Too SMART for your own good: Multitask Fine-Tuning pre-trained minBERT through Regularized Optimization and hyper-parameter optimization

Stanford CS224N Default Project

Proud Mpala
Department of Computer Science
Stanford University
pmpala@stanford.edu

Wayne Chinganga
Department of Computer Science
Stanford University
waynecc@stanford.edu

Abstract

This paper presents an approach to fine-tuning pre-trained minBERT models through regularized optimization techniques, specifically focusing on the use of Smoothness-inducing Adversarial Regularization and Bregman Proximal point optimization (SMART), Multitask loss, and Hyper-parameter fine tuning. The primary goal is to address the issue of over fitting that commonly occurs during the transfer learning process when pre-trained language models are fine-tuned on smaller, domain-specific datasets. By implementing these regularization techniques, we aimed to achieve improved generalization and robustness in fine-tuned models. However, our experimental results exhibited signs of over-regularization despite achieving some "smoothness" in training. We provide insights into the performance of our SMART model and discuss potential avenues for future research. Notably, our findings highlight the sensitivity of various tasks to different learning rates and the challenges in balancing regularization to achieve optimal model performance. We also discuss the ethical considerations surrounding bias, fairness, and safety in fine-tuning language models, emphasizing the need for careful monitoring and mitigation strategies throughout the training process.

1 Key Information to include

- TA mentor: Sonia Chu
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Training language models often requires a large amount of labeled data and this is often expensive to achieve. A work-around that researchers have come up with is to take resources from a domain that has abundant resources and train language models in that domain and then do further training on limited data in the domain that we care aboutDodge et al. (2020); this is referred to as transfer learning. Existing large pre-trained language models include embeddings from Language Models(ELMo), Generative Pre-trained Transformer(GPT), Bidirectional Encoder Representations from Transformers(BERT) and Text-To-Text Transfer Transformer(T5). These models can have as much as 11 billion parameters(T5), about 110 million for original BERT. Given this vast number of parameters, the extremely high complexity means that in transfer learning the model is highly susceptible to

over-fitting. There are several methods trying to get around over-fitting namely: heuristic learning rate, freeze/unfreeze layers or adding additional parameters which are fewer than the original ones and only tuning the new parameters. The paper claims that these require significantly more effort to achieve and proposes a different technique called *Smoothness-inducing Adversarial Regularization and Bregman Proximal point optimization* (SMART) that reduces over-fitting and requires less effort than the aforementioned alternatives. In most cases, we want to apply these fine-tuned models not only to one tasks but to multiple related tasks. In many cases, we want to apply these fine-tuned models not just to one task but to multiple related tasks. To address this, multitask learning can be employed, which involves optimizing the model to perform well across various tasks simultaneously. A critical component of multitask learning is effectively balancing the losses from each task. In our work, we experimented with several strategies for combining multitask losses to enhance model performance. Hyper-parameter optimization plays a crucial role in fine-tuning language models. We utilized a combination of grid search and random search methods to find the optimal hyper-parameters. Initially, grid search allowed us to systematically explore a predefined set of hyper-parameters, ensuring thorough coverage of the potential parameter space.

3 Related Work

Several explorations are undertaken as part of a larger research initiative on ways to optimally and efficiently fine-tune pre-trained models. Aggressive fine-tuning of pre-trained models can lead to over fitting of the downstream examples due to the relatively large number of parameters that are being tuned. The resultant over-fitting phenomenon is not only limited to language models but even the simplest logistic regression models can exhibit over-fitting when the number of parameters is larger than the training data. This scenario usually result in models that are highly complex and as has been concluded in various settings, our findings also conclude that a simpler models generalizes well on unseen data. The SMART framework explored in our work generalize to other machine learning tasks like computer vision which also have a vast number of parameters and thus prone to over-fit during fine-tuning. We also explore various strategies for combining multitask losses to improve the model's generalization capabilities. This is tied to the area of multi-task fine-tuning and more generally, multi-task training. Researchers are always looking to find better generalizations, this is still gaining momentum, a good example is the training of models for tasks related to coding. Liu et al. Hyper-parameter optimization plays a crucial role in fine-tuning pre-trained models effectively. Various methods such as grid search, random search, and more advanced techniques like Bayesian optimization are employed to identify the optimal set of hyper-parameters. These methods help in exploring the parameter space efficiently to find configurations that enhance model performance and generalization. In our work, we utilized a combination of grid search and random search methods to find the optimal hyper-parameters. **?**. Our approach of integrating hyper parameter optimization with multitask loss strategies is designed to address the specific challenges of fine-tuning large pre-trained models, such as susceptibility to over fitting and the need for efficient training processes. This combination not only enhances model performance on individual tasks but also contributes to the broader field of NLP and AI by providing insights into better optimization practices for pre-trained models.

4 Approach

We first implemented the default minBERT model from the default project handout. We then extended model based on the following main improvements: 1) Multitask Fine-tuning: we added the MultiTask layer for multitask classification(used inspiration from multiplicative attention in dealing with sentence pairs), 2) Smoothness Inducing Regularization, Bregman Proximal Point Optimization which are based on the SMART framework Jiang et al. (2020). We extensively explored hyper-parameter fine tuning mainly applying the grid-search approach in order to cover the diverse configurations of our model. Yu and Zhu (2020)

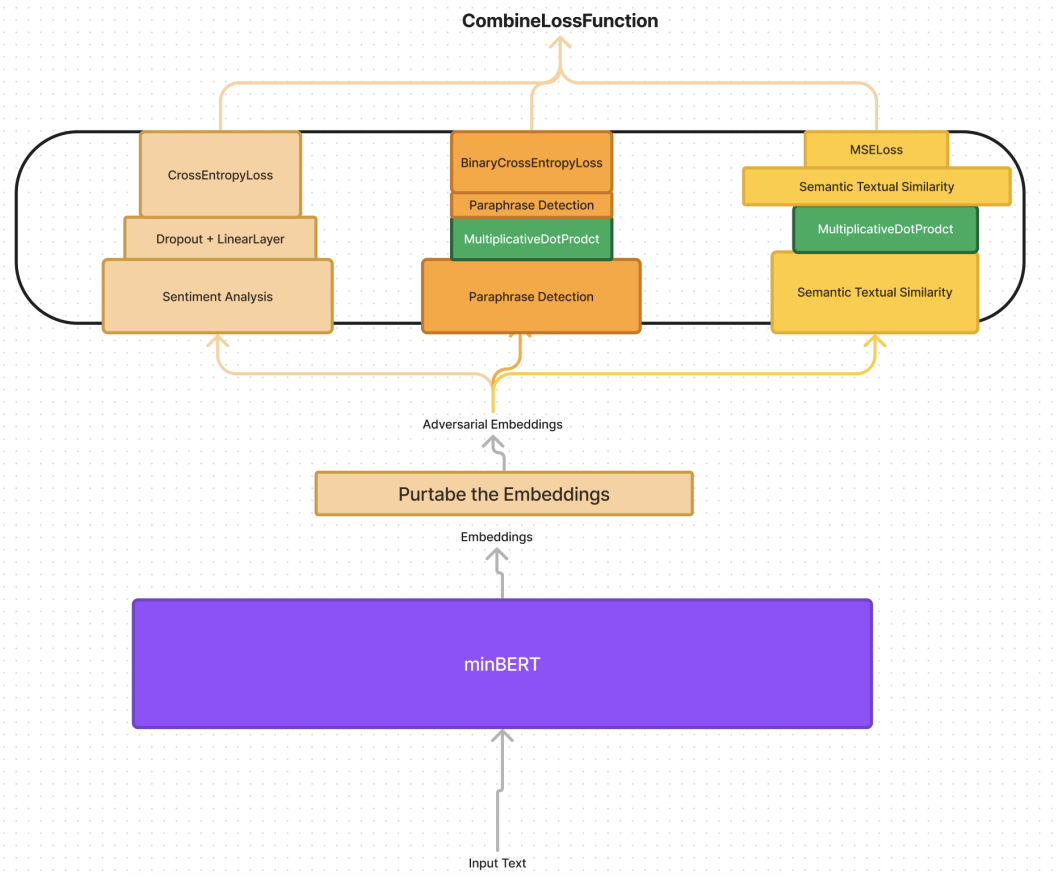

Figure 1: Architecture for our minBERT + SMART implementation

Multitask Fine-tuning: Instead of fine-tuning minBERT on individual tasks, we made use of multi-task learning to update BERT.

$$\mathcal{L}_{total} = \textbf{CombinationLossFunction}(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$$

In this project we explore 4 variants of **CombinationLossFunction**:

- *Addition*: This is the simplest method where we sum the loss functions of all tasks.

$$\mathcal{L}_{total} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \tag{1}$$

- *Dynamic Loss Balancing*: This method adjusts the weights dynamically during training based on the magnitude of the losses. This ensures that no single loss dominates the training process. The total loss is computed as:

$$\mathcal{L}_{total} = w_1\mathcal{L}_1 + w_2\mathcal{L}_2 + w_3\mathcal{L}_3$$

  where

$$w_i = \frac{\mathcal{L}_{total}}{3\mathcal{L}_i}$$

  for $i \in \{1, 2, 3\}$.

- *Automatic Loss Weighting*: This method uses learnable parameters for each loss weight, allowing the model to automatically balance the losses during training. The total loss is computed as:

$$\mathcal{L}_{total} = w_1\mathcal{L}_1 + w_2\mathcal{L}_2 + w_3\mathcal{L}_3$$

  where w_1, w_2, and w_3 are learnable parameters.

- *Uncertainty-based Weighting Loss*: This approach uses uncertainty as a measure to weight each loss, as suggested in "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics" Kendall et al. (2018). The total loss is computed as:

$$\mathcal{L}_{total} = \frac{1}{2\sigma_1^2}\mathcal{L}_1 + \frac{1}{2\sigma_2^2}\mathcal{L}_2 + \frac{1}{2\sigma_3^2}\mathcal{L}_3 + \log\sigma_1 + \log\sigma_2 + \log\sigma_3$$

  where σ_1, σ_2, and σ_3 represent the uncertainty associated with each task.

Smoothness Inducing Regularization: This is a form of adversarial training which instead of generating adversarial examples and training on them for a smoother model it virtualizes the process and has the same effect as adversarial training. Given the model $f(\cdot;\theta)$ and n data points $\{x_i, y_i\}$ where x_i is the embedding of the input sentences and y_i is the label of the associated task. The overall loss function is:

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

where $\mathcal{L}(\theta) = \frac{1}{n}\sum \ell(f(x_i;\theta), y_i)$ and $\ell(\cdot,\cdot)$ is the loss function of given task and λ_s is the tuning parameters. To induce smoothness, we define the regularization term

$$\mathcal{R}_s(\theta) = \frac{1}{n}\sum \mathtt{max}_{||\tilde{x_i}-x_i||\leq\epsilon}\, \ell_s(f(\tilde{x_i},\theta), f(x_i,\theta)$$

where ϵ is a tuning parameter and ℓ_s is chosen as symmetric KL divergence for models that output probability distributions and a square loss for scalar producing model.

Breggman Point Optimization: This momentum accelerated optimization method is proposed to offer stable updates by penalization aggressive updates at each iteration. This ensures that fine-tuning does not result in parameters that are further from the pre-trained weights. Let $f(\cdot;\theta)$ denote the weights from pre-trained model.

$$\theta_{t+1} = \mathtt{argmin}_\theta \mathcal{F}(\theta) + \mu\mathcal{D}_{Breg}(\theta, \theta_t)$$

where μ is a tuning parameter and the Bregman divergence:

$$\mathcal{D}_{Breg}(\theta, \theta_t) = \frac{1}{n}\sum \ell(f(x_i;\theta), f(x_i;\tilde{\theta}_t))$$

for exponentially moving average $\tilde{\theta}_t = (1-\mathcal{B})\theta_t + \mathcal{B}\tilde{\theta}_{t-1}$.

MultiplicativeDotProduct: This is inspired by multiplicative attention. Given two vectors u and v. We perform multiplicative dot product by inserting weights in-between them to perform the operation $u^{\mathrm{T}}Wv$; the matrix W can be learned.

5 Experiments

5.1 Data

Quora Dataset which consists of 404,298 question pairs with labels indicating whether particular instances are paraphrases of one another. The data is provided in the following splits: train (283,010 examples), dev (40,429 examples), test (80,859 examples). SemEval STS Benchmark Dataset which consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). The provided data is in the following splits: train (6,040 examples), dev (863 examples), test (1,725 examples).

5.2 Evaluation method

To evaluate the models we used accuracy for Paraphrase Detection task on the Quora Dataset and on the Sentiment Analysis task on the SST Dataset. For Semantic Textual Analysis task on the STS Dataset we used Pearson Correlation of the true similarity values against the predicted values. To put all these scores together into a multi-task score we used the formular:

$$\text{MultiTaskScore} = \frac{\mathtt{sst\_acc} + \frac{\mathtt{sts\_corr}+1}{2}}{3}$$

Chatterjee

5.3 Experimental details

To improve on the base minBERT model, we first investigated how best to train on the 3 given downstream tasks. We explored 3 various approaches, sequential training, zipped datasets training(will default to the smallest) and cycle zipped datasets(default to length of the largest) and repeat the smaller ones. We faced some implementation challenges with our SMART implementation that used up most of the time that would have been otherwise devoted to running experiments. To mitigate this, we introduced shrinking of the training datasets for faster training using a shrinking factor M. This essentially selects $\frac{1}{M}$ random examples from each training dataset and then uses those for training. Having this hyper-parameter M helped us to balance between being fast and also being accurate in terms of the training datasets.

Our first experiment was to determine the best layers to have on top of the pre-trained BERT model for the three tasks that we have. We evaluated 3 different layers: linear layer on sentiment plus simple dot product on embeddings, multiplicative dot product on paired tasks, we did this for both fine tuning last layer and full model. Then we investigated if we were still experiencing the over fitting phenomenon even with the shrinking of the training sets. Then we investigated what loss function to use between Simple Add, Dynamic Loss Balancing, Automatic Loss Weighting, Uncertainty-based Weighting Loss.

To optimize our model's hyper-parameters, we employed a combination of grid search and random search methods. Initially, we used grid search to systematically explore a predefined set of hyper-parameters, ensuring thorough coverage of the potential parameter space. Towards the end of our experimentation, we switched to random search to explore a broader range of hyper-parameter values more efficiently. This dual approach allowed us to balance between thorough exploration and computational efficiency.

We used ADAM as our optimizer with learning rate of 1×10^{-5}. A batch size of 8. The maximum number of epochs was set to 5. We used a dropout rate of 0.1. For SMART, we set the purtubation size $\epsilon = 10^{-5}$ and $\sigma = 10^{-5}$. We set $\eta = 10^{-3}$. We use $\beta = 0.99$ for the first 10% of the updates $t \leq 0.T$ and $\beta = 0.999$ for the rest of the training. *Our SMART models use these configurations unless stated otherwise*.

5.4 Results

Table 1: Results from using different batch iterator structures

Training Setting	**Fine-Tune Mode**	**Best Dev Accuracy**
Sequential Training	Last Linear Layer	0.413
Sequential Training	Full Model	0.371
Zipped Datasets without Cycling	Last Linear Layer	0.408
Zipped Datasets without Cycling	Full Model	0.430
Zipped Datasets with Cycling	Last Linear Layer	0.421
Zipped Datasets with Cycling	**Full Model**	**0.525**

Table 2: Choosing the last layer to use for Paraphrase Detection and Semantic Textual Similarity

Model	**M**	**Dev Accuracy**
Base Model	50	0.400
	100	0.403
Multiplicative Attention Dot Product	50	0.472
	100	**0.511**

Table 3: Performance of Various Loss Structures

Fine-Tune Mode	M	Multi-Task Loss	Dev Accuracy
Full-Model	100	Simple Add	0.523
	100	Dynamic Loss Balancing	0.508
	100	Combined Loss	0.506
	100	Uncertainty Loss	0.506
Full-Model	50	Simple Add	**0.528**
	50	Dynamic Loss Balancing	0.473
	50	Combined Loss	0.457
	50	Uncertainty Loss	0.457

6 Analysis

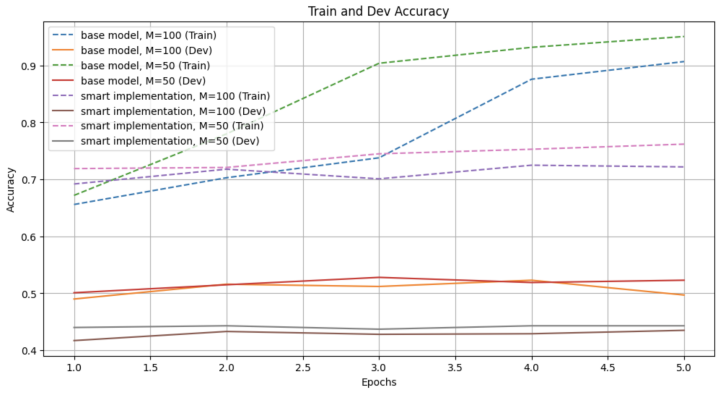

Figure 2: This figure shows the time series of training accuracy for the `train` and `dev` datasets for the Base Model against our SMART model. The scores are based on 1

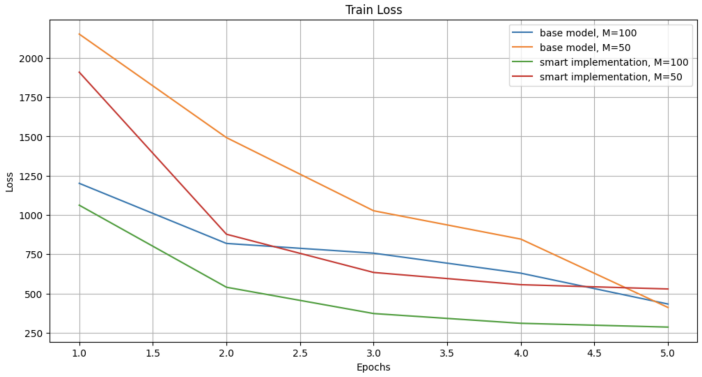

Figure 3: This figure shows the time series of training loss for the `train` and `dev` datasets for the Base Model against our SMART model

Our SMART model seems incapacitated in terms of expressive power. Though the training loss is decreases for all the configurations, the non-SMART model seems to have a sharper decrease in loss. We noted that the training with SMART, though constrained in terms of gradient, is relatively more stable. We suspect this would be the smoothness induced by the Bregman Proximal Point optimization.

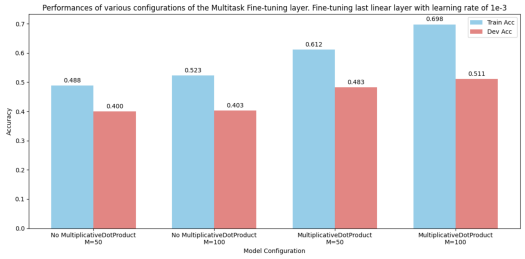

Figure 4: The figure shows the performance in terms of best accuracy for Base Model with **no** MultiplicativeDotProduct(direct dot product of embeddings) and with MultiplicativeDotProduct

We tried different values of $\eta = \{0.001, 0.1, 1, 10, 100\}$ and these did not change the train or dev accuracy. This is likely because were were using clamping as the projection Π for the expression $\tilde{x_i} \leftarrow \Pi_{||\tilde{x_i} - x_i||_\infty \leq \epsilon}(\tilde{x_i} + \eta \tilde{g_i})$Jiang et al. (2019). For this reason its possible that the updates were being clamped in all the experiments resulting in very similar adversarial examples $\tilde{x_i}$s thus giving the same outcome.

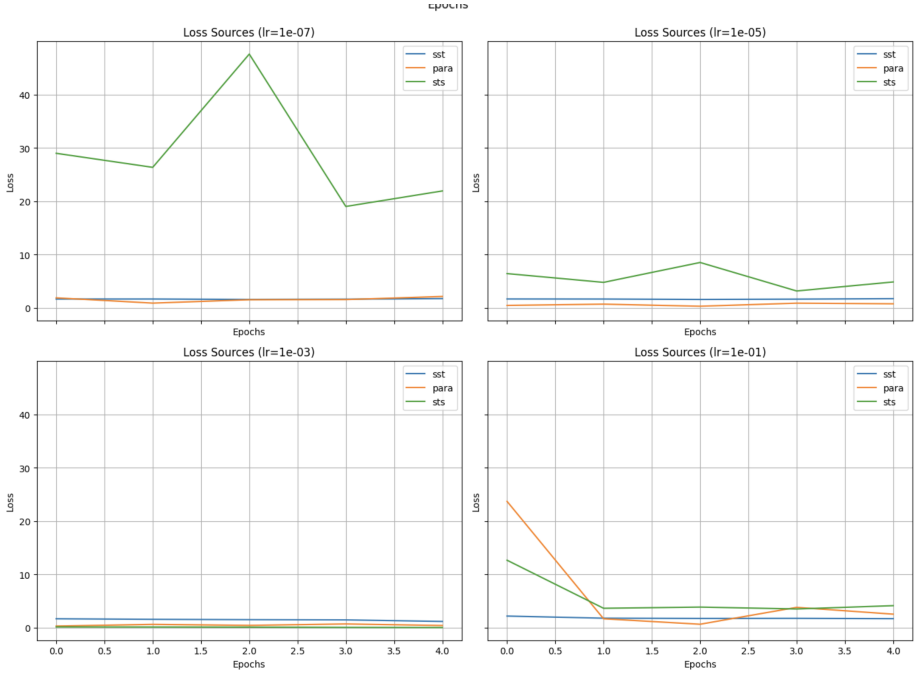

Figure 5: Figure showing the relative sources of the training loss across different learning rates on the SMART implementation default smart parameters except for the learning rate and M = 100

Across the different learning rates, the learning rate $1e-5$ and $1e-3$ seem to be stable under the given configurations of other hyper-parameters. Its also interesting to note that Semantic Textual Similarity task is more sensitive to very low learning rates and the Paraphrase Detection task is more sensitive to high learning rate. The Sentiment Analysis task remains stable across various learning rates.

7 Conclusion

In conclusion, our study presents an approach to fine-tuning pre-trained minBERT models, leveraging Smoothness-inducing Adversarial Regularization and Bregman Proximal point optimization (SMART), multitask learning, and hyper-parameter optimization. Despite encountering challenges

such as over-regularization, our experiments demonstrate the potential of SMART in inducing smoothness during training, albeit with some limitations on model expressiveness. Our findings underscore the importance of dynamically balancing losses in multitask learning and systematically exploring hyper-parameter configurations for optimal model performance. While there is room for improvement, particularly in refining the SMART framework and addressing implementation challenges, our work provides valuable insights into optimizing fine-tuning processes for pre-trained language models. Future research should focus on refining regularization techniques, exploring novel multitask learning strategies, and developing efficient hyper-parameter optimization methods tailored for large-scale language models.

8 Ethics Statement

Bias and fairness: Large pre-trained models like minBERT models may inherit biases present in the training data, leading to biased predictions in sentiment analysis, paraphrase detection, and semantic textual similarity tasks. These biases could manifest themselves in word embeddings and trickle down to downstream tasks and perpetuate stereotypes or discrimination, especially if the datasets used for fine-tuning contain biased or skewed samples. Compromised Safety during fine-tuning: Pre-trained models sometimes have guardrails that control for strong language and potentially harmful content. During fine-tuning, there is a chance that the safety alignments can be compromised especially if done aggressively. Qi et al. (2022) Several studies have shown that adversarial training partially deals with bias of the model output. By extension since SMART is a variant of adversarial training, it potentially has the capacity to deal with bias due to reducing sensitivity in perturbations of the input. Han and Baldwin (2024)

Targeted manipulations in fine-tuning: Careful manipulations during fine-tuning can sometimes correct for biases.Wang and Russakovsky (2022). This can be done through minor manipulations to the proportion of the dataset from underrepresented subcategories. Regular safety audits: To mitigate harmful content from the language model, one can conduct regular safety audits throughout the fine-tuning process to evaluate the model's performance and identify any instances of compromised safety. These audits can involve both automated checks for safety violations and manual inspections by domain experts to ensure compliance with safety standards.

References

S. Chatterjee. Edsterm. ed discussion. `https://edstem.org/us/courses/57406/discussion/4976790`. Accessed: 2024-05-19.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Xudong Han and Timothy Baldwin. 2024. Diverse adversaries for mitigating bias in training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Melbourne, Australia. Association for Computational Linguistics. School of Computing and Information Systems, The University of Melbourne, Victoria 3010, Australia.

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:1911.03437v5*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190. Association for Computational Linguistics.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bingchang Liu, Chaoyu Chen, Cong Liao, Zi Gong, Huan Wang, Zhichao Lei, Ming Liang, Dajun Chen, Min Shen, Hailian Zhou, Hang Yu, and Jianguo Li. Mftcoder: Boosting code llms with multitask fine-tuning. *arXiv preprint arXiv:2304.12345*.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2022. Fine-tuning aligned language models compromises safety, even when users do not intend to! *Princeton University*. Lead Authors: Xiangyu Qi, Yi Zeng, Tinghao Xie, Equal Advising: Prateek Mittal, Peter Henderson.

Angelina Wang and Olga Russakovsky. 2022. Overwriting pretrained bias with finetuning data. *Princeton University*.

Tong Yu and Hong Zhu. 2020. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*.