentence-pair use cases for Quota and STS Datasets.

4.5 Baseline

Our baseline is to only train last layers (i.e. task-specific heads) with CLS pooling.

4.6 Additional Notes

We explored to integrate **Gradient Surgery** (Yu et al., 2020) to align the gradients from different tasks and mitigate conflicts. However, we encountered out-of-memory errors and did not continue.

5 Experiments

5.1 Data

The following section describes three datasets used in our research.

The **Stanford Sentiment Treebank (SST)** Socher et al. (2013) dataset contains 11,855 single sentences from movie reviews with phrases annotated for sentiment analysis task (sentence classification).

The **Quora Dataset** consists of 404,298 question pairs labeled to indicate if they are paraphrases of each other. It is used for paraphrase detection task (sentence-pair classification).

The **SemEval STS Benchmark Dataset** Agirre et al. (2013) includes 8,628 sentence pairs with similarity scores ranging from 0 (unrelated) to 5 (equivalent meaning). It is used for semantic textual analysis task (sentence-pair regression).

Dataset	**Train Examples**	**Dev Examples**	**Test Examples**
Stanford Sentiment Treebank (SST)	8,544	1,101	2,210
Quora Dataset	283,010	40,429	80,859
SemEval STS Benchmark Dataset	6,040	863	1,725

Table 1: Summary of datasets and their respective splits.

5.2 Evaluation method

We use the following evaluation metrics for each task:

5.3 Experimental details

For general hyperparameters, we use learning rate as 10^{-5} for full model finetuning and 10^{-3} for last layers only (task-specific heads) finetuning, batch size as 8, maximum number of epochs as 10.

Dataset	Domain	Metrics
SST	Sentence Classification	Accuracy
Quota	Sentence-Pair Classification	Accuracy
STS	Sentence-Pair Regression	Pearson Correlation Coefficient

Table 2: Summary of tasks, domains, and metrics.

For SMART regularization hyperparameters, we use mostly the same set of choices as the original paper ((Jiang et al., 2020)): the perturbation size $\epsilon = 10^{-6}$ and $\sigma = 10^{-5}$. We set $\mu = 1$, total iterations $T = 1$, regularization factor $\lambda_s = 1$. The learning rate η is set to 10^{-3}.

We use AdamW optimizer, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay as 0.01.

We run training on Nvidia T4. To speed up training, we implemented Mixed Precision Training (Micikevicius et al., 2017). It decreases around 50% training time.

5.4 Results

5.4.1 Test Leaderboard

Model	Final Score	SST Acc.	Paraphrase Acc.	STS Corr.
Full-Finetuning + SMART($\lambda_s = 0.02$)	0.759	0.509	0.858	0.821

Table 3: Best model performance on SST, Paraphrase, and STS test dataset and leaderboard.

Above is our best model performance on Test Leaderboard. In this report, we use a single BERT model as the shared backbone, rather than fine-tuning different BERT models for each task and make predictions individually. The latter will have better scores, but it is against the essence of Multitask BERT models.

In the following sections, we report different experiments we explored and our analysis on the results.

5.4.2 Pooling Mechanism

Model	Final Score	SST Acc.	Paraphrase Acc.	STS Corr.
Last Layers + CLS	0.562	0.385	0.677	0.246
Last Layers + Mean Pooling	**0.701**	**0.460**	**0.776**	**0.733**

Table 4: Comparison on CLS Pooling and Mean Pooling.

The results in Table 4 highlight the superiority of Mean Pooling over CLS Pooling across various tasks. Mean Pooling achieved a significantly higher final score (0.701) compared to CLS Pooling (0.562). Specifically, Mean Pooling led to improvements in all three tasks: SST accuracy (0.460 vs. 0.385), paraphrase detection accuracy (0.776 vs. 0.677), and STS correlation (0.733 vs. 0.246).

These findings suggest that Mean Pooling, which averages embeddings from all tokens, provides a more comprehensive representation of the input sequences, enhancing the model's ability to generalize and perform well on different tasks. In contrast, CLS Pooling, which relies on the [CLS] token alone, may miss important contextual details. Therefore, Mean Pooling should be considered the preferred strategy.

5.4.3 SMART Regularization

Model	Final Score	SST Acc.	Paraphrase Acc.	STS Corr.
Full-Finetuning	0.763	0.504	0.862	0.849
Full-Finetuning + SMART($\lambda_s = 0.02$)	**0.770**	**0.528**	**0.862**	**0.841**
Full-Finetuning + SMART($\lambda_s = 0.1$)	0.761	0.514	0.861	0.817
Full-Finetuning + SMART($\lambda_s = 1.0$)	0.762	0.519	0.843	0.846

Table 5: Comparison on with and w/o SMART Regularization using different weight λ_s

The experiments of Full-Finetuning with different SMART regularization mechanisms are reported in Table 5. We use the weight hyper-parameter of SMART Regularization $\lambda_s \in [0.02, 0.1, 1.0]$.

The original paper (Jiang et al., 2020) mentioned there are only slight differences in model performance when $\lambda_s \in [1, 10]$. Regularization is unreasonably strong when $\lambda_s \geq 100$, and unreasonably weak when $\lambda_s \leq 0.1$. However, from our experiments, we found that using $\lambda_s = 0.02$ surprisingly has the best performance. Using a higher λ_s (0.1 and 1.0) did not result in significant improvements in the final score compared to not using SMART Regularization.

This discrepancy between our findings and the original paper presents that choosing the weight parameter λ_s for SMART Regularization is challenging.

Task Sensitivity: It is also worth to mention that different tasks exhibit varying sensitivity to adversarial perturbations. Sentence classification (SST) benefits more from SMART regularization than Sentence-pair classification and regression tasks (Paraphrase and STS).

Computational Overhead: SMART regularization introduces additional computational overhead due to the generation of adversarial examples. From experiments, we noticed the training time increases roughly from 2 hours to 6 hours when adopting SMART regularization, which means using SMART regulization triples the training time.

5.4.4 Training Strategy

Model	**Final** Score	**SST Acc.**	**Paraphrase Acc.**	**STS Corr.**
Full-Finetuning with RR	**0.763**	**0.504**	**0.862**	**0.849**
Full-Finetuning + ARR ($r = 1$)	0.753	0.503	0.843	0.825
Full-Finetuning + ARR ($r = 2$)	0.754	0.499	0.863	0.800

Table 6: Comparison on with and w/o Adaptive Round Robin (ARR) training strategy

In Table 6, we compared the performance of different training strategies, Round Robin and Adaptive Round Robin with different initial epoch size r.

The results demonstrate that the Full-Finetuning with Round Robin achieved the highest final score. The ARR strategy, aimed at mitigating overfitting for the SST and STS datasets, did not show the expected improvements. Instead, there are noticeable drops for STS correlation for both $r = 1$ and $r = 2$.

These findings suggest that the ARR training strategy did not outperform the simpler RR strategy. There are a few possible reasons:

- In ARR, the intial focus on the larger Quota dataset may cause the model to overly specialized to that dataset. When switching to smaller datasets for SST and STS, the model may not adapt well due to the strong initial bias towards the Quota dataset.
- Additionally, it might be an evidence for the gradient conflicts from different tasks, especially that the model is fine-tuned on tasks sequentially rather than simultaneously.
- The learning rates might be sub-optimital for ARR. The sudden shifts between tasks might require dynamic adjustments of learning rates.

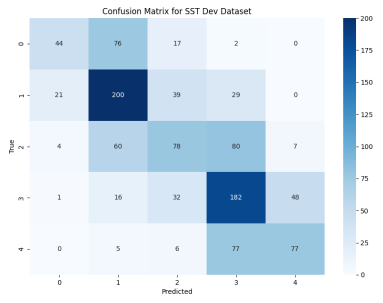
6 Analysis

6.1 SST Dataset

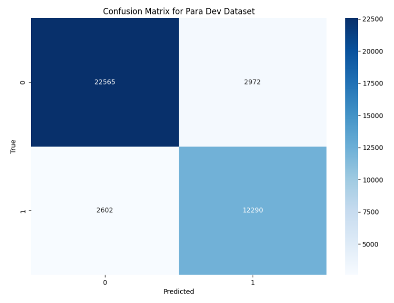
Confusion Matrix for SST dataset is shown in Figure 2a. For the mis-classifications in off-diagonal values, most of then are between adjacent classes (e.g., +1 or -1). This indicates that the model occasionally confuses adjacent sentiment classes, which is understandable given the subjective nature of sentiment analysis.

6.2 Paraphrase Dataset

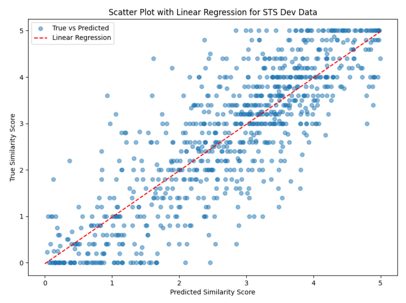
The Confusion Matrix for Paraphrase dataset is shown in Figure 2b. Additionally, we can compute the precision, recall and F1 score in 7. We observe a balanced F1 score, which is essential for overall model robustness. It suggests that the model is not overly biased towards either precision or recall.



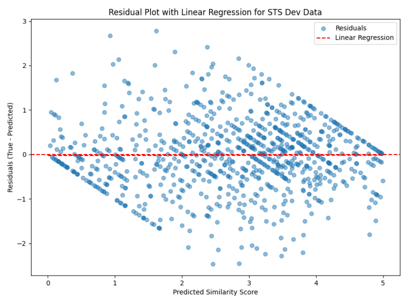
(a) Confusion Matrix for SST Dev Dataset



(b) Confusion Matrix for Para Dev Dataset



(c) STS Scatter Plot with Linear Regression



(d) STS Residual Plot with Linear Regression

Figure 2: Model Performance Analysis

Metric	**Precision**	**Recall**	**Accuracy**	**F1 Score**
Score	0.805	0.825	0.862	0.815

Table 7: Performance Metrics for Paraphrase Detection on Dev Dataset

6.3 STS Dataset

For STS dataset regression task, the Scatter Plot and Residual Plot with Linear Regressions are presented in Figure 2c and 2d. For the Scatter Plot, most of the points lie closet to the diagonal line. There is noticeable spread around the diagonal, especially at the higher and lower ends of the similarity scores. This suggests that while the model performs well on average, it may struggle at extreme values. From the Residual Plot, we observe that the residuals are centered around zero, indicating there is no systematic bias in predictions.

7 Conclusion

In this research, we explored the adaptation of BERT for multiple downstream tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity. Our approach involved multi-task fine-tuning, employing task-specific heads, and integrating innovative techniques such as Pooling mechanisms, SMART regularization and adaptive training strategies to enhance model robustness.

Our key findings include:

- **Multi-Task Learning Effectiveness**: The multi-task learning approach with a shared BERT backbone and task-specific heads proved effective in leveraging shared representations while allowing specialization for individual tasks. Fine-tuning both the BERT model and the last layers demonstrates substantial performance improvements across all tasks.
- **Pooling Strategies**: Mean Pooling consistently outperformed CLS Pooling across all tasks. Mean Pooling, with the average for all token embeddings in one sentence, can provide a more comprehensive representation of the semantic meaning of input sequences.

- **SMART Regularization**: Integrating SMART regulization with a carefully chosen weight parameter ($\lambda_s = 0.02$) can significantly enhance model robustness. However, having a higher value of λ_s, unexpectedly, did not show further benefits, and in some cases, adversely affected the performance.
- **Training Strategies**: The Round Robin (RR) training strategy emerged as a more effective method. In contrast, the Adaptive Round Robin (ARR) strategy did not yield the expected improvements on imbalanced training datasets, and in some cases, led to lower performance metrics.

Our research provides valuable experiments and insights into the challenges and strategies for adapting pre-trained language models to multiple downstream tasks. Our findings underscore the importance of balanced training strategies, the potential of adversial SMART regulization to enhance model robustness, and the superiority of Mean Pooling over CLS Pooling. These contributions are beneficial for developing more effective multi-task learning models for future studies.

7.0.1 Limitations and Future Work

While our research achieved some notable improvements, several limitations remain that we plan to address in future work:

- **Regularization**: The SMART regularization require further investigation to fully understand their potentials. Developing a more effective method for hyperparameter tuning, such as optimizing λ_s, is necessary.
- **Training Strategies**: Given that ARR did not mitigate the overfitting issue as expected, there is a need to investigate more effective training strategies for imbalanced datasets.
- **Reproducibility**: Additional experiments are needed to verify whether the observed results are consistently reproducible or if they are influenced by random initialization effects.
- **Extended Datasets**: Extended evaluations on diverse datasets and tasks are important to validate the generalizability of our findings.
- **Dynamic Learning Rates**: We currently use the same learning rate as 10^{-5} for full fine-tuning on both the BERT backbone and the task-specific heads. Future work should explore the use of higher learning rates for the task-specific heads to enhance performance.

8 Ethics Statement

Here we analyze two potential ethical challenges and societal risks.

1. Bias in Language Models and Training Data: Just like many pre-trained language models, BERT model is known to inherit and potentially increase the biases presented in the training data. These biases can include gender, race and social classes, which may lead to unfair representation when applied to downstream tasks. For instance, studies have shown that BERT models can reflect gender biases and produce different sentiment valuations for male and female sample versions (Jentzsch and Turan, 2022). Additionally, BERT has been found to display ethnic biases, which can vary across different languages (Ahn and Oh, 2021).

One mitigation strategy is to introduce better data representation during model training, such as data augmentation for underrepresented groups, perform better data cleaning to turn biased data into neutral data or re-weighting of training samples. In Ahn and Oh (2021), it leverages on the multilingual training of M-BERT to counterbalance biases presented in individual monolingual BERT models. This can help to reduce ethnic bias with the integration over multiple language texts.

2. Misuse of Models: Another potential risk is the misuse of models. For example, paraphrase detection models can be used to create deceptive content, while sentiment analysis model can be used to censor internet comments or manipulate public opinions.

To prevent this issue, it is crucial to set strict model usage guidelines and policy and limit the acceptable applications of these models.

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.

Jaimeen Ahn and Alice Oh. 2021. Mitigating language-dependent ethnic bias in BERT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 533–549, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*.

Sophie Jentzsch and Cigdem Turan. 2022. Gender bias in BERT - measuring and analysing biases through sentiment rating in a realistic downstream classification task. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 184–199, Seattle, Washington. Association for Computational Linguistics.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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.

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.