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) research. Many existing 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 large capacity of pre-trained models, aggressive fine-tuning often causes the adapted model to overfit the data of downstream tasks and forget the knowledge of the pre-trained model. To address the above issue in a more principled manner, we propose a new computational framework for robust and efficient fine-tuning for pre-trained language models. Specifically, our proposed framework contains two important ingredients: 1. Smoothness-inducing regularization, which effectively manages the capacity of the model; 2. Bregman proximal point optimization, which is a class of trust-region methods and can prevent knowledge forgetting. Our experiments demonstrate that our proposed method achieves the state-of-the-art performance on multiple NLP benchmarks.[1]

1 Introduction

The success of natural language processing (NLP) techniques often relies on huge amounts of labeled data for many applications. However, large amounts of labeled data are usually prohibitive or expensive to obtain. To address these issues, researchers resort to transfer learning.

Transfer learning considers the scenario where we have limited labeled data for the target task, but we have a relevant task in a different domain with a large amount of data. The goal is to adapt the knowledge from the rich-resource task to the low-resource target task. 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 large-capacity model is trained for the high-resource task. The second stage is fine-tuning, where the large-capacity model is adapted to the target task with low resources.

For applications in NLP, most popular transfer learning methods usually pre-train a general language model (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019). Such a language model can capture general semantic information that can be further used in downstream NLP tasks. The general language model is particularly attractive, because it can be trained in a completely unsupervised manner using only unlabeled data, which is extremely cheap to fetch from the internet nowadays. For example, the well-known "Common Crawl project" is producing text data extracted from web pages at a rate of about 20TB per month. The resulting extremely large text corpus allows us to train extremely large neural network-based general language models. To the best of our knowledge, by far the largest language model, T5 developed by Google, has an enormous size of about 11 billion parameters (Raffel et al., 2019).

For the fine-tuning stage in the aforementioned NLP applications, researchers usually adopt stochastic gradient descent-type (SGD) algorithms such as ADAM (Kingma and Ba, 2014) and RAdam (Liu et al., 2019a) to adapt the pre-trained language model to the target tasks. Although a limited data is available, such a fine-tuning approach has achieved state-of-the-art performance in many popular NLP benchmarks (Devlin et al., 2018; Liu et al., 2019d; Yang et al., 2019; Lan et al., 2019; Dong et al., 2019; Raffel et al., 2019).

[*] Work was done during an internship at Microsoft.

[1] The code and pre-trained models will be made publicly available after the acceptence of the paper for publication.

However, due to limited data from the downstream tasks and the extremely large capacity of the pre-trained model, aggressive fine-tuning can easily make the adapted model seriously overfit the data of the downstream task and forget the knowledge of the pre-trained model. To mitigate such an issue, the aforementioned 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.

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

(I) Smoothness-inducing Adversarial Regularization, which can effectively manage the capacity of the pre-trained model. Popular examples include virtual adversarial training regularization proposed in Miyato et al. (2018), TRADES regularization in Zhang et al. (2019), and local linearity regularization in Qin et al. (2019).

(II) Bregman Proximal Point Optimization, which is a class of trust-region optimization methods and can prevent knowledge forgetting. Popular examples includes proximal point method proposed in Rockafellar (1976), generalized proximal point method (Teboulle, 1997; Eckstein, 1993), accelerated proximal point methods, and other variants (Güler, 1991, 1992; Parikh et al., 2014).

We compare our proposed method with several state-of-the-art competitors proposed in (Zhu et al., 2019; Liu et al., 2019b,d; 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 ensembling) to achieve three state-of-the-art results on GLUE, even compared with all existing ensemble models and the T5 model, which contains 11 billion parameters.

The rest of the paper is organized as follows: Section 2 briefly reviews existing work, Section 3 introduces our proposed method in detail, Section 4 presents experimental results on various natural language processing tasks, and Section 6 draws a conclusion.

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 $\mathcal{A}$. $\mathcal{D}_{KL}(P||Q) = \sum_k p_k \log \frac{p_k}{q_k}$ denotes the KL-divergence of two discrete distributions P and Q with the associated parameters p_k's and q_k's, respectively.

2 Background

We briefly introduce transformer-based language models, pre-training, fine-tuning, and a concurrent work.

The transformer models were originally proposed in Vaswani et al. (2017) for neural machine translation. Their superior performance motivated (Devlin et al., 2018) to propose a bidirectional transformer-based language model named BERT. Specifically, (Devlin et al., 2018) pre-trained such a 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., 2019), enlarging model size (Lan et al., 2019; Raffel et al., 2019), enlarging training corpora (Liu et al., 2019d; Yang et al., 2019; Raffel et al., 2019) and multi-tasking (Liu et al., 2019b,c).

After pre-training, the language model can be adapted to downstream tasks and further fine-tuned. Specifically, we replace the top layer of the language model with a task-specific head and then continue to train the model using SGD-type algorithms, e.g., ADAM, with the data of the target task. To prevent overfitting, existing heuristics include choosing a small learning rate or a triangular learning rate schedule, running ADAM only for a small number of iterations, and other tricks in (Howard and Ruder, 2018; Peters et al., 2019; Houlsby et al., 2019; Stickland and Murray, 2019), as mentioned earlier.

While this paper was being prepared, we found that a related fine-tuning method – FreeLB was independently proposed by Zhu et al. (2019). Specifically, they proposed a robust optimization-based adversarial training method to improve the fine-tuning performance. We give more detailed discussions on the difference between Zhu et al. (2019) and our method in Section 5.

3 The Proposed Method

We describe the proposed computational framework – called ***SMART***[2] 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 ***BR**regman **pR**roximal **p**oin**T** op**T**imization.*

3.1 Smoothness-Inducing Adversarial Regularization

We propose introducing strong 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 sentence and y_i's are the associated label, 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, $\mathcal{R}_{\mathrm{s}}(\theta)$ is the smoothness-inducing adversarial regularizer, and $\lambda_{\mathrm{s}} > 0$ is a tuning parameter. Here $\mathcal{R}_{\mathrm{s}}(\theta)$ can be chosen as either of the following two options:

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

$$[\mathbf{B}]\text{:}\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n}\sum_{i=1}^n \max_{\|\widetilde{x}_i - x_i\|_p \le \epsilon} |\ell(f(\widetilde{x}_i;\theta), y_i)$$
$$- \ell(f(x_i;\theta), y_i) - (\widetilde{x}_i - x_i)^\top \nabla_x \ell(f(\widetilde{x}_i;\theta), y_i)|,$$

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., $\ell_s(p, q) = (p-q)^2$. We remark that $[\mathbf{A}]$ was first used in (Miyato et al., 2018) for semi-supervised learning with $p = 2$, then used 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 $p = \infty$; $[\mathbf{B}]$ was first used in (Qin et al., 2019) for harnessing the adversarial examples in image classification with $p = \infty$. We further remark that the calculation of $\mathcal{R}_{\mathrm{s}}(\theta)$ involves a maximization problem and can be solved efficiently by projected gradient ascent.

All the aforementioned regularizers are essentially measuring the local Lipschitz continuity of f under the metric ℓ_s. Therefore, by minimizing the objective function in (1), we can encourage f to be smooth within the neighborhoods of all x_i's. Roughly speaking, the output of f does not change much if we inject a small perturbation (ℓ_p norm bounded by ϵ) to x_i. To the best of our knowledge, we are the first to apply the smoothness-inducing adversarial regularization to the fine-tuning of pre-trained language models.

3.2 Bregman Proximal Point Optimization

We propose to develop a class of Bregman proximal point optimization methods to solve (1). Specifically, we use the 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

$$\theta_{t+1} = \operatorname*{argmin}_\theta \mathcal{F}(\theta) + \mu\mathcal{D}_{\mathrm{Breg}}(\theta, \theta_t), \qquad (2)$$

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

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

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

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

et al., 2000). Consequently, the Bregman proximal point method can effectively prevent the model from aggressiving updating and retain the knowledge of the pre-trained model $f(\cdot;\theta_0)$.

Moreover, we remark that when

$$\mathcal{D}_{\text{Breg}}(\theta,\theta_t) = \|\theta-\theta_t\|_2^2,$$

the Bregman proximal point method is reduced to the standard proximal point method. Existing literature has shown that the Bregman proximal point method is capable of adapting to the information geometry of machine learning models and achieving better computational performance than the standard proximal point method in many applications (Raskutti and Mukherjee, 2015).

3.3 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 accelerated Bregman proximal point (ABPP) method takes

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

where $\widetilde{\theta}_t = (1-\beta)\theta_t + \beta\widetilde{\theta}_{t-1}$ and $\beta \in (0,1)$ is the momentum parameter. We remark that the ABPP 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 ABPP method in Algorithm 1.

4 Experiment

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. 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. Table 1 summarizes the information of these tasks.

4.1 Datasets

• **GLUE**. The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding (NLU) tasks.

Algorithm 1 SMART: We use the smoothness-inducing adversarial regularizer [**A**] with $p = \infty$ and the accelerated Bregman proximal point method. For notational 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 Update denotes the update rule of the ADAM method for optimizing (2).

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 update $\widetilde{x}_i$'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
            $\widetilde{g}_i \leftarrow \frac{g_i(\widetilde{x}_i,\bar{\theta}_s)}{\|g_i(\widetilde{x}_i,\bar{\theta}_s)\|_\infty}$
            $\widetilde{x}_i \leftarrow \Pi_{\|\widetilde{x}_i - x\|\le\epsilon}(\widetilde{x}_i + \eta\widetilde{g}_i)$
        end for
        $\bar{\theta}_{s+1} \leftarrow \mathrm{Update}(\bar{\theta}_s)$ based on $\mathcal{B}$
    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

As shown in Table 1, it includes question answering (Rajpurkar et al., 2016), linguistic acceptability (Warstadt et al., 2018), 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

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

Table 1: Summary of the three benchmarks: GLUE, SNLI and SciTail.

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.

4.2 Implementation Details

Our implementation of SMART is based on BERT[3], RoBERTa[4], MT-DNN[5] and HNN[6]. We used ADAM (Kingma and Ba, 2014) and RADAM (Liu et al., 2019a) 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 over 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, the perturbation size ϵ is set to 10^{-5} and number of iterations $T_{\tilde{x}}$ is set 1. We set $\mu = 1$ and $\lambda_s = 1$. The learning rate η in Algorithm 1 is set to 10^{-3}. Lastly, we simply set $S = 1, T_{\tilde{x}} = 1$ in Algorithm 1.

4.3 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. To verify the generalization of our SMART, we evaluate our framework on two pre-trained models, the base BERT model (Devlin et al., 2018) and the large RoBERTa model (Liu et al., 2019d), which are available publicly. Most of our analyses are done with the base BERT to make our results comparable to other work, since the base BERT 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 large RoBERTa model.

1. BERT (Devlin et al., 2018): This is the base BERT model released by the authors. In (Devlin et al., 2018), authors only reported 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

Model	MNLI-m/mm Acc	QQP Acc/F1	RTE Acc	QNLI Acc	MRPC Acc/F1	CoLA Mcc	SST Acc	STS-B P/S Corr
BERT BASE								
BERT(Devlin et al., 2018)	84.4/-	-	-	88.4	-/86.7	-	92.7	-
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
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**
RoBERTa LARGE								
RoBERTa(Liu et al., 2019d)	90.2/-	92.2/-	86.6	94.7	-/90.9	68.0	96.4	92.4/-
PGD(Zhu et al., 2019)	90.5/-	92.5/-	87.4	94.9	-/90.9	69.7	96.4	92.4/-
FreeAT(Zhu et al., 2019)	90.0/-	92.5/-	86.7	94.7	-/90.7	68.8	96.1	92.4/-
FreeLB(Zhu et al., 2019)	90.6/-	**92.6**/-	88.1	95.0	-/91.4	**71.1**	96.7	92.7/-
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 2: 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	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												
MT-DNN[1]	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9/87.4	96.0	86.3	89.0	42.8	87.6	335M
XLNet[2]	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	90.4	47.5	88.4	360M
RoBERTa[3]	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[4]	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[5]	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[6]	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*
Single Model												
BERT$_{\text{LARGE}}$[7]	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[1]	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
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[9]	50.2	88.4	356M

Table 3: GLUE test set results scored using the GLUE evaluation server. The number below each task denotes the number of training examples. The state-of-the-art results are in **bold**. All the results were obtained from https://gluebenchmark.com/leaderboard on November 2, 2019. The mark "-" denotes the missed result of the latest GLUE version. The official GLUE new rules do not allow to use a pairwise ranking objective for QNLI (Liu et al., 2019c), thus SMART only uses the classification objective. Model references: [1] (Liu et al., 2019c); [2] (Yang et al., 2019); [3] (Liu et al., 2019d); [4](Zhu et al., 2019); [5](Wang et al., 2019); [6](Lan et al., 2019); [7] (Devlin et al., 2018); [8] (Raffel et al., 2019) and [9] (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 clever parameter sharing).

development results on a few tasks, thus we reproduced the baseline results, which are denoted by **BERT$_{\text{ReImp}}$**.

2. RoBERTa (Liu et al., 2019d): the large RoBERTa released by authors, and these are reported results on GLUE dev.

3. PGD (Madry et al., 2017): It is a standard Projected Gradient Descent (PGD) adversarial learning approach built on top of the large RoBERTa.

4. FreeAT (Shafahi et al., 2019): an optimized adversarial learning by recycling gradient information built on top of the large RoBERTa.

5. FreeLB (Zhu et al., 2019): an improved version of FreeAT, also used RoBERTa as its encoder.

6. SMART: our proposed method as described in section 3. We use both the base BERT model (SMART$_{\text{BERT}}$) and the large RoBERTa model (SMART$_{\text{RoBERTa}}$) as our encoder to verify the generalization of SMART.

The main results are reported in Table 2. This table can be clustered into two groups based on different encoders: the base BERT model (the first group) and the large RoBERTa 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 (Devlin et al., 2018). 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. Note that our model is built on the top of BERT and RoBERTa and this improvement demonstrates the effectiveness and generalization of SMART. For example, 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 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 is more robust and that alleviates the domain shifting issue. Furthermore, on the small tasks, the improvement of SMART is even larger: e.g., comparing with BERT, $\text{SMART}_{\text{BERT}}$ obtained 71.2% (vs 63.5%) on RTE and CoLA 59.1% (vs 54.7%) 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 (Zhu et al., 2019). From the bottom rows in Table 2, SMART outperforms PGD and FreeAT across the all 8 GLUE tasks. Comparing with the current state-of-the-art adversarial training model, FreeLB (Zhu et al., 2019), 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 robustness of our model. Note that we only set $T_{\tilde{x}} = 1$ in Algorithm 1 and we observed that a larger $T_{\tilde{x}}$ helps to improve the performance, which will be explored as future work.

Table 3 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.

4.4 Ablation Study

Model	MNLI	RTE	QNLI	SST	MRPC
	Acc	Acc	Acc	Acc	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 4: Ablation study of SMART on 5 GLUE tasks. Note that all models used the BERT base model as their encoder.

Note that due to limitation of time and computational resources, all the experiments reported below are based on the **base BERT** 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 base as the encoder for fast training. Furthermore, we also include the BERT 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 notated as $-\mathcal{D}_{\text{Breg}}$.

The results are reported in Table 4. 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 regu-

[7] In our experiments, we use BERT referring the BERT base model, which has 110 million parameters, and RoBERTa referring the RoBERTa large model, which has 356 million parameters, unless stated otherwise.

larization 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 demostrating the effectiveness of each module.

4.5 SMART vs Multi-task Learning

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
SMART-MT-DNN$_{\text{v0}}$	**85.7**	80.2	**92.0**	93.3	91.5
SMART-MT-DNN	**85.7**	**81.2**	**92.0**	93.5	**91.7**

Table 5: Comparison between SMART and Multi-Task Learning.

It has been shown that multi-task learning (MTL) (Caruana, 1997; Liu et al., 2015, 2019c) 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., 2019c), we first "pre-trained" shared embeddings using MTL with SMART, denoted as **SMART-MT-DNN** [8], and then adapted the training data on each task on top of the shared embeddings. We also include a baseline which finetuned each task on the publicly released MT-DNN checkpoint [9], which is indicated as **SMART-MT-DNN$_{\text{v0}}$**.

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, SMART-MT-DNN$_{\text{v0}}$ 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., 2019c). We observe that SMART-MT-DNN outperformes SMART-MT-DNN$_{\text{v0}}$ and MT-DNN across all 5 tasks (except MT-DNN on SST) showing that SMART improves the generalization of MTL.

4.6 Domain Adaptation on SNLI and SciTail

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
SMART-MT-DNN	**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
SMART-MT-DNN	**82.3**	**88.6**	**91.3**	**96.1**

Table 6: Domain adaptation results on SNLI and SciTail.

In real-world applications, collecting labeled training data for new domains or tasks is prohibitively expensive. Thus, one of the most important criteria of building practical system is how to adapt it quickly to new tasks and domains. In this section, we evaluate the model using the aforementioned criteria. Following (Liu et al., 2019c), 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 6. We observe that both MT-DNN and SMART-MT-DNN significantly outperform the BERT baseline. Comparing with MT-DNN, SMART-MT-DNN also achieves some improvements indicating the robustness of SMART.

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

[8] Due to time limitation, we only trained jointly using MTL on MNLI, RTE, QNLI, SST and MRPC, while MT-DNN was trained on the whole GLUE tasks excpet CoLA. We observed that adding more GLUE tasks improved generalization of the model leading to a better result.

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

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

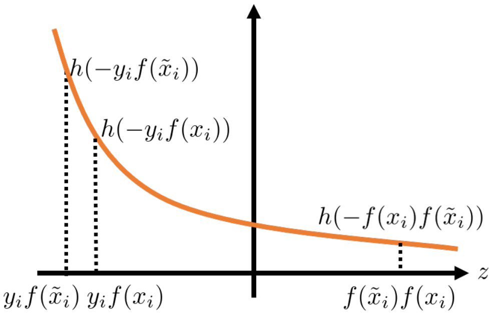
Table 7: Results on the SNLI and SciTail dataset.

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 Discussion

We now highlight the difference between adversarial training used in FreeLB (Zhu et al., 2019) and smoothness-inducing adversarial regularization used in SMART. The adversarial training encourages the model to predict the annotated label for adversarial samples. In contrast, our regularization encourages the model to make consistent predictions within the neighborhood. Existing literature has shown that the adversarial training has a regularization effect, which can improve generalization in supervise learning tasks (Xu and Mannor, 2012). Though both methods can improve the generalization of the model, the adversarial training method is often sensitive to the noisy labels, which are common in NLP applications, especially when there exists language ambiguity. Here we provide a simple illustration using binary classification as an example. Specifically, we consider a data point (x_i, y_i), where $y_i \in \{-1, 1\}$ is a noisy/wrong label. The true label should be y_i. Here we assume the model f makes a correct prediction of the true label $-y_i$, i.e., sign$(f(x_i) \neq y_i$. We denote the composite loss functions of the adversarial training and our method for (x_i, y_i) as

Figure 1: The logistic loss of adversarial training and smoothness-inducing regularization.



$$\ell_{\text{adv}}(x_i, y_i) = h(-y_i f(x_i)) + \lambda h(-y_i f(\widetilde{x}_i)),$$
$$\ell_{\text{S}}(x_i, y_i) = h(-y_i f(x_i)) + \lambda h(-f(x_i) f(\widetilde{x}_i)),$$

respectively, where $h(z) = \log(1 + \exp(-z))$ and $\widetilde{x}_i$ is the adversarial sample. Figure 1 illustrates the difference of two composite loss functions. As can be seen, our regularization only penalizes a small discrepancy between $-f(x_i)$ and $f(\widetilde{x}_i))$. In contrast, the adversarial training method injects an adversarial perturbs to x_i and yields $h(-yf(\widetilde{x}_i)) > h(-yf(x_i))$. Therefore, the adversarial training imposes a much larger penalty to the noisy label than that of our method. This explains why our method suffers less from noisy labels than the adversarial training in FreeLB.

6 Conclusion

We propose a robust and efficient computation framework, SMART, for fine-tuning large scale natural language models in a more principle manner. The framework is designed to alleviate the overfitting and forgetting issues in the fine-tuning procedure. 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) 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 result in a significant performance gain. Our proposed fine-tuning framework is well structured and can be generalized to solve more transfer learning problems. We will explore this direction as future work.

Acknowledgments

We thank Jade Huang, Niao He, Chris Meek, and Liyuan Liu for valuable discussions and comments, Microsoft Research Technology Engineering team for setting up GPU machines.

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, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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. *arXiv preprint arXiv:1905.03197*.

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.

Pengcheng He, Xiaodong Liu, Weizhu Chen, and Jianfeng Gao. 2019. A hybrid neural network model for commonsense reasoning. *arXiv preprint arXiv:1907.11983*.

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. *arXiv preprint arXiv:1902.00751*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

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 winograd schema challenge. *arXiv preprint arXiv:1905.06290*.

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. 2019a. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

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. 2019b. 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. 2019c. 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.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

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.

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 Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Alhussein Fawzi, Soham De, Robert Stanforth, Pushmeet Kohli, et al. 2019. Adversarial robustness through local linearization. *arXiv preprint arXiv:1907.02610*.

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.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *arXiv preprint arXiv:1904.12843*.

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. *arXiv preprint arXiv:1902.02671*.

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.

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. *arXiv preprint arXiv:1804.07461*.

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. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

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.

Huan Xu and Shie Mannor. 2012. Robustness and generalization. *Machine learning*, 86(3):391–423.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*.

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, Thomas Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*.