SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization

Haoming Jiang *
Georgia Tech
jianghm@gatech.edu

Pengcheng He, Weizhu Chen
Microsoft Dynamics 365 AI
{penhe,wzchen}@microsoft.com

Xiaodong Liu, Jianfeng Gao
Microsoft Research
{xiaodl,jfgao}@microsoft.com

Tuo Zhao
Georgia Tech
tourzhao@gatech.edu

Abstract

Transfer learning has fundamentally changed the landscape of natural language processing (NLP). Many state-of-the-art models are first pre-trained on a large text corpus and then fine-tuned on downstream tasks. However, due to limited data resources from downstream tasks and the extremely high complexity of pre-trained models, aggressive fine-tuning often causes the fine-tuned model to overfit the training data of downstream tasks and fail to generalize to unseen data. To address such an issue in a principled manner, we propose a new learning framework for robust and efficient fine-tuning for pre-trained models to attain better generalization performance. The proposed framework contains two important ingredients: 1. Smoothness-inducing regularization, which effectively manages the complexity of the model; 2. Bregman proximal point optimization, which is an instance of trust-region methods and can prevent aggressive updating. Our experiments show that the proposed framework achieves new state-of-the-art performance on a number of NLP tasks including GLUE, SNLI, SciTail and ANLI. Moreover, it also outperforms the state-of-the-art T5 model, which is the largest pre-trained model containing 11 billion parameters, on GLUE. [1]

1 Introduction

The success of natural language processing (NLP) techniques relies on huge amounts of labeled data in many applications. However, large amounts of labeled data are usually prohibitive or expensive to obtain. To address this issue, researchers have resorted to transfer learning.

Transfer learning considers the scenario, where we have limited labeled data from the target domain for a certain task, but we have relevant tasks with a large amount of data from different domains (also known as out-of-domain data). The goal is to transfer the knowledge from the high-resource domains to the low-resource target domain. Here we are particularly interested in the popular two-stage transfer learning framework (Pan and Yang, 2009). The first stage is pre-training, where a high-capacity model is trained for the out-of-domain high-resource relevant tasks. The second stage is fine-tuning, where the high-capacity model is adapted to the low-resource task in the target domain.

For many applications in NLP, most popular transfer learning methods choose to pre-train a large language model, e.g., ELMo (Peters et al., 2018), GPT (Radford et al., 2019) and BERT (Devlin et al., 2019). Such a language model can capture general semantic and syntactic information that can be further used in downstream NLP tasks. The language model is particularly attractive, because it can be trained in a completely unsupervised manner with huge amount of unlabeled data, which are extremely cheap to fetch from internet nowadays. The resulting extremely large multi-domain text corpus allows us to train huge language models. To the best of our knowledge, by far the largest language model, T5, has an enormous size of about 11 billion parameters (Raffel et al., 2019).

For the second fine-tuning stage, researchers adapt the pre-trained language model to the target task/domain. They usually replace the top layer of the language model by a task/domain-specific sub-network, and then continue to train the new model with the limited data of the target task/domain. Such a fine-tuning approach accounts for the low-resource issue in the target task/domain, and has achieved state-of-the-art performance in many popular NLP benchmarks (Devlin et al., 2019; Liu et al., 2019c; Yang et al.,

* Work was done during an internship at Microsoft Dynamics 365 AI.

[1] https://github.com/namisan/mt-dnn

2019; Lan et al., 2019; Dong et al., 2019; Raffel et al., 2019).

Due to the limited data from the target task/domain and the **extremely high complexity** of the pre-trained model, **aggressive fine-tuning** often makes the adapted model overfit the training data of the target task/domain and therefore does not generalize well to unseen data. To mitigate this issue, the fine-tuning methods often rely on hyper-parameter tuning heuristics. For example, Howard and Ruder (2018) use a heuristic learning rate schedule and gradually unfreeze the layers of the language model to improve the fine-tune performance; Peters et al. (2019) give a different suggestion that they only adapt certain layers and freeze the others; (Houlsby et al., 2019; Stickland and Murray, 2019) propose to add additional layers to the pre-trained model and fine-tune both of them or only the additional layers. However, these methods require significant tuning efforts.

To fully harness the power of fine-tuning in a more principled manner, we propose a new learning framework for robust and efficient fine-tuning on the pre-trained language models through regularized optimization techniques. Specifically, our framework consists of two important ingredients for preventing overfitting:

(I) To effectively control the **extremely high complexity** of the model, we propose a *Smoothness-inducing Adversarial Regularization* technique. Our proposed regularization is motivated by local shift sensitivity in existing literature on robust statistics. Such regularization encourages the output of the model not to change much, when injecting a small perturbation to the input. Therefore, it enforces the smoothness of the model, and effectively controls its capacity (Mohri et al., 2018).

(II) To prevent **aggressive updating**, we propose a class of *Bregman Proximal Point Optimization* methods. Our proposed optimization methods introduce a trust-region-type regularization (Conn et al., 2000) at each iteration, and then update the model only within a small neighborhood of the previous iterate. Therefore, they can effectively prevent aggressive updating and stabilize the fine-tuning process.

We compare our proposed method with several state-of-the-art competitors proposed in (Zhu et al., 2020; Liu et al., 2019b,c; Lan et al., 2019; Raffel et al., 2019) and show that our proposed method significantly improves the training stability and generalization, and achieves comparable or better performance on multiple NLP tasks. We highlight that our single model with 356M parameters (without any ensemble) can achieve three state-of-the-art results on GLUE, even compared with all existing ensemble models and the T5 model (Raffel et al., 2019), which contains 11 billion parameters. Furthermore, we also demonstrate that the proposed framework complements with SOTA fine-tuning methods (Liu et al., 2019b) and outperforms the T5 model.

We summarize our contribution as follows: 1. We introduce the smoothness-inducing adversarial regularization and proximal point optimization into large scale language model fine-tuning; 2. We achieve state-of-the-art results on several popular NLP benchmarks (e.g., GLUE, SNLI, SciTail, and ANLI).

Notation: We use $f(x;\theta)$ to denote a mapping f associated with the parameter θ from input sentences x to an output space, where the output is a multi-dimensional probability simplex for classification tasks and a scalar for regression tasks. $\Pi_{\mathcal{A}}$ denotes the projection operator to the set $\mathcal{A}$. $\mathcal{D}_{KL}(P||Q) = \sum_k p_k \log(p_k/q_k)$ denotes the KL-divergence of two discrete distributions P and Q with the associated parameters of p_k and q_k, respectively.

2 Background

The transformer models were originally proposed in Vaswani et al. (2017) for neural machine translation. Their superior performance motivated Devlin et al. (2019) to propose a bidirectional transformer-based language model named BERT. Specifically, Devlin et al. (2019) pre-trained the BERT model using a large corpus without any human annotation through unsupervised learning tasks. BERT motivated many follow-up works to further improve the pre-training by introducing new unsupervised learning tasks (Yang et al., 2019; Dong et al., 2019; Joshi et al., 2020), enlarging model size (Lan et al., 2019; Raffel et al., 2019), enlarging training corpora (Liu et al., 2019c; Yang et al., 2019; Raffel et al., 2019) and multi-tasking (Liu et al., 2019a,b).

The pre-trained language model is then adapted to downstream tasks and further fine-tuned. Specifically, the top layer of the language model can be replaced by a task-specific layer and then continue to train on downstream tasks. To prevent overfitting, existing heuristics include choosing a

small learning rate or a triangular learning rate schedule, and a small number of iterations, and other fine-tuning tricks mentioned in (Howard and Ruder, 2018; Peters et al., 2019; Houlsby et al., 2019; Stickland and Murray, 2019).

Our proposed regularization technique is related to several existing works (Miyato et al., 2018; Zhang et al., 2019; Shu et al., 2018). These works consider similar regularization techniques, but target at other applications with different motivations, e.g., semi-supervised learning, unsupervised domain adaptation and harnessing adversarial examples in image classification.

Our proposed optimization technique covers a large class of Bregman proximal point methods in existing literature on optimization, including vanilla proximal point method proposed in Rockafellar (1976), generalized proximal point method (Teboulle, 1997; Eckstein, 1993), accelerated proximal point method, and other variants (Güler, 1991, 1992; Parikh et al., 2014).

There is a related fine-tuning method – FreeLB Zhu et al. (2020), which adapted a robust adversarial training method. However, our framework focuses on the local smoothness, leading to a significant performance improvement. More discussion and comparison are provided in Section 4.

3 The Proposed Method

We describe the proposed learning framework – ***SMART*** for robust and efficient fine-tuning of pre-trained language models. Our framework consists of two important ingredients: ***SM****oothness-inducing* ***A****dversarial* ***R****egularization* and *B**R**egman p**R**oximal poin**T** op**T**imization*[2].

3.1 Smoothness-Inducing Adversarial Regularization

We propose to impose an explicit regularization to effectively control the model complexity at the fine-tuning stage. Specifically, given the model $f(\cdot;\theta)$ and n data points of the target task denoted by $\{(x_i, y_i)\}_{i=1}^n$, where x_i's denote the embedding of the input sentences obtained from the first embedding layer of the language model and y_i's are the associated labels, our method essentially solves the following optimization for fine-tuning:

$$\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\mathrm{s}} \mathcal{R}_{\mathrm{s}}(\theta), \qquad (1)$$

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

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

and $\ell(\cdot,\cdot)$ is the loss function depending on the target task, $\lambda_{\mathrm{s}} > 0$ is a tuning parameter, and $\mathcal{R}_{\mathrm{s}}(\theta)$ is the smoothness-inducing adversarial regularizer. Here we define $\mathcal{R}_{\mathrm{s}}(\theta)$ as

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n}\sum_{i=1}^n \max_{\|\widetilde{x}_i - x_i\|_p \le \epsilon} \ell_{\mathrm{s}}(f(\widetilde{x}_i;\theta), f(x_i;\theta)),$$

where $\epsilon > 0$ is a tuning parameter. Note that for classification tasks, $f(\cdot;\theta)$ outputs a probability simplex and ℓ_{s} is chosen as the symmetrized KL-divergence, i.e.,

$$\ell_{\mathrm{s}}(P, Q) = \mathcal{D}_{\mathrm{KL}}(P||Q) + \mathcal{D}_{\mathrm{KL}}(Q||P);$$

For regression tasks, $f(\cdot;\theta)$ outputs a scalar and ℓ_{s} is chosen as the squared loss, i.e., $\ell_s(p, q) = (p-q)^2$. Note that the computation of $\mathcal{R}_{\mathrm{s}}(\theta)$ involves a maximization problem and can be solved efficiently by projected gradient ascent.

We remark that the proposed smoothness-inducing adversarial regularizer was first used in Miyato et al. (2018) for semi-supervised learning with $p = 2$, and then in Shu et al. (2018) for unsupervised domain adaptation with $p = 2$, and more recently in Zhang et al. (2019) for harnessing the adversarial examples in image classification with $p = \infty$. To the best of our knowledge, we are the first applying such a regularizer to fine-tuning of pre-trained language models.

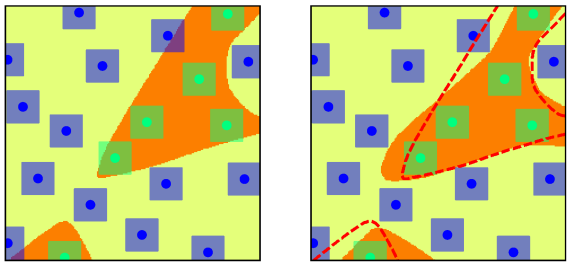
The smoothness-inducing adversarial regularizer is essentially measuring the local Lipschitz continuity of f under the metric ℓ_s. More precisely speaking, the output of f does not change much if we inject a small perturbation (ℓ_p norm bounded by ϵ) to x_i. Therefore, by minimizing the objective in (1), we can encourage f to be smooth within the neighborhoods of all x_i's. Such a smoothness-inducing property is particularly helpful to prevent overfitting and improve generalization on a low resource target domain for a certain task. An illustration is provided in Figure 1.

Note that the idea of measuring the local Lipschitz continuity is similar to the local shift sensitivity criterion in existing literature on robust statistics, which dates back to 1960's (Hampel, 1974; Huber, 2011). This criterion has been used to characterize the dependence of an estimator on the value of one of the sample points.

3.2 Bregman Proximal Point Optimization

We propose to develop a class of Bregman proximal point optimization methods to solve (1). Such optimization methods impose a strong penalty at

[2]The complete name of our proposed method is ***SMAR***^{3}***T***^{2}, but we use ***SMART*** for notational simplicity.



(a) (b)

Figure 1: Decision boundaries learned without (a) and with (b) smoothness-inducing adversarial regularization, respectively. The red dotted line in (b) represents the decision boundary in (a). As can be seen, the output f in (b) does not change much within the neighborhood of training data points.

each iteration to prevent the model from aggressive update. Specifically, we use a pre-trained model as the initialization denoted by $f(\cdot;\theta_0)$. At the $(t+1)$-th iteration, the vanilla Bregman proximal point (VBPP) method takes

$$\theta_{t+1} = \text{argmin}_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \theta_t), \quad (2)$$

where $\mu > 0$ is a tuning parameter, and $\mathcal{D}_{\text{Breg}}(\cdot,\cdot)$ is the Bregman divergence defined as

$$\mathcal{D}_{\text{Breg}}(\theta, \theta_t) = \frac{1}{n}\sum_{i=1}^n \ell_{\text{s}}(f(x_i;\theta), f(x_i;\theta_t)),$$

where ℓ_s is defined in Section 3.1. As can be seen, when μ is large, the Bregman divergence at each iteration of the VBPP method essentially serves as a strong regularizer and prevents θ_{t+1} from deviating too much from the previous iterate θ_t. This is also known as the trust-region type iteration in existing optimization literature (Conn et al., 2000). Consequently, the Bregman proximal point method can effectively retain the knowledge of the out-of-domain data in the pre-trained model $f(\cdot;\theta_0)$. Since each subproblem (2) of VBPP does not admit a closed-form solution, we need to solve it using SGD-type algorithms such as ADAM. Note that we do not need to solve each subproblem until convergence. A small number of iterations are sufficient to output a reliable initial solution for solving the next subproblem.

Moreover, the Bregman proximal point method is capable of adapting to the information geometry (See more details in Raskutti and Mukherjee (2015)) of machine learning models and achieving better computational performance than the standard proximal point method (i.e., $\mathcal{D}_{\text{Breg}}(\theta,\theta_t) = \|\theta-\theta_t\|_2^2$) in many applications.

Acceleration by Momentum. Similar to other optimization methods in existing literature, we can accelerate the Bregman proximal point method by introducing an additional momentum to the update. Specifically, at the $(t+1)$-th iteration, the momentum Bregman proximal point (MBPP) method takes

$$\theta_{t+1} = \text{argmin}_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \widetilde{\theta}_t), \quad (3)$$

where $\widetilde{\theta}_t = (1-\beta)\theta_t + \beta\widetilde{\theta}_{t-1}$ is the exponential moving average and $\beta \in (0,1)$ is the momentum parameter. The MBPP method is also called the "Mean Teacher" method in existing literature (Tarvainen and Valpola, 2017) and has been shown to achieve state-of-the-art performance in popular semi-supervised learning benchmarks. For convenience, we summarize the MBPP method in Algorithm 1.

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 $g_i(\widetilde{x}_i, \bar{\theta}_s) = \frac{1}{|\mathcal{B}|}\sum_{x_i\in\mathcal{B}} \nabla_{\widetilde{x}} \ell_{\text{s}}(f(x_i;\bar{\theta}_s), f(\widetilde{x}_i;\bar{\theta}_s))$ and $\text{AdamUpdate}_{\mathcal{B}}$ 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
    $\bar{\theta}_{s+1} \leftarrow \text{AdamUpdate}_{\mathcal{B}}(\bar{\theta}_s)$
  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

4 Experiment – Main Results

We demonstrate the effectiveness of SMART for fine-tuning large language models using GLUE (Wang et al., 2018) by comparing with existing state-of-the-art methods. Dataset details can be found in Appendix A.

4.1 Implementation Details

Our implementation of SMART is based on BERT[3] (Wolf et al., 2019), RoBERTa [4] (Liu et al., 2019c), MT-DNN [5] (Liu et al., 2020b) and HNN[6]. We used ADAM (Kingma and Ba, 2014) and RADAM (Liu et al., 2020a) as our optimizers with a learning rate in the range $\in \{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ and a batch size $\in \{16, 32, 64\}$. The maximum number of epochs was set to 6. A linear learning rate decay schedule with warm-up of 0.1 was used, unless stated otherwise. We also set the dropout rate of all the task specific layers as 0.1, except 0.3 for MNLI and 0.05 for CoLA. To avoid gradient exploding, we clipped the gradient norm within 1. All the texts were tokenized using wordpieces and were chopped to spans no longer than 512 tokens. For SMART, we set the perturbation size $\epsilon = 10^{-5}$ and $\sigma = 10^{-5}$. We set $\mu = 1$ and $\lambda_s \in \{1, 3, 5\}$. The learning rate η in Algorithm 1 is set to 10^{-3}. We set β = 0.99 for the first 10% of the updates ($t \leq 0.1T$) and $\beta = 0.999$ for the rest of the updates ($t > 0.1T$) following (Tarvainen and Valpola, 2017). Lastly, we simply set $S = 1, T_{\tilde{x}} = 1$ in Algorithm 1.

4.2 GLUE Main Results

We compare SMART with a range of strong baselines including large pre-trained models and approaches with adversarial training, and a list of state-of-the-art models that have been submitted to the GLUE leaderboard. SMART is a generic framework, we evaluate our framework on two pre-trained models, the $\text{BERT}_{\text{BASE}}$ model (Devlin et al., 2019) and the $\text{RoBERTa}_{\text{LARGE}}$ model (Liu et al., 2019c), which are available publicly. Most of our analyses are done with the $\text{BERT}_{\text{BASE}}$ to make our results comparable to other work, since $\text{BERT}_{\text{BASE}}$ has been widely used as a baseline. To make our result comparable to other state-of-the-art models, we also evaluate the framework on the $\text{RoBERTa}_{\text{LARGE}}$ model.

• BERT (Devlin et al., 2019): This is the $\text{BERT}_{\text{BASE}}$ model released by the authors. In Devlin et al. (2019), authors only reported the development results on a few tasks, thus we reproduced the baseline results, which are denoted by $\textbf{BERT}_{\textbf{ReImp}}$.

• RoBERTa (Liu et al., 2019c): This is the $\text{RoBERTa}_{\text{LARGE}}$ released by authors, and we present the reported results on the GLUE dev.

• PGD, FreeAT, FreeLB (Zhu et al., 2020): They are three adversarial training approaches built on top of the $\text{RoBERTa}_{\text{LARGE}}$.

• SMART: our proposed method as described in section 3. We use both the $\text{BERT}_{\text{BASE}}$ model ($\text{SMART}_{\text{BERT}}$) and the $\text{RoBERTa}_{\text{LARGE}}$ model ($\text{SMART}_{\text{RoBERTa}}$) as the pretrained model to evaluate the effectiveness of SMART.

The main results are reported in Table 1. This table can be clustered into two groups based on different pretrained models: the $\text{BERT}_{\text{BASE}}$ model (the first group) and the $\text{RoBERTa}_{\text{LARGE}}$ model (the second group). The detailed discussions are as follows.

For a fair comparison, we reproduced the BERT baseline ($\text{BERT}_{\text{ReImp}}$), since several results on the GLUE development set were missed. Our reimplemented BERT baseline is even stronger than the originally reported results in Devlin et al. (2019). For instance, the reimplemented model obtains 84.5% (vs. 84.4%) on MNLI in-domain development in terms of accuracy. On SST-2, $\text{BERT}_{\text{ReImp}}$ outperforms BERT by 0.2% (92.9% vs. 92.7%) accuracy. All these results demonstrate the fairness of our baselines.

Comparing with two strong baselines BERT and RoBERTa [7], SMART, including $\text{SMART}_{\text{BERT}}$ and $\text{SMART}_{\text{RoBERTa}}$, consistently outperforms them across all 8 GLUE tasks by a big margin. Comparing with BERT, $\text{SMART}_{\text{BERT}}$ obtained 85.6% (vs. 84.5%) and 86.0% (vs. 84.4%) in terms of accuracy, which is 1.1% and 1.6% absolute improvement, on the MNLI in-domain and out-domain settings. Even comparing with the state-of-the-art model RoBERTa, $\text{SMART}_{\text{RoBERTa}}$ improves 0.8% (91.1% vs. 90.2%) on MNLI in-domain development set. Interestingly, on the

[3] https://github.com/huggingface/transformers
[4] https://github.com/pytorch/fairseq
[5] https://github.com/namisan/mt-dnn
[6] https://github.com/namisan/mt-dnn/tree/master/hnn

[7] In our experiments, we use BERT referring the $\text{BERT}_{\text{BASE}}$ model, which has 110 million parameters, and RoBERTa referring the $\text{RoBERTa}_{\text{LARGE}}$ model, which has 356 million parameters, unless stated otherwise.

Model	MNLI-m/mm Acc	QQP Acc/F1	RTE Acc	QNLI Acc	MRPC Acc/F1	CoLA Mcc	SST Acc	STS-B P/S Corr
$\text{BERT}_{\text{BASE}}$								
BERT (Devlin et al., 2019)	84.4/-	-	-	88.4	-/86.7	-	92.7	-
$\text{BERT}_{\text{ReImp}}$	84.5/84.4	90.9/88.3	63.5	91.1	84.1/89.0	54.7	92.9	89.2/88.8
$\text{SMART}_{\text{BERT}}$	**85.6/86.0**	**91.5/88.5**	**71.2**	**91.7**	**87.7/91.3**	**59.1**	**93.0**	**90.0/89.4**
$\text{RoBERTa}_{\text{LARGE}}$								
RoBERTa (Liu et al., 2019c)	90.2/-	92.2/-	86.6	94.7	-/90.9	68.0	96.4	92.4/-
PGD (Zhu et al., 2020)	90.5/-	92.5/-	87.4	94.9	-/90.9	69.7	96.4	92.4/-
FreeAT (Zhu et al., 2020)	90.0/-	92.5/-	86.7	94.7	-/90.7	68.8	96.1	92.4/-
FreeLB (Zhu et al., 2020)	90.6/-	**92.6**/-	88.1	95.0	-/91.4	**71.1**	96.7	92.7/-
$\text{SMART}_{\text{RoBERTa}}$	**91.1/91.3**	92.4/89.8	**92.0**	**95.6**	**89.2/92.1**	70.6	**96.9**	**92.8/92.6**

Table 1: Main results on GLUE development set. The best result on each task produced by a single model is in **bold** and "-" denotes the missed result.

Model /#Train	CoLA 8.5k	SST 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	**Score**	#param
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1	-
Ensemble Models												
RoBERTa[1]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5	356M
FreeLB[2]	68.0	96.8	93.1/90.8	92.4/92.2	**74.8**/90.3	91.1/90.7	98.8	88.7	89.0	50.1	88.8	356M
ALICE[3]	69.2	97.1	93.6/91.5	92.7/92.3	74.4/**90.7**	90.7/90.2	**99.2**	87.3	89.7	47.8	89.0	340M
ALBERT[4]	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3/91.0	**99.2**	89.2	91.8	50.2	89.4	235M*
MT-DNN-SMART†	69.5	**97.5**	**93.7/91.6**	**92.9/92.5**	73.9/90.2	91.0/90.8	**99.2**	89.7	94.5	50.2	**89.9**	356M
Single Model												
$\text{BERT}_{\text{LARGE}}$[5]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5	335M
MT-DNN[6]	62.5	95.6	90.0/86.7	88.3/87.7	72.4/89.6	86.7/86.0	93.1	75.5	65.1	40.3	82.7	335M
T5[8]	**70.8**	97.1	91.9/89.2	92.5/92.1	74.6/90.4	**92.0/91.7**	96.7	**92.5**	**93.2**	**53.1**	89.7	11,000M
$\text{SMART}_{\text{RoBERTa}}$	65.1	**97.5**	**93.7/91.6**	**92.9/92.5**	74.0/90.1	91.0/90.8	95.4	87.9	91.8[8]	50.2	88.4	356M

Table 2: GLUE test set results scored using the GLUE evaluation server. The state-of-the-art results are in **bold**. All the results were obtained from https://gluebenchmark.com/leaderboard on December 5, 2019. SMART uses the classification objective on QNLI. Model references: [1] Liu et al. (2019c); [2]Zhu et al. (2020); [3]Wang et al. (2019); [4]Lan et al. (2019); [5] Devlin et al. (2019); [6] Liu et al. (2019b); [7] Raffel et al. (2019) and [8] He et al. (2019), Kocijan et al. (2019). * ALBERT uses a model similar in size, architecture and computation cost to a 3,000M BERT (though it has dramatically fewer parameters due to parameter sharing). † Mixed results from ensemble and single of MT-DNN-SMART and with data augmentation.

MNLI task, the performance of SMART on the out-domain setting is better than the in-domain setting, e.g., (86.0% vs. 85.6%) by $\text{SMART}_{\text{BERT}}$ and (91.3% vs. 91.1%) by $\text{SMART}_{\text{RoBERTa}}$, showing that our proposed approach alleviates the domain shifting issue. Furthermore, on the small tasks, the improvement of SMART is even larger. For example, comparing with BERT, $\text{SMART}_{\text{BERT}}$ obtains 71.2% (vs. 63.5%) on RTE and 59.1% (vs. 54.7%) on CoLA in terms of accuracy, which are 7.7% and 4.4% absolute improvement for RTE and CoLA, respectively; similarly, $\text{SMART}_{\text{RoBERTa}}$ outperforms RoBERTa 5.4% (92.0% vs. 86.6%) on RTE and 2.6% (70.6% vs. 68.0%) on CoLA.

We also compare SMART with a range of models which used adversarial training such as FreeLB. From the bottom rows in Table 1, SMART outperforms PGD and FreeAT across the all 8 GLUE tasks. Comparing with the current state-of-the-art adversarial training model, FreeLB, SMART outperforms it on 6 GLUE tasks out of a total of 8 tasks (MNLI, RTE, QNLI, MRPC, SST-2 and STS-B) showing the effectiveness of our model.

Table 2 summarizes the current state-of-the-art models on the GLUE leaderboard. SMART obtains a competitive result comparing with T5 (Raffel et al., 2019), which is the leading model at the GLUE leaderboard. T5 has 11 billion parameters,

while SMART only has 356 millions. Among this super large model (T5) and other ensemble models (e.g., ALBERT, ALICE), SMART, which is a single model, still sets new state-of-the-art results on SST-2, MRPC and STS-B. By combining with the Multi-task Learning framework (MT-DNN), MT-DNN-SMART obtains new state-of-the-art on GLUE, pushing the GLUE benchmark to 89.9%. More discussion will be provided in Section 5.3.

5 Experiment – Analysis and Extension

In this section, we first analyze the effectiveness of each component of the proposed method. We also study that whether the proposed method is complimentary to multi-task learning. We further extend SMART to domain adaptation and use both SNLI (Bowman et al., 2015) and SciTail (Khot et al., 2018) to evaluate the effectiveness. Finally, we verified the robustness of the proposed method on ANLI (Nie et al., 2019).

5.1 Ablation Study

Note that due to the limitation of time and computational resources, all the experiments reported below are based on the **BERT$_{\text{BASE}}$** model. In this section, we study the importance of each component of SMART: smoothness-inducing adversarial regularization and Bregman proximal point optimization. All models in this study used the BERT$_{\text{BASE}}$ as the encoder for fast training. Furthermore, we also include the BERT$_{\text{BASE}}$ model as an additional baseline for a fair comparison. SMART denotes the proposed model. Then we set λ_s to 0, which denotes as -$\mathcal{R}_{\text{s}}$. The model with $\mu = 0$ is noted as -$\mathcal{D}_{\text{Breg}}$.

Model	MNLI Acc	RTE Acc	QNLI Acc	SST Acc	MRPC Acc
BERT	84.5	63.5	91.1	92.9	89.0
SMART	**85.6**	**71.2**	**91.7**	**93.0**	**91.3**
-$\mathcal{R}_{\text{s}}$	84.8	70.8	91.3	92.8	90.8
-$\mathcal{D}_{\text{Breg}}$	85.4	**71.2**	91.6	92.9	91.2

Table 3: Ablation study of SMART on 5 GLUE tasks. Note that all models used the BERT$_{\text{BASE}}$ model as their encoder.

The results are reported in Table 3. It is expected that the removal of either component (smooth regularization or proximal point method) in SMART would result in a performance drop. For example, on MNLI, removing smooth regularization leads to a 0.8% (85.6% vs. 84.8) performance drop, while removing the Breg proximal point optimization, results in a performance drop of 0.2% (85.6% vs. 85.4%). It demonstrates that these two components complement each other. Interestingly, all three proposed models outperform the BERT baseline model demonstrating the effectiveness of each module. Moreover, we obersere that the generalization performance benefits more from SMART on small datasets (i.e., RTE and MRPC) by preventing overfitting.

5.2 Error Analysis

To understand why SMART improves the performance, we analyze it on the ambiguous samples of MNLI dev set containing 3 classes, where each sample has 5 annotations. Based on the degree of agreement between these annotations, we divide the samples into 4 categories: 1) **5/0/0** all five annotations are the same; 2) **4/1/0** four annotations are the same; 3) **3/2/0** three annotations are the same and the other two annotations are the same; 4) **3/1/1** three annotations are the same and the other two annotations are different.

Figure 2 summarizes the results in terms of both accuracy and KL-divergence: $-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{3} p_j(x_i)\log(f_j(x_i))$. For a given sample x_i, the KL-Divergence evaluates the similarity between the model prediction $\{f_j(x_i)\}_{j=1}^{3}$ and the annotation distribution $\{p_j(x_i)\}_{j=1}^{3}$. We observe that SMART$_{\text{RoBERTa}}$ outperforms RoBERTa across all the settings. Further, on high degree of ambiguity (low degree of agreement), SMART$_{\text{RoBERTa}}$ obtains an even larger improvement showing its robustness to ambiguity.

5.3 SMART with Multi-task Learning

It has been shown that multi-task learning (MTL, Caruana (1997); Liu et al. (2015, 2019b)) has a regularization effect via alleviating overfitting to a specific task. One question is whether MTL helps SMART as well. In this section, we are going to answer this question. Following Liu et al. (2019b), we first "pre-trained" shared embeddings using MTL with SMART, denoted as **MT-DNN-SMART** [8], and then adapted the training data on each task on top of the shared embeddings. We also include a baseline which fine-tuned each task

[8]Due to limitation of computational resources, we only trained jointly using MTL on MNLI, RTE, QNLI, SST and MRPC, while MT-DNN was trained on the whole GLUE tasks except CoLA.

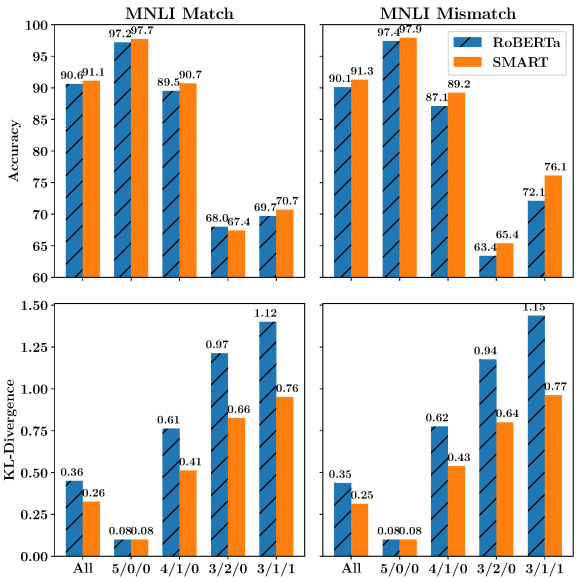


Figure 2: Score breakdown by degree of agreement.

on the publicly released MT-DNN checkpoint [9], which is indicated as **MT-DNN-SMART$_{v0}$**.

Model	MNLI Acc	RTE Acc	QNLI Acc	SST Acc	MRPC F1
BERT	84.5	63.5	91.1	92.9	89.0
MT-DNN	85.3	79.1	91.5	**93.6**	89.2
SMART	85.6	71.2	91.6	93.0	91.3
MT-DNN-SMART$_{v0}$	**85.7**	80.2	**92.0**	93.3	91.5
MT-DNN-SMART	**85.7**	**81.2**	**92.0**	93.5	**91.7**

Table 4: Comparison between SMART and MTL.

We observe that both MT-DNN and SMART consistently outperform the BERT model on all five GLUE tasks. Furthermore, SMART outperforms MT-DNN on MNLI, QNLI, and MRPC, while it obtains worse results on RTE and SST, showing that MT-DNN is a strong counterpart for SMART. By combining these two models, MT-DNN-SMART$_{v0}$ enjoys advantages of both and thus improved the final results. For example, it achieves 85.7% (+0.1%) on MNLI and 80.2% (+1.1%) on RTE comparing with the best results of MT-DNN and SMART demonstrating that these two techniques are orthogonal. Lastly we also trained SMART jointly and then finetuned on each task like Liu et al. (2019b). We observe that MT-DNN-SMART outperformes MT-DNN-SMART$_{v0}$ and MT-DNN across all 5 tasks (except MT-DNN on SST) showing that SMART improves the generalization of MTL.

[9] It is from: https://github.com/namisan/mt-dnn. Note that we did not use the complicated answer module, e.g., SAN (Liu et al., 2018).

Model	0.1%	1%	10%	100%
SNLI Dataset (Dev Accuracy%)				
#Training Data	549	5,493	54,936	549,367
BERT	52.5	78.1	86.7	91.0
MT-DNN	82.1	85.2	88.4	91.5
MT-DNN-SMART	**82.7**	**86.0**	**88.7**	**91.6**
SciTail Dataset (Dev Accuracy%)				
#Training Data	23	235	2,359	23,596
BERT	51.2	82.2	90.5	94.3
MT-DNN	81.9	88.3	91.1	95.8
MT-DNN-SMART	**82.3**	**88.6**	**91.3**	**96.1**

Table 5: Domain adaptation on SNLI and SciTail.

5.4 Domain Adaptation

In this section, we evaluate our model on the domain adaptation setting. Following Liu et al. (2019b), we start with the default training/dev/test set of SNLI and SciTail. Then, we randomly sample 0.1%, 1%, 10% and 100% of its training data, which is used to train a model.

The results are reported in Table 5. We observe that both MT-DNN and MT-DNN-SMART significantly outperform the BERT baseline. Comparing with MT-DNN, MT-DNN-SMART also achieves some improvements indicating the robustness of SMART. Furthermore, MT-DNN-SMART outperforms current state-of-the-art on the SNLI/SciTail test.

5.5 Results on SNLI and SciTail

In Table 7, we compare our methods, using all in-domain training data, against several state-of-the-art models. We observe that SMART obtains the same improvement on SNLI in the BERT setting. Combining SMART with MT-DNN achieves a significant improvement, e.g., our BASE model even outperforms the BERT$_{LARGE}$ model. Similar observation is found on SciTail and in the BERT$_{LARGE}$ model setting. We see that incorporating SMART into MT-DNN achieves new state-of-the-art results on both SNLI and SciTail, pushing benchmarks to 91.7% on SNLI and 95.2% on SciTail.

5.6 Robustness

One important property of the machine learning model is its robustness to adversarial attack. We

Method	Dev				Test			
	R1	R2	R3	All	R1	R2	R3	All
MNLI + SNLI + ANLI + FEVER								
$\text{BERT}_{\text{LARGE}}$ (Nie et al., 2019)	57.4	48.3	43.5	49.3	-	-	-	44.2
$\text{XLNet}_{\text{LARGE}}$ (Nie et al., 2019)	67.6	50.7	48.3	55.1	-	-	-	52.0
$\text{RoBERTa}_{\text{LARGE}}$ (Nie et al., 2019)	73.8	48.9	44.4	53.7	-	-	-	49.7
$\text{SMART}_{\text{RoBERTa-LARGE}}$	74.5	50.9	47.6	**57.1**	72.4	49.8	50.3	**57.1**
ANLI								
$\text{RoBERTa}_{\text{LARGE}}$ (Nie et al., 2019)	71.3	43.3	43.0	51.9	-	-	-	-
$\text{SMART}_{\text{RoBERTa-LARGE}}$	74.2	49.5	49.2	**57.1**	72.4	50.3	49.5	**56.9**

Table 6: Experiment Result for Each Round of ANLI.

Model	Dev	Test
SNLI Dataset (Accuracy%)		
$\text{BERT}_{\text{BASE}}$	91.0	90.8
$\text{BERT}_{\text{BASE}}$+SRL(Zhang et al., 2018)	-	90.3
$\text{MT-DNN}_{\text{BASE}}$	91.4	91.1
$\text{SMART}_{\text{BERT-BASE}}$	91.4	91.1
$\text{MT-DNN-SMART}_{\text{BASEv0}}$	91.7	91.4
$\text{MT-DNN-SMART}_{\text{BASE}}$	91.7	91.5
$\text{BERT}_{\text{LARGE}}$+SRL(Zhang et al., 2018)	-	91.3
$\text{BERT}_{\text{LARGE}}$	91.7	91.0
$\text{MT-DNN}_{\text{LARGE}}$	92.2	91.6
$\text{MT-DNN-SMART}_{\text{LARGEv0}}$	**92.6**	**91.7**
SciTail Dataset (Accuracy%)		
GPT (Radford et al., 2018)	-	88.3
$\text{BERT}_{\text{BASE}}$	94.3	92.0
$\text{MT-DNN}_{\text{BASE}}$	95.8	94.1
$\text{SMART}_{\text{BERT-BASE}}$	94.8	93.2
$\text{MT-DNN-SMART}_{\text{BASEv0}}$	96.0	94.0
$\text{MT-DNN-SMART}_{\text{BASE}}$	96.1	94.2
$\text{BERT}_{\text{LARGE}}$	95.7	94.4
$\text{MT-DNN}_{\text{LARGE}}$	96.3	95.0
$\text{SMART}_{\text{BERT-LARGE}}$	96.2	94.7
$\text{MT-DNN-SMART}_{\text{LARGEv0}}$	**96.6**	**95.2**

Table 7: Results on the SNLI and SciTail dataset.

test our model on an adversarial natural language inference (ANLI) dataset (Nie et al., 2019).

We evaluate the performance of SMART on each subset (i.e., R1,R2,R3) of ANLI dev and test set. The results are presented in Table 6. Table 6 shows the results of training on combined NLI data (ANLI (Nie et al., 2019) + MNLI (Williams et al., 2018) + SNLI (Bowman et al., 2015) + FEVER (Thorne et al., 2018)) and training on only ANLI data. In the combined data setting, we observe that $\text{SMART}_{\text{RoBERTa-LARGE}}$ obtains the best performance compared with all the strong baselines, pushing benchmarks to 57.1%. In case of the $\text{RoBERTa}_{\text{LARGE}}$ baseline, $\text{SMART}_{\text{RoBERTa-LARGE}}$ outperforms 3.4% absolute improvement on dev and 7.4% absolute improvement on test, indicating the robustness of SMART. We obverse that in the ANLI-only setting, $\text{SMART}_{\text{RoBERTa-LARGE}}$ outperforms the strong $\text{RoBERTa}_{\text{LARGE}}$ baseline with a large margin, +5.2% (57.1% vs. 51.9%)

6 Conclusion

We propose a robust and efficient computation framework, SMART, for fine-tuning large scale pre-trained natural language models in a principled manner. The framework effectively alleviates the overfitting and aggressive updating issues in the fine-tuning stage. SMART includes two important ingredients: 1) smooth-inducing adversarial regularization; 2) Bregman proximal point optimization. Our empirical results suggest that SMART improves the performance on many NLP benchmarks (e.g., GLUE, SNLI, SciTail, ANLI) with the state-of-the-art pre-trained models (e.g., BERT, MT-DNN, RoBERTa). We also demonstrate that the proposed framework is applicable to domain adaptation and results in a significant performance improvement. Our proposed fine-tuning framework can be generalized to solve other transfer learning problems. We will explore this direction as future work.

Acknowledgments

We thank Jade Huang, Niao He, Chris Meek, Liyuan Liu, Yangfeng Ji, Pengchuan Zhang, Oleksandr Polozov, Chenguang Zhu and Keivn Duh for valuable discussions and comments, and Microsoft Research Technology Engineering team for setting up GPU machines. We also thank the anonymous reviewers for valuable discussions.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC09*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Andrew R Conn, Nicholas IM Gould, and Ph L Toint. 2000. *Trust region methods*, volume 1. Siam.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

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.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. pages 13042–13054.

Jonathan Eckstein. 1993. Nonlinear proximal point algorithms using bregman functions, with applications to convex programming. *Mathematics of Operations Research*, 18(1):202–226.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Osman Güler. 1991. On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419.

Osman Güler. 1992. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

Pengcheng He, Xiaodong Liu, Weizhu Chen, and Jianfeng Gao. 2019. A hybrid neural network model for commonsense reasoning. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 13–21.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Peter J Huber. 2011. *Robust statistics*. Springer.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for the winograd schema challenge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4837–4842.

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*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020a. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*.

Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. 2020b. The microsoft toolkit of multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:2002.07972*.

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

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.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. *Foundations of machine learning*. MIT press.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Neal Parikh, Stephen Boyd, et al. 2014. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *ACL 2019*, page 7.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Garvesh Raskutti and Sayan Mukherjee. 2015. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457.

R Tyrrell Rockafellar. 1976. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew 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.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.

Marc Teboulle. 1997. Convergence of proximal-like algorithms. *SIAM Journal on Optimization*, 7(4):1069–1083.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *EMNLP 2018*, page 353.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482.

Zhuosheng Zhang, Yuwei Wu, Zuchao Li, Shexia He, and Hai Zhao. 2018. I know what you want: Semantic learning for text comprehension.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding.

A Datasets

The GLUE benchmark, SNLI, SciTail and ANLI is briefly introduced in the following sections. The detailed description can be found in (Wang et al., 2018; Bowman et al., 2015; Khot et al., 2018; Nie et al., 2019). Table 8 summarizes the information of these tasks.

• **GLUE**. The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding (NLU) tasks. As shown in Table 8, it includes question answering (Rajpurkar et al., 2016), linguistic acceptability (Warstadt et al., 2019), sentiment analysis (Socher et al., 2013), text similarity (Cer et al., 2017), paraphrase detection (Dolan and Brockett, 2005), and natural language inference (NLI) (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Levesque et al., 2012; Williams et al., 2018). The diversity of the tasks makes GLUE very suitable for evaluating the generalization and robustness of NLU models.

• **SNLI**. The Stanford Natural Language Inference (SNLI) dataset contains 570k human annotated sentence pairs, in which the premises are drawn from the captions of the Flickr30 corpus and hypotheses are manually annotated (Bowman et al., 2015). This is the most widely used entailment dataset for NLI. The dataset is used only for domain adaptation in this study.

• **SciTail** This is a textual entailment dataset derived from a science question answering (SciQ) dataset (Khot et al., 2018). The task involves assessing whether a given premise entails a given hypothesis. In contrast to other entailment datasets mentioned previously, the hypotheses in SciTail are created from science questions while the corresponding answer candidates and premises come from relevant web sentences retrieved from a large corpus. As a result, these sentences are linguistically challenging and the lexical similarity of premise and hypothesis is often high, thus making SciTail particularly difficult. The dataset is used only for domain adaptation in this study.

• **ANLI**. The Adversarial Natural Language Inference (ANLI, Nie et al. (2019)) is a new large-scale NLI benchmark dataset, collected via an iterative, adversarial human-and-model-in-the-loop procedure. Particular, the data is selected to be difficult to the state-of-the-art models, including BERT and RoBERTa.

B Hyperparameters

As for the sensitivities of hyper-parameters, in general the performance of our method is not very sensitive to the choice of hyper-parameters as detailed below.

- We only observed slight differences in model performance when $\lambda_s \in [1, 10]$, $\mu \in [1, 10]$ and $\epsilon \in [10^{-5}, 10{-}4]$. When $\lambda_s \geq 100$, $\mu \geq 100$ or $\epsilon \geq 10^{-3}$, the regularization is unreasonably strong. When $\lambda_s \leq 0.1$, $\mu \leq 0.1$ or $\epsilon <= 1e-6$, the regularization is unreasonably weak.

- The algorithm is not sensitive to σ, any $\sigma \leq \epsilon$ works well.

- $p = \infty$ makes the size of perturbation constraint to be the same regardless of the number of dimensions. For $p = 2$, adversarial perturbation is sensitive to the number of dimensions (A higher dimension usually requires a larger perturbation), especially for sentences with different length. As a result, we need to make less tuning effort for $p = \infty$. For other values of p, the associated projections are computationally inefficient.

- We observed a minor improvement by using a larger S or a larger $T_{\widetilde{x}}$. The minor improvement comes with an increased cost of computation. When $S = T_{\widetilde{x}} = 1$, SMART requires 3 more forward passes and 3 more backward passes per iteration, compared with direct fine-tuning. In practice, it takes about 3 times the original training time. In terms of memory usage, it approximately doubles the GPU memory usage.

- We set $\beta = 0.99$ for the first 10% of the updates ($t <= 0.1T$) and $\beta = 0.999$ for the rest of the updates ($t > 0.1T$) following (Tarvainen and Valpola, 2017), which works well in practice.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr
Pairwise Text Classification						
SNLI	NLI	549k	9.8k	9.8k	3	Accuracy
SciTail	NLI	23.5k	1.3k	2.1k	2	Accuracy
ANLI	NLI	163k	3.2k	3.2k	3	Accuracy

Table 8: Summary of the four benchmarks: GLUE, SNLI, SciTail and ANLI.