Implementing RO-BERT from Scratch: A BERT Model Fine-Tuned through Regularized Optimization for Improved Performance on Sentence-Level Downstream Tasks

Stanford CS224N Default Project

Cat Fergesen
Department of Symbolic Systems
Stanford University
catf@stanford.edu

Clarisse Hokia
Department of Biomedical Data Science
Stanford University
chokia@stanford.edu

Abstract

Transfer learning is a powerful tool for natural language processing that applies the robustness and complexity of a model trained on a large dataset to downstream tasks in smaller target domains. However, NLP frameworks that utilize transfer learning may have a high risk of overfitting due to the incongruous sizes of the pre-training and fine-tuning datasets. Many research groups have proposed methods for improving the resilience of the original BERT model to overfitting. We focus specifically on the SMART framework, which was proposed by Jiang et al. (2020) and includes smoothness-inducing adversarial regularization and momentum Bregman proximal point optimization (MBPP). We also implement our own slanted triangle learning rate method for selecting ideal hyper-parameters, taking inspiration from the slanted learning rate method proposed by Howard and Ruder (2018). Though packages for the SMART framework exists, we choose to implement it from scratch to test the replicability of Jiang et al. (2020)'s published work. We also tailor our slanted triangular learning rate method for our specific context. Though we attained some promising results, our implementation does not attain the levels of success reported by Jiang et al. (2020) or Howard and Ruder (2018). Our findings suggest the difficulty of implementing these adjustments in practice and highlight the need for increased accessibility in the field of machine learning research in order for research insights to extend their benefit to the real world.

1 Key Information

- Mentor: Rohan Taori
- Team Contributions: Cat worked primarily on the baseline SMART and MBPP extensions. Clarisse worked primarily on the baseline vanilla implementation and slanted triangular learning rate extensions. However, both members of the group improved upon each others' baselines to achieve the reported results.

2 Introduction

Transfer learning is a process where a model that has been trained for one task with a large amount of data is fine-tuned to perform a related task with a smaller target domain. Large language models like ELMo (Peters et al. (2018)), GPT (Brown et al. (2020)), and BERT (Devlin et al. (2019)) are often selected for pre-training, because their power in capturing linguistic information can be useful for downstream tasks. They are also compatible with cheaply available unlabeled data sourced

from the internet. A common framework for implementing transfer learning includes replacing the task-specific layer in the more complex model and then training the model on downstream tasks using the dataset from the target domain. Transfer learning was first suggested in the context of NMT by Vaswani et al. (2017) and later led to the development of BERT by Devlin et al. (2019).

In spite of the strengths of transfer learning, however, overfitting is a significant issue during the second stage of the transfer learning process, because the data from the target domain is so limited and the pre-trained model is highly complex. Many researchers have used hyper-parameter tuning heuristics to respond to overfitting and improve the generalizability of the model, such as using a heuristic learning rate schedule, gradually unfreezing layers or only adapting certain layers and freezing others, and adding layers (Howard and Ruder (2018), Peters et al. (2018), Houlsby et al. (2019), Stickland and Murray (2019)). However, these methods demand significant tuning and are therefore unsatisfying solutions to the problem on their own.

One promising adjustment that has been shown to improve a model's resilience against overfitting is SMART, or S**M**oothness-inducing **A**dversarial **R**egularization and B**R**egman p**R**oximal poin**T** op**T**imization, which was suggested by Jiang et al. (2020). The SMART framework is used to fine-tune the language model that is used in the pre-training stage of the transfer learning process. SMART includes (1) smoothness-inducing adversarial regularization, which deals with the high complexity of the large language model to prevent overfitting during the pre-training stage, and (2) momentum Bregman proximal point optimization, which penalizes the model for aggressive updating in the fine-turning stage which may cause overfitting. Smoothness-inducing adversarial regularization prevents the output of the model from changing even when small perturbations are injected. Performing regularization in this way prevents the output of the model from changing much within the neighborhoods of all the training points, thus preventing overfitting at the stage in the process of transfer learning that modifies the model. Meanwhile, the momentum Bregman proximal point optimization (MBPP) method accelerates the vanilla Bregman proximal point method by introducing the momentum parameter. MBPP was selected by Jiang et al. (2020) due to its state-of-the-art performance in other learning benchmarks. Also called the "mean-teacher" method, the MBPP method prevents an iteration from deviating significantly from the previous iteration. Jiang et al. (2020) argue that the MBPP method allows the model to "effectively retain" the pre-trained model's knowledge of the out-of-domain data.

We implement the SMART framework from scratch in order to test the replicability of Jiang et al. (2020)'s work. Since hyper-parameter selection can greatly impact model performance, we also implement our own slanted triangle learning rate method, inspired by the work of Howard and Ruder (2018). Like the mehtod proposed by Howard and Ruder (2018), our method increases and decays the learning rate according to an update schedule that is informed by the model's performance. Jiang et al. (2020) and Howard and Ruder (2018) both report high performance of their models. Jiang et al. (2020) find that $\text{SMART}_{\texttt{BERT}}$ performs better than BERT across all 8 GLUE tasks by over 1%. $\text{SMART}_{\texttt{BERT}}$ also has results that are comparable to T5, which Jiang et al. (2020) note has 11 billion parameters in comparison to $\text{SMART}_{\texttt{BERT}}$'s 356 million. Howard and Ruder (2018) find that their slanted triangular learning rate method outperforms other common learning rate schedules on large datasets. Our implementation of the suggested adjustments show improvements upon the baseline, but in spite of a comprehensive approach, our implementation fails to attain the success reported by Jiang et al. (2020) and Howard and Ruder (2018). Therefore, we maintain that Jiang et al. (2020) and Howard and Ruder (2018) present promising solutions to overfitting at a high level, while also arguing that their work is not adequately reproducible to a level that would benefit real-world implementations of improved transfer learning frameworks.

3 Related Work

We combine the work of Jiang et al. (2020) and Howard and Ruder (2018) to create a framework that intends to improve the resilience of pre-trained language models against overfitting. Jiang et al. (2020)'s SMART framework is based on the work of Miyato et al. (2018), Zhang et al. (2019), and Shu et al. (2018), who used regularization methods similar to SMART for other applications. Jiang et al. (2020)'s SMART contributes to the wider research context by presenting the first application of these methods to pre-trained language models. Howard and Ruder (2018) present the slanted

[2]See Jiang et al. (2020) for full citations.

triangular learning rate method as an extension of the triangular learning rates method developed by Smith (2017) that includes a shorter increase period and longer decay period. Improvements to transfer learning like those proposed by Jiang et al. (2020) and Howard and Ruder (2018) are valuable to the NLP community because in a landscape of limited data, models must become more resilient to overfitting in order to generalize more accurately to unseen data points and ultimately provide better representations of language for a variety of NLP tasks.

4 Approach

In addition to implementing a baseline BERT model that uses the Adam optimizer for sentiment classification and multitask classification, as outlined in the default project handout, we also implement the SMART framework and our own slanted triangular learning rate method from scratch. Other than the provided source code, all our code is our own. We will proceed by outlining our approach to each sub-element of our framework.

1. **Vanilla minBERT.** We implemented a baseline minBERT model as outlined in the default project handout. To ensure that we had a robust baseline on which to implement our two extensions, we developed a vanilla model, which included (1) adding 2-3 linear layers to the pooler outputs of the baseline minBERT model; (2) experimenting with utilizing ReLU, softmax, and sigmoid computations in the downstream tasks; and (3) finetuning on each of the three downstream tasks.

2. **Smoothness-inducing adversarial regularization.** We implemented the smoothness-inducing adversarial regularization and the momentum Bregman proximal point method by integrating the algorithm provided in the default project handout with Jiang et al. (2020)'s Algorithm 1 (see Appendix). Jiang et al. (2020) define the smoothness-inducing adversarial regularizer as $\mathcal{R}(\theta)$ as follows:

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

where $f(x;\theta)$ as a model associated with the parameter θ that maps input sentences x to an output space. Consider n points of the target task such that $\{(x_i, y_i)\}_{i=1}^{n}$ and x_i refers to the embedding of the input sentences from the first embedding layer of BERT. Let y_i refers to the associated labels. $\lambda_s > 0$ and $\epsilon > 0$ are tuning parameters. ℓ_s is the loss function depending on the target task, which is chosen to be symmetrized KL-divergence. ℓ_s is defined as follows:

$$\ell_s(P, Q) = \mathcal{D}_{\text{KL}}(P\|Q) + \mathcal{D}_{\text{KL}}(Q\|P),$$

where $\mathcal{D}_{KL}(P||Q)$ is the KL-divergence of two discrete distributions P and Q such that $\mathcal{D}_{KL}(P||Q) = \sum_k p_k log(p_k/q_k)$. The associated parameter for P is p_k and the associated parameter for Q is q_k.

Smoothness-inducing adversarial regularization prevents the output of the model f from changing even when a small perturbation is injected to x_i by minimizing the optimization defined as follows:

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

where $\mathcal{L}(\theta)$ is the loss function defined as:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i)$$

[3] See Howard and Ruder (2018) for full citations.

3. **Momentum Bregman proximal point optimization (MBPP).** As previously stated, our implementation of the MBPP method is based on Jiang et al. (2020)'s Algorithm 1. Given a model $f(\cdot;\theta)$ and n data points $\{(x_i, y_i)\}_{i=1}^n$, we compute

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

where $\mathcal{D}_{\text{Breg}}(\theta, \tilde{\theta}_t)$ is the Bregman divergence, described by the equation below,

$$\mathcal{D}_{\text{Breg}}(\theta, \tilde{\theta}_t) = \frac{1}{n} \sum_{i=1}^{n} \ell_s(f(x_i;\theta), f(x_i, \theta_t)) \tag{2}$$

and where $\mu > 0$ is a tuning parameter and $\tilde{\theta}_t = (1-\beta)\theta_t + \beta\tilde{\theta}_{t-1}$.

4. **Slanted triangular learning rate.** We implement our own version of the slanted triangular learning rate method based on the work of Howard and Ruder (2018).

   Since the multitask classifier allows the model to train on different learning rates, our implementation of the slanted triangular learning rate method uses the passed-in learning rate as the initial learning rate then steeply increases the learning rate linearly to thrice the learning rate in the first $20-25\%$ of batches then gradually decreases the learning rate linearly back down to the initial learning rate, as shown in the equation below:

$$new\_lr = \begin{cases} \frac{lr}{100}(iter\_num) + lr & \text{if iter\_num} \leq 200 \\ -\frac{lr}{300}(iter\_num) + (3.5 \times lr) & \text{otherwise} \end{cases} \tag{3}$$

5 Experiments

5.1 Data

We are using the pre-processed datasets specified in the default project handout and provided via the default project GitHub repository. The provided datasets are the Stanford Sentiment Treebank (SST) dataset (11,855 examples), the CFIMDB dataset (2434 examples), a subset of the Quora dataset (202,152 examples), and the SemEval STS Benchmark dataset (8628 examples). Sentiment classification labels are predicted using BERT embeddings of the SST and CFIMDB datasets, paraphrase detection is predicted using BERT embeddings of the Quora dataset subset, and sentence similarity is predicted using BERT embeddings of the SemEval STS Benchmark dataset.

5.2 Evaluation method

As described in the project handout, we evaluated the accuracy of our results using the provided model_eval_sst() and model_eval_multitask() functions in the evaluation.py file. The model_eval_sst() and model_eval_multitask() functions are used to evaluate accuracies or correlation at the end of each epoch for the train and dev sets. We made small adjustments to these functions when manipulating the return values of the predicting functions, particularly for our implementation of the SMART framework. The dev accuracies and correlation were noted for each version of the sentiment classifier and multitask classifier BERT implementations to ensure we develop a strong baseline with which to test extensions on. The accuracies and correlations are compared in the results section below.

5.3 Experimental details

The following five model configurations were implemented: (1) Baseline, which is an implementation of the multitask classifier as outlined in the default project handout that was finetuned only on sentiment classification; (2) Vanilla, which is an extension of the Baseline model that is finetuned on all three downstream tasks; (3) T-Vanilla, which is an extension of the Vanilla model that includes an implementation of our slanted triangular learning rate method; (4) T-Bregman, which is an extension of T-Vanilla that includes an implementation of the MBPP method; and (5) T-SMART, which is

an extension of T-Bregman that includes an implementation of smoothness-inducing adversarial regularization.

Each of the 4 extensions of the Baseline model were pretrained with a learning rate of $1e-3$ and a number of epochs $\in [2, 6, 10]$ and then finetuned with a learning rate of $1e-5$ and a number of epochs $\in [2, 6, 10]$. For one pretrain epoch, training time on all three datasets took approximately 2 minutes total, and for one finetune epoch, training time on all three datasets took approximately 18 minutes total. Our implementation of our slanted triangular learning rate method use the learning rate parameter passed to the model to calculate the range of learning rates that the model will use. Additional learning rates were experimented with, though the learning rates listed consistently resulted in optimal performance.

5.4 Results

	Accuracy and Pearson Correlation Scores from Test Leaderboard
SST test accuracy	0.502
Paraphrase test accuracy	0.674
STS test correlation	0.374

Figure 1 demonstrates the mean pre-training dev accuracies for the five different model configurations, and **Figure 2** demonstrates the mean finetuning dev accuracies for the five different model configurations. We see that the extensions have improved accuracy of the paraphrase classification and textual similarity classification tasks in comparison with the Baseline model, but all four extensions perform more poorly in comparison with the Baseline model for the sentiment task. T-Bregman performs the most poorly of all the extensions, which is expected, as T-Bregman includes the MBPP method without the smoothness-inducing adversarial regularization method that Jiang et al. (2020) intend for MBPP to work with in tandem. T-SMART has similar but slightly less optimal performance as compared to the Vanilla and T-Vanilla models, which may point to errors in our from-scratch implementation of the highly complex algorithm as proposed by Jiang et al. (2020).

6 Analysis

Our five-model system intends to demonstrate how adjustments to the transfer learning process can build upon one another and work in tandem as part of a transfer learning framework. The Baseline model shows some success, but it is limited; by finetuning the model on three downstream tasks, the Vanilla model improves upon the generalizability of the Baseline model and completes the transfer learning process. Our extensions, the slanted triangular learning rate method and SMART, intend to further improve upon the Vanilla model in order to make the model more resilient against overfitting. Our slanted triangular learning rate method acknowledges the key role that hyper-parameters play in the performance of a model and adjusts the learning rate according to the model's performance. On the dev set, we see that the slanted triangular learning rate method (as demonstrated by T-Vanilla) improves the accuracy of the Vanilla model. However, we see that the slanted triangular learning rate method cannot make up for a model framework that is in itself lacking; for example, the T-Bregman model, which adds the MBPP method to the T-Vanilla model, performs poorly despite integrating two extensions. This can be explained by the fact that the MBPP method performs best in coordination with the SMART method, and therefore, we would not expect to see significant increases in performance between T-Vanilla and T-Bregman.

Acknowledging the intended harmony of the MBPP method and smoothness-inducing adversarial regularization, we would expect to see increased performance from the T-SMART model, which includes the slanted triangular learning rate and the SMART framework, in comparison to the four other models. However, T-SMART performs either similarly to or worse than the three other extension models and performs more poorly than the Baseline model on the paraphrase detection and textual similarity tasks. We credit this performance to the difficulty of implementing the SMART method from scratch. Though Jiang et al. (2020) present a reasonable, high level description of their method, their work would be more informative for other researchers in the field of natural language processing who already have experience with building transfer learning frameworks and

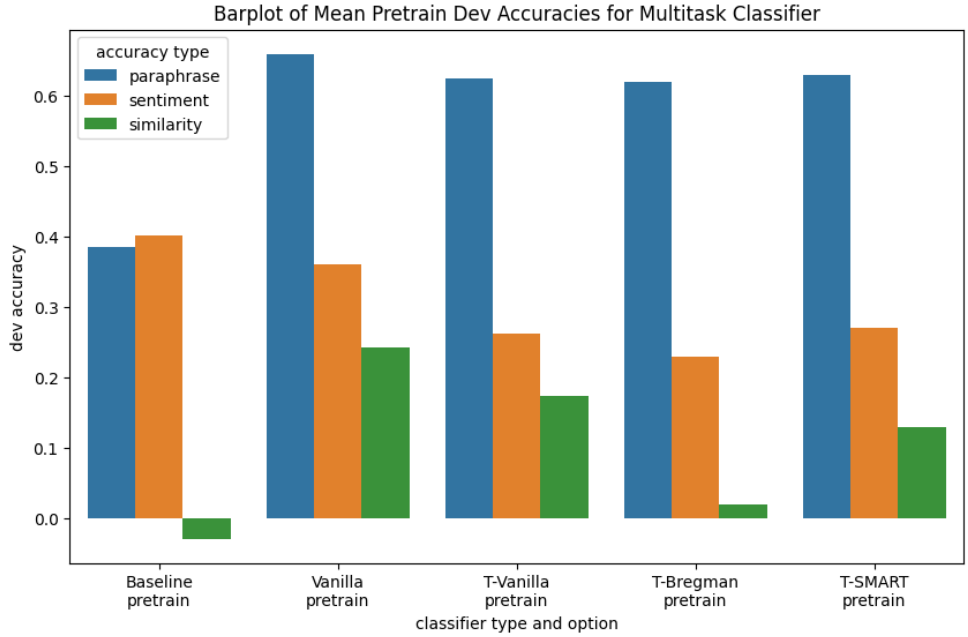

Figure 1: Barplot of Mean Pretrain Dev Accuracies for Multitask Classifier

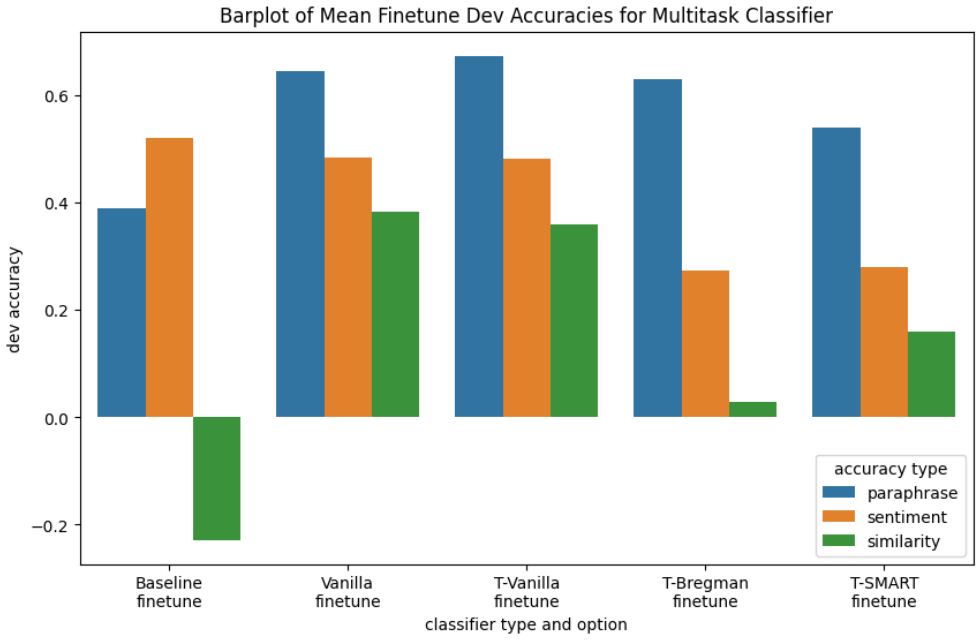

Figure 2: Barplot of Mean Finetune Dev Accuracies for Multitask Classifier

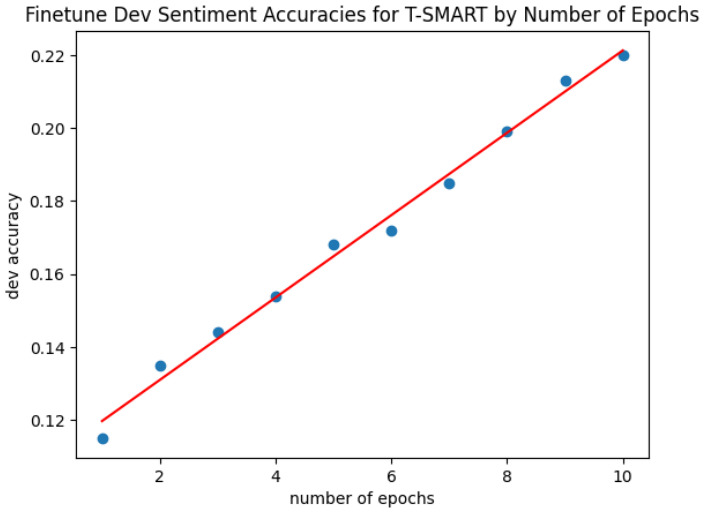

Figure 3: Finetune Dev Sentiment Accuracies for T-SMART by Number of Epochs

have access to the infrastructure necessary to implement them. Handling memory and compute constraints constituted a significant portion of our work toward the implementation of our framework. In order to make progress in the development of our framework, we leveraged different computing platforms and debugging methods. For example, after developing our Baseline locally, we trained our model using Google Colab. When we found that Colab GPU limitations were insufficient for running the Vanilla model on all three datasets, we continued to develop our framework while only testing extensions on one classification task at a time. We then transitioned to training our model using a virtual machine on the Google Cloud Platform and similarly tested our extensions on one classification task at a time. While collecting some of our experimental data, we limited our model to 2 epochs due to compute limitations.

The impact of memory limitations is best demonstrated by **Figure 3**. **Figure 3** visualizes the performance of a smaller version of our T-SMART model that was fine-tuned exclusively for sentiment classification on a subset of the data that was about 10% the size of the datasets that we used for our other models. Due to the memory demands of the T-SMART method, we were unable to train the model on any larger subset of the data. We would encounter CUDA memory limitations that we were unable to solve. These limitations also prevented us from being able to finetune the model on more than one downstream task at a time. However, we see that our implementation of T-SMART shows promise, as **Figure 3** demonstrates a gradual increase in accuracy over a small number of epochs and a small set of training data points. We hypothesize that if had been equipped with the ideal infrastructure for implementing the T-SMART method, our implementation would have seen a level of success more similar to that proposed by Jiang et al. (2020).

7 Conclusion

In the RO-BERT multitask classifier implementation, we have demonstrated the improvements in model performance that finetuning on downstream tasks, applying a slanted triangular learning rate method, applying the MBPP method, and implementing smoothness-inducing adversarial regression have on a baseline minBERT model. However, the performance of these four model extensions did not reach the levels of success reported by Jiang et al. (2020) or Howard and Ruder (2018), demonstrating the difficulty of demonstrating replicability in natural language research and highlighting the need for increased accessibility in the field of machine learning research in order for research insights to extend their benefit to the real world. Future work on RO-BERT could involve identifying and working with research mentors to better analyze replicability and researching modifications on these model extensions to achieve improved performance on par with the work of Jiang et al. (2020) and Howard and Ruder (2018).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

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

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. 2018. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ruixuan Zhang, Zhuoyu Wei, Yu Shi, and Yining Chen. 2019. Bert-al: Bert for arbitrarily long document understanding.

A Appendix

Algorithm 1, as presented by Jiang et al. (2020) to describe their SMART framework (Jiang et al., 2020).

Algorithm 1 SMART: We use the smoothness-inducing adversarial regularizer with $p = \infty$ and the momentum Bregman proximal point method.

Notation: For simplicity, we denote $g_i(\widetilde{x}_i, \bar{\theta}_s) = \frac{1}{|\mathcal{B}|}\sum_{x_i \in \mathcal{B}} \nabla_{\widetilde{x}} \ell_s(f(x_i; \bar{\theta}_s), f(\widetilde{x}_i; \bar{\theta}_s))$ and $\mathrm{AdamUpdate}_{\mathcal{B}}$ denotes the ADAM update rule for optimizing (3) using the mini-batch $\mathcal{B}$; $\Pi_{\mathcal{A}}$ denotes the projection to $\mathcal{A}$.

Input: T: the total number of iterations, $\mathcal{X}$: the dataset, θ_0: the parameter of the pre-trained model, S: the total number of iteration for solving (2), σ^2: the variance of the random initialization for $\widetilde{x}_i$'s, $T_{\widetilde{x}}$: the number of iterations for updating $\widetilde{x}_i$'s, η: the learning rate for updating $\widetilde{x}_i$'s, β: momentum parameter.

$\widetilde{\theta}_1 \leftarrow \theta_0$
for $t = 1, .., T$ **do**
  $\bar{\theta}_1 \leftarrow \theta_{t-1}$
  for $s = 1, .., S$ **do**
    Sample a mini-batch $\mathcal{B}$ from $\mathcal{X}$
    For all $x_i \in \mathcal{B}$, initialize $\widetilde{x}_i \leftarrow x_i + \nu_i$ with $\nu_i \sim \mathcal{N}(0, \sigma^2 I)$
    for $m = 1, .., T_{\widetilde{x}}$ **do**
      $\widetilde{g}_i \leftarrow \frac{g_i(\widetilde{x}_i, \bar{\theta}_s)}{\|g_i(\widetilde{x}_i, \bar{\theta}_s)\|_\infty}$
      $\widetilde{x}_i \leftarrow \Pi_{\|\widetilde{x}_i - x\|_\infty \le \epsilon}(\widetilde{x}_i + \eta \widetilde{g}_i)$
    end for
    $\bar{\theta}_{s+1} \leftarrow \mathrm{AdamUpdate}_{\mathcal{B}}(\bar{\theta}_s)$
  end for
  $\theta_t \leftarrow \bar{\theta}_S$
  $\widetilde{\theta}_{t+1} \leftarrow (1 - \beta)\bar{\theta}_S + \beta \widetilde{\theta}_t$
end for

Output: θ_T