PTP: Boosting Stability and Performance of Prompt Tuning with Perturbation-Based Regularizer

by

Lichang Chen

Bachelor of Engineering, Zhejiang University, 2020

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2023

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Lichang Chen

It was defended on

April 12, 2023

and approved by

Wei Gao, PhD, Professor, DEPARTMENT OF ELECTRIC AND COMPUTER ENGINEERING

Liang Zhan, PhD, Professor, DEPARTMENT OF ELECTRIC AND COMPUTER ENGINEERING

Thesis Advisor: Heng Huang, PhD, Professor, DEPARTMENT OF ELECTRIC AND COMPUTER ENGINEERING

PTP: Boosting Stability and Performance of Prompt Tuning with Perturbation-Based Regularizer

Lichang Chen, M.S.

University of Pittsburgh, 2023

Recent studies show that prompt tuning can better leverage the power of large language models than fine-tuning on downstream natural language understanding tasks. However, the existing prompt tuning methods have training instability issues, as the variance of scores under different random seeds is quite large. To address this critical problem, we first investigate and find that the loss landscape of vanilla prompt tuning is precipitous when it is visualized, where a slight change of input data can cause a big fluctuation in the loss landscape. This is an essential factor that leads to the instability of prompt tuning. Based on this observation, we introduce perturbation-based regularizers, which can smooth the loss landscape, into prompt tuning. We propose a new algorithm, called Prompt Tuning with Perturbation-based regularizer (PTP), which can not only alleviate training instability dramatically but also boost the performance of prompt tuning. We design two kinds of perturbation-based regularizers, including random-noise-based and adversarial-based. In particular, our proposed perturbations are flexible on both text space and embedding space. Extensive experiments show the effectiveness of our proposed methods in stabilizing the training. Our new algorithms improve the state-of-the-art prompt tuning methods by 1.94% and 2.34% on SuperGLUE and FewGLUE benchmarks, respectively.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

Releasing the burden of training models from scratch while keeping the outstanding performance on downstream tasks, pretrained Language Models (LMs) brought NLP to a new era [32, 15, 37]. Since BERT [39], fine-tuning all the parameters of pretrained LMs becomes a common practice. However, it is memory-consuming to store a copy of the entire LM for each downstream task due to the number of parameters in LM can be 10B or even 100B [37, 2].

Recently, inspired by the success of GPT-3 [2] on few-shot and zero-shot learning with manually created prompts, there has been a surging interest in prompting that freezes pretrained LM and wraps up the input sequence with natural language templates. However, natural language prompts are handcrafted by experts and the performance is not comparable with fine-tuning methods. To tackle it, [21, 24] proposed prompt tuning, which prepends the input sequence with continuous embeddings and only tunes these embeddings during training. [27, 26] verified the effectiveness of prompt tuning on natural language understanding (NLU) tasks under both few-shot learning and supervised learning settings, which is comparable to the fine-tuning methods but with much fewer (1000× less) task-specific tunable parameters. However, under different random seeds, we observe that the current prompt tuning methods suffer from a high variance of scores, which indicates they suffer from training instability issues.

To investigate the factor that causes the instability of prompt tuning, we visualize the loss landscape of the vanilla prompt tuning and observe that there exist many sharp crests, as shown in Figure 2(a), which harms the training stability. Motivated by the recent study [3] which shows that perturbation-based regularizers are powerful tools to smooth the loss landscape and stabilize the training of machine learning systems, we introduce them into prompt tuning to address the lack of stability and generalization issues. To be specific, we propose Prompt Tuning with Perturbation-based regularizer (PTP) algorithm to make the training stable and boost the performance of prompt tuning.

Specifically, we consider two kinds of perturbations in PTP, Random-Noise-based (PTP-

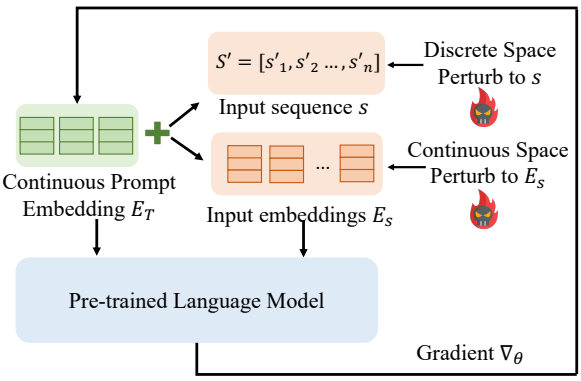


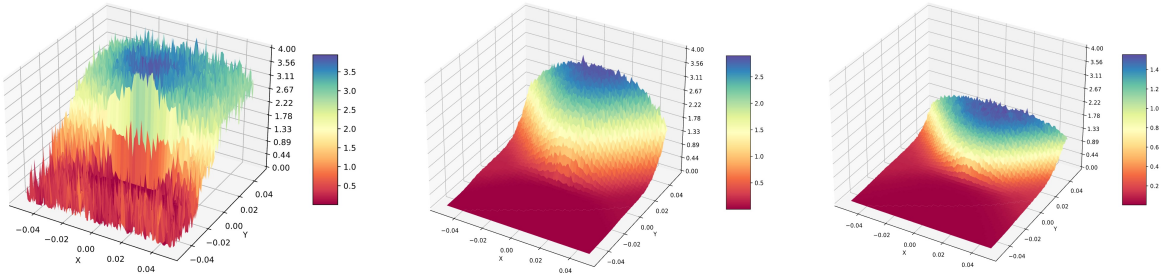
Figure 1: A simple pipeline of our **P**ropmt **T**uning with **P**erturbation-based regularizor (PTP) algorithm.

RN) perturbation and ADVersarial-based (PTP-ADV) perturbation. PTP-RN is motivated by randomized smoothing [6], which applies the neighborhood averaging and makes the neural network smoother while PTP-ADV is motivated by adversarial training, a method proposed to make the predictor capable of resisting the adversarial examples [12] as well as boosting the clean accuracy of the predictors [41, 44]. Moreover, in order to bring more flexibility and make our exploration more comprehensive, we apply perturbations to both text (discrete) and embedding (continuous) space, as depicted in Figure 1.

In the experiments, we conduct extensive experiments to evaluate our proposed algorithms, PTP-RN and PTP-ADV, on SuperGLUE [40] and FewGLUE [36] benchmark. By applying the PTP algorithm on text or embedding space to the existing prompt tuning methods, we can boost the performance of prompt tuning on SuperGLUE and FewGLUE benchmarks by 1.94% and 2.34% as well as make prompt tuning more stable. We also present a comprehensive ablation study and analysis of our algorithms with different perturbations.

Our contributions can be summarized as:

- We propose a new PTP algorithm to tackle the training instability problem in prompt tuning, which can also boost the performance. Together with PTP algorithm, we design two types of perturbations as our implicit regularizers, which are Random-Noise-based



(a) Original (a) PTP-RN (b) PTP-ADV

Figure 2: The loss landscapes on different continuous prompt tuning procedures. The X-axis and Y-axis denote the magnitude of perturbations (gradient direction) and another random orthogonal direction. Z-axis represents the cross-entropy loss of the different training methods.

perturbation (PTP-RN) and ADVersarial-based perturbation (PTP-ADV).

- Moreover, as depicted in Figure 1, our proposed PTP-ADV and PTP-RN can be applied to both text space and embedding space, which makes our perturbations more flexible.
- We conduct extensive experiments to evaluate the effectiveness of our algorithms on SuperGLUE and FewGLUE benchmarks. The experimental results demonstrate our proposed PTP algorithm can boost the standard performance of prompt tuning on FewGLUE and SuperGLUE by 2.34% and 1.94%, respectively. It also shows the great power of our algorithm in improving training stability.

2.0 Preliminaries

2.1 Prompt Tuning

2.1.1 Discrete Prompt.

Discrete prompt, also known as hard prompt [25, 8, 16, 14], is typically a template composed of task descriptions and original input texts. [2] created templates for GPT-3 based on their introspection and make it suitable for various downstream tasks, such as machine translation, QA, *etc.* Utilizing discrete prompts, they can achieve stunning results on NLU tasks under few-shot learning settings. By employing the predefined templates and converting the tasks into cloze questions, [35, 36] showed that even with 'greener' backbone [20], which has 100x fewer parameters than GPT-3, they can also reach prevailing results on the few-shot version of SuperGLUE benchmark [40] (also known as FewGLUE). [27] utilized the existing discrete templates and tuned embeddings of the selected tokens, which achieves SOTA results on FewGLUE. As the few-shot scenario is common and useful, in this paper, we also test the few-shot learning ability of our PTP in our experiments.

2.1.2 Continuous Prompt Tuning.

As prompts aim to boost LM's performance, it is not necessary to make tokens of prompts interpretable. Without the limit of tokens being natural words, Li *et al.* [24] proposed to prepend a series of tunable embeddings E_T to the input embedding sequence as prompt and optimize them with training data, which provides in-context information for LMs to condition on. [21] prepended a sequence of special tokens T to the input sequence, and similarly, they tune the embeddings E_T (embeddings of special tokens) on the downstream tasks. Moreover, to further leverage the power of prompt embeddings, [26] presented PT2, a method that adds the trainable prompt embeddings to every layer of pretrained LMs as prefix embeddings. To keep the consistency of the notation, we also apply the same prompt embedding representation E_T to represent trainable prompts in every layer of LMs.

2.2 Adversarial Training

Over the past few years, Adversarial Training (AT) has demonstrated impressive results in improving model robustness [12, 38, 1]. AT can be formulated as a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x_i,y_i)\sim\mathbb{D}} \left[\max_{\|\delta\|_p \leq \epsilon} \mathcal{L}\left(\theta, x+\delta, y\right) \right], \tag{2.1}$$

where $\mathcal{L}$ is the loss function, $\|\cdot\|_p$ represents ℓ_p-norm distance and ϵ denotes the perturbation budget. Madry *et al.* [29] proposed PGD algorithm to compute an adversarial example (inner maximization problem) iteratively as:

$$\delta^{t+1} = \Pi_{\|\delta\|_\infty \leq \epsilon} \left(\delta^t + \alpha \operatorname{sign}\left(\nabla_\delta L(\theta, \delta^t, y) \right) \right), \tag{2.2}$$

where t is the iteration step; $\Pi_{\|\delta\|_\infty \leq \epsilon}$ projects perturbation δ into ϵ-ball.

Besides enhancing the robustness against adversarial examples, adversarial training has been shown great power in boosting the standard performance of image classifier [41], visual language representation learning [11], and GNN [19]. In this paper, we also apply a similar idea to prompt tuning and focus on boosting its performance rather than its adversarial robustness.

Different from adversarial attacks on images, in NLP attack, due to the discrete nature of text space, it is typically formulated as a combinatorial optimization problem to create adversarial input sequence s', which is classically solved by heuristic search while maintaining the semantic similarity of the original input sequence s [23, 33, 30]. However, the searching algorithms of adversarial attacks, such as beam search [10], greedy search based on word importance [33], and deletion-based searching [17], are usually slow because of the high computation cost on sentence encoding and large search space [42]. [43] proposed A2T algorithm to accelerate the heuristic search process, which replaces the slow USE encoder [23] with DistilBERT [34] to calculate the cosine similarity between original input text and perturbed text, and they obtain significant speedup comparing to the textfooler [17]. Thus, in this paper, we adapt the attacking algorithm in [43] to generate noises in text space.

Table 1: Re-implementation results of PT2 [26] with RoBERTa-large backbone on SuperGLUE benchmark. Acc.: mean accuracy. Var.: variance score computed by 5 runs with different random seeds.

	RTE	BoolQ	WiC
Acc.	87.7 ± 1.81	83.9 ± 0.92	72.0 ± 1.38
Var.	1.45	0.74	1.16

2.2.1 Preliminary

Before introducing the proposed algorithms, we first briefly describe the word to embedding process of LMs as well as the final embedding input of the continuous prompt tuning. Given the n word input sequence $s = [s_1, \ldots, s_n]$ and word to embedding function f_V, where V denotes embedding matrix, the input embedding $E_s = [e(s_1), \ldots, e(s_n)] \in \mathbb{R}^{n \times d}$ can be obtained by $E_s = f_V(s)$, where d denotes the dimension of the word embedding. Let continuous prompts be $E_T = [e_t^1, \ldots, e_t^m] \in \mathbb{R}^{m \times d}$, the update of E_T can be either directly, or through a reparameterization encoder P like MLP/LSTM, $P(E_T) = [h_0, \ldots, h_m] \in \mathbb{R}^{m \times d}$. For simplicity, we still use E_T to represent the output of $P(E_T)$, and the final input embedding sequence is written as $[E_T; E_s] \in \mathbb{R}^{(m+n) \times d}$.

The training objective of continuous prompt tuning can be formulated as:

$$\min_{\theta} \mathbb{E}_{(s,y)\sim\mathbb{D}} \left[\mathcal{L} \left(\mathcal{M} \left(\theta, s, y \right) \right) \right], \tag{2.3}$$

where M denotes LM; θ represents the trainable parameters of E_T and prompt encoder P while $\mathbb{D}$ is the underlying data distribution.

2.2.2 Proposed Formulation

Although continuous prompt tuning [26] could achieve comparable performance with the fine-tuning method by only using 0.1% to 3% trainable parameters, it suffers from unstable

Algorithm 1: PGD on prompt tuning

Require: Perturbation iteration n and size α. The bound of perturbation ϵ ;

for *epoch* $= 1, \ldots, n$ **do**

  E_s.requires_grad $\leftarrow$ True;

  $\hat{y}_i \leftarrow \arg\max_y \left(\Pr\left[y|\mathcal{M}\left(I'\right)\right]\right)$;

  $\mathcal{L}(\hat{y}, y)$.backward();

  $E'_s \leftarrow E_s + \alpha * E_s.grad.sign()$;

  $\delta = \Pi_{\|\delta\|_\infty \leq \epsilon}(E'_s - E_s)$;

  $E'_s \leftarrow E_s + \delta$;

  $\mathcal{M}$.zero_grad();

return E'_s

training issues: as shown in Table 1, even only changing the random seed in different runs, the final performance is very unstable. To investigate the issue, we plot the loss landscape of the vanilla prompt tuning as Figure 2 and observe sharp crests in a small local region with a small noise, which means a small perturbation on embedding space would cause a significant reduction in the final accuracy. It is also known as the training instability problem [3]. To address this challenge in prompt tuning, we propose perturbation-based regularizers to force the loss landscape to be smooth. Specifically, we introduce two versions of perturbation-based regularizers that can be formulated as follows:

$$\begin{aligned} &\min_\theta \mathbb{E}_{(s,y)\sim\mathbb{D}}\left[\mathcal{L}\left(\mathcal{M}\left(\theta, s+\delta, y\right)\right)\right], \text{s.t.} \\ &\text{PTP-RN: } \delta \sim \mathcal{N} \\ &\text{PTP-ADV: } \delta = \max_{\|\delta\|\leq\epsilon} \mathcal{L}\left(\theta, s+\delta, y\right), \end{aligned} \tag{2.4}$$

where $\mathcal{N}$ denotes Gaussian distribution. For PTP-RN, we minimize θ under small random perturbation, aiming to force the model to focus on perturbed pair $(s+\delta, y)$ and have a robust prediction within the neighborhood of s. It is related to the idea of randomized smoothing [6], which obtains a smoother predictor via randomly averaging the neighborhood of the given function. For PTP-ADV, the perturbation δ is generated by adversarial attack

algorithms such as PGD [29], A2T [43], and the worst-case training loss is minimized under small perturbation bounded by ϵ. The idea is motivated by adversarial training, which is usually applied as a form of adversarial defense. [12, 4]

Since we are the first to investigate the training stability issue of the prompt tuning and it is still unknown which space to inject perturbation δ is better, we apply it on both text and embedding space to bring more flexibility and make our exploration more comprehensive. The perturbed sequence s' or the perturbed embedding E'_s can be obtained as

$$\begin{aligned} s' &= s + \delta, \\ E'_s &= E_s + \delta, \end{aligned} \tag{2.5}$$

where s, E_s denote the input sequence and the input embedding, respectively. It is worth noticing that if the perturbation is on s (text space), through f_V, the perturbed s' will be converted into input embedding, which is also denoted as E'_s.

The main idea of our proposed formulation is that we force our algorithm to not only learn from the clean data pair (s, y) but also perturbed data pair $(s + \delta, y)$ to make the training more smooth.

2.2.3 PTP-RN

2.2.3.1 Embedding Space (RG Perturbation)

In embedding space, how to create perturbed examples is still an unsolved problem. But since the ultimate effects of PTP-RN are the only thing we care about, not the interpretability, it is feasible for us to add random-noise-based perturbation on word embeddings. Given the embeddings of the input sequence $E_s = [e(s_0), e(s_1), \ldots, e(s_n)]$, where $e(s_i) \in \mathbb{R}^d$, PTP-RN in embedding space samples δ from Gaussian distribution and randomly selects some embeddings to perturb, which make sure. The perturbation δ can be formulated as:

$$\begin{aligned} E'_s &= E_s + \delta, \text{s.t.} \\ \delta &= \{\delta_1, 0, \ldots, \delta_i, 0\}, \end{aligned} \tag{2.6}$$

where $\delta \in \mathbb{R}^{n \times d}$ has the same length as the input embeddings; i denotes the number of embeddings being perturbed and $\delta_{n=1,\ldots,i} \sim \mathcal{N}(0, \sigma \mathbb{I}_d)$, with d denoting the dimension of the

word embedding and σ controlling the magnitude of perturbation. We represent PTP-RN on embedding space as PTP+RG.

2.2.3.2 Text Space (RM Perturbation)

In text space, similarly, our goal is to create label-preserving and perturbed input data to augment the training data and make the training stable. Given an input sequence s, PTP-RN randomly selects some tokens and converts them into [MASK]. The perturbed sequence s' can be formulated as:

$$s' = \{s_0, [\text{MASK}], \ldots, [\text{MASK}], s_n\}, \tag{2.7}$$

where we perturb i tokens. It should be noticed that unlike BERT pretraining process [39], where the model predicts the label of [MASK], our model will not predict anything on the tokens we mask and we just use [MASK] token as a perturbation on discrete space. PTP+RM is used to denote PTP-RN on text embedding space.

2.2.4 PTP-ADV

2.2.4.1 Embedding Space (PGD Perturbation)

Different from previous PGD training methods which focus on improving the models' robustness, we aim to smooth the loss landscapes and boost the performance of prompt tuning by adding adversarial-based regularization. Given the embedding sequence E_s, PTP-ADV adopts multi-step PGD to generate perturbations on embedding space. The perturbation δ is computed iteratively as:

$$\begin{aligned} E'_s &= E_s + \delta^t, \text{s.t.} \\ \delta^t &= \Pi_{\|\delta\|_\infty \leq \epsilon} \left(\delta^{t-1} + \alpha \nabla_\delta L\right) \end{aligned} \tag{2.8}$$

where δ^t denotes the t-th iterations of PGD perturbation and it will be added to the input embedding sequence E_s after all the iterations are finished. Algorithm 1 shows the implementation details of our PGD attack on prompt tuning. PTP+PGD is applied to denote our PTP-ADV algorithm with perturbation on embedding space.

2.2.4.2 Text Space (A2T Perturbation)

Furthermore, to enhance the flexibility of PTP-ADV and boost model generalization ability, we apply it to the text space: PTP-ADV adopts the attack algorithm in A2T [43] to generate its perturbation δ, which is an algorithm composed of NLP attack and adversarial training. Given the input sequence s, the perturbed sequence s', with A2T perturbation, can be represented as:

$$s' = \{s_0, s'_1, \ldots, s'_{n-1}, s_n\}, \tag{2.9}$$

where s'_i denotes the perturbed word. For simplicity, we also call it PTP+A2T.

Algorithm **??** provides the details about the whole training process of PTP algorithm. In the standard prompt tuning, the input of LM is composed of prompt embedding E_T and input embedding E_s, which is denoted as $I = [E_T; E_s]$. After LM gives a prediction of I, we backpropagate the loss to update E_T. In training with perturbed data part (Line 10-16, Algorithm **??**), the discrete or continuous space perturbations of input data are generated by PTP-RN or PTP-ADV firstly. Then the perturbed input is employed to conduct training with original label y, which also plays a data-augmentation role to boost the performance of the prompt tuning.

We conducted empirical studies on two popular natural language understanding (NLU) benchmarks: SuperGLUE benchmark [40] and FewGLUE benchmark [36]. We tested the proposed framework in both fully-supervised and few-shot settings to verify the effectiveness of our proposed PTP-RN and PTP-ADV algorithm with perturbations on both text and embedding space.

Algorithm 2: PTP

Require: Prompt embeddings E_T; input embeddings E_s; trainable parameter θ for prompt encoder P and E_T; Training data D; Pre-trained LM $\mathcal{M}$; Loss function $\mathcal{L}$;

Initialize parameters θ;

for *epoch* $= 1, \ldots, K$ **do**

  /* *standard prompt tuning* */

  Sample a minibatch data (s, y) from D;

  Θ.requires_grad $\leftarrow$ True;

  $I \leftarrow [E_T; E_s]$;

  $\hat{y} \leftarrow arg\max_y (\Pr[y|\mathcal{M}(I)])$;

  $\mathcal{L}(\hat{y}, y)$.backward() and update E_T ;

  /* *training with perturbed data* */

  Θ.requires_grad $\leftarrow$ False;

  Apply PTP-RN or PTP-ADV to s or E_s;

  $I' \leftarrow [E_T; E'_s]$;

  Θ.requires_grad $\leftarrow$ True;

  $\hat{y}_i \leftarrow arg\max_y (\Pr[y|\mathcal{M}(I')])$;

  $\mathcal{L}(\hat{y}, y)$.backward() and update E_T;

Table 2: Results of our proposed PTP algorithm in fully-supervised learning settings. We employ the large-size version of BERT and RoBERTa models (BERT-Large size: 335M and RoBERTa-large size: 355M, respectively). We use bold font to mark the best and red subscript to mark the improvement compared to the PT2.

Method	BoolQ		CB		WiC		RTE	
	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
FT	77.7	86.9	94.6	98.2	74.9	75.6	70.4	86.6
PT2	75.8	84.8	94.6	100	75.1	73.4	78.3	89.5
PTP+A2T	76.4	85.7	94.5	99.6	75.8	72.9	78.6	89.7
PTP+RG	77.3	86.2	95.8	99.8	**76.7**	75.5	79.9	90.6
PTP+RM	77.4	85.9	95.7	100	76.4	75.2	79.6	91.6
PTP+PGD	**78.3**	**86.7**	**96.1**	**100**	76.6	**75.7**	**80.3**	**92.0**

Method	COPA		MultiRC(F1a)		ReCoRD		WSC	
	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
FT	69.0	94.0	70.5	85.7	70.6	89.0	68.3	63.5
PT2	73.0	93.0	70.6	82.5	72.8	89.3	68.3	63.5
PTP+A2T	73.3	93.2	71.4	82.6	73.6	89.7	68.5	63.8
PTP+RG	**75.1**	93.9	72.6	**84.9**	74.9	90.5	69.4	65.0
PTP+RM	74.6	93.8	72.9	84.4	74.8	90.6	69.2	64.8
PTP+PGD	74.7	**94.1**	**73.4**	84.6	**75.1**	**91.9**	**69.7**	**65.0**

Table 3: Results of our PTP algorithm in Few-shot learning(32 training examples) settings. PT: P-tuning [27] and the backbone LM is alberta-xxl-v2. We use **bold** font to mark the best. The red subscript denotes the increase of our method compared with the baseline method PT. Dev 32: development set contains 32 unused examples from the training set, same as [27]. Full Dev: original development set.

(Dev 32)	BoolQ	CB	WiC	RTE	MultiRC		WSC	COPA
Method	(Acc.)	(F1)	(Acc.)	(Acc.)	(EM)	(F1a)	(Acc.)	(Acc.)
PET Best	75.1	83.5	52.6	65.7	35.2	75.0	80.4	83.3
PT	77.8	92.3	56.3	76.5	36.1	75.0	84.6	87.0
PTP+RM	79.9	93.2	58.1	**78.6**	36.2	78.3	85.9	88.6
PTP+RG	79.5	**93.7**	58.0	77.7	36.6	78.1	85.4	88.3
PTP+A2T	78.6	92.6	56.6	77.4	36.5	76.0	84.7	87.7
PTP+PGD	**80.2**	93.5	**58.5**	78.5	**37.4**	**78.9**	**86.0**	**88.9**
PET(Full Dev)	79.4	59.4	52.4	69.8	37.9	77.3	80.1	95.0
iPET(Full Dev)	80.6	92.4	52.2	74.0	33.0	74.0	-	-

3.0 Experiment

3.0.1 Experimental Setup

SuperGLUE benchmark [40] contains 8 challenging natural language understanding (NLU) tasks. We also include the few-shot version of SuperGLUE, FewGLUE benchmark [36] to test the ability of our algorithm, which consists of 32 training samples in each dataset on SuperGLUE. Following [36, 27], we report results on 7 of 8 NLU tasks in few-shot settings.

In fully-supervised setting, the full training set of each task in SuperGLUE [40] is employed during the prompt tuning process. Besides, in the model selection part, we adopt the whole validation set. As few-shot learning ability of prompt tuning can reduce the cost of annotations in real-world applications, following [36, 27], we also test our algorithm under few-shot settings. To be specific, we use the training set provided by FewGLUE [36], the few-shot version of SuperGLUE, containing 32 training pairs in each task. Besides, we use the same version of the development set as [27] to select models, which are created by randomly choosing 32 unused training pairs.

We include 2 prompt tuning methods P-tuning [27] (PT) and P-tuning-v2 [26] (PT2) as baselines. PT is the state-of-the-art method in FewGLUE benchmark while PT2 also achieves excellent performance in SuperGLUE benchmark. We defer the hyperparameters such as learning rate and prompt length in Appendix A.1. We also leave some figures and tables in Appendix.

Following the settings in [26, 27], we include BERT-large [9] and RoBERTa-large [28] for fully-supervised settings and ALBERTA-xxlarge-v2 [20] for few-shot settings. To have a fair comparison with the baseline methods, for fully-supervised settings, all backbone LMs are frozen, except in fine-tuning, same as [26]. For few-shot learning settings, backbone LMs are tuned with trainable prompt embeddings, same as [27].

3.0.2 Results on Fully-supervised Setting

In fully-supervised settings, Table 2 demonstrates the results of our proposed PTP algorithm with 4 different perturbations on all 8 tasks of SuperGLUE benchmark. It is worth noticing that PTP+PGD achieves the best performance in almost all datasets except WiC (BERT), COPA(BERT), and MultiRC (RoBERTa). Overall, the best method PTP+PGD outperforms the baseline method PT2 by 1.94% (with BERT-large backbone) and 1.63% (with RoBERTa-large backbone) on average.

PTP with PGD and RG perturbation on continuous space (embedding space) are perform better than PTP with RM and A2T, which indicates perturbing on continuous space is more effective than perturbing on discrete space in fully-supervised settings. As for pretrained LMs (BERT-large and RoBERTa-large), results show the superb learning ability of our PTP algorithm regardless of which backbone.

3.0.3 Results on Few-shot Setting

In few-shot learning settings, we employ FewGLUE, also known as few-shot version of SuperGLUE. PET [36] and iPET [35] are the methods using discrete prompts. We test the previous SOTA method on FewGLUE, PT [27] , as our baseline method and validate on the same development set (Dev 32). As illustrated in [27], for a fair comparison, the results of PET Besst (Dev 32) are reported as removing all the additional tricks like ensemble, distillation, etc. PET (Full Dev) and iPET (Full Dev) denote the methods with the original validation set.

Our main results are shown in Table 3. PTP achieves better results than the previous state-of-the-art method PT in all 7 tasks, which verifies the effectiveness of our algorithms in few-shot NLU tasks. Especially, PTP+PGD outperforms the previous PT by 2.34% on average. Comparing the PTP+PGD (Dev 32) to the methods with the original dev set, it still has an advantage on most of the tasks (5 of 7) while the results are similar in BoolQ (better than PET with full dev set but worse than iPET). The PTP with RM and RG perturbation method also achieve remarkable improvement when compared to the baseline method PT. Moreover, PTP with A2T perturbation can also boost the performance of the baseline by a

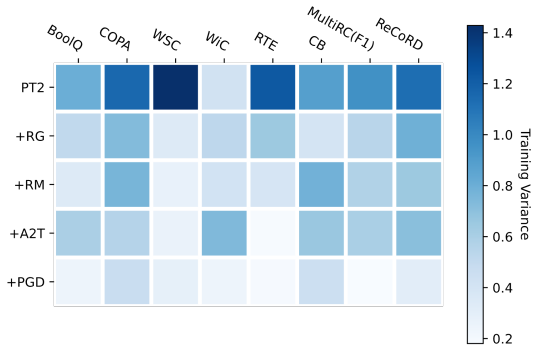


Figure 3: The variance of the scores on the dev sets from SuperGLUE Benchmark. We compute it on 5 runs with different random seeds to report. The reported experiments are all using BERT-large models as backbones.

small margin.

3.0.4 Results on Improving Training Stability

In addition to improved generalization performance, the proposed method could stable the training process. Figure 2 provides strong evidence that our proposed PTP-RN and PTP-ADV training methods have a much smoother loss landscape compared to the vanilla prompt tuning. For the few-shot learning setting, we demonstrate the results in Table 6. It could be seen that all our PTP methods have smaller variances than the baseline method. Specifically, PTP+PGD has the smallest variance in 5 runs, which indicates its training and generalization stability. Compared with the PTP-RN methods (RG, RM), PTP-ADV methods (A2T, PGD) achieve smaller variance. We also conduct the experiments under fully-supervised learning settings in Figure 3. It shows that in all 8 tasks from SuperGLUE benchmark, our proposed method can still reduce the training variance of different runs with the same hyperparameter except the seeds.

4.0 Ablation Study

In this section, we conduct comprehensive ablation studies of our perturbation methods under both fully-supervised and few-shot learning settings.

4.1 RG Perturbation.

We investigate the strength of gaussian noise δ and number of perturbed embeddings i affects the performance (see Eq. (2.6)). Specifically, we run experiments with σ , which is the variance of the added Gaussian noise, from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and the number of word embeddings perturbed, which is denoted as i in Eq. (2.6), from $\{1, 5, 10, 20\}$. Under the fully-supervised learning setting, we report the results on COPA, RTE, WiC tasks in Figure 4. The results show the appropriate σ is supposed to be 10^{-2} and the number of perturbed embeddings to be 5. With large σ and large i, PTP+RG is more likely to fail in comparison to the baseline method. Under few-shot learning settings, we select results in MultiRC task to report, as shown in Table 7. The encouraging result also demonstrates that the best choice of σ and number of embeddings perturbed is 10^{-3} and 5, respectively.

4.2 PGD Perturbation.

We investigate how different α (See Eq. (2.8)) and PGD iterations affect the performance. We run experiments with α from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and PGD iterations from 1 to 5. Under fully-supervised learning settings, we present the the results of COPA dataset in Table 8. It shows that large α in PGD will be detrimental to the performance. Under few-shot learning settings, Figure 5 demonstrates the results of different α and different iterations of PTP+PGD on few-shot settings. In all 3 datasets, when α is 10^{-3}, not too small nor too large, and PGD iters is 4, PTP+PGD can achieve outstanding performance.

4.3 RM Perturbation.

We investigate how different numbers of random [MASK] inserted affects the PTP. Formally, the number of [MASK] inserted is defined as i in Eq. (2.7). We conduct experiments with i from 1 to 10. Under few-shot learning settings, Figure 6 presents the results of PTP with RM perturbation on FewGLUE (MultiRC and RTE dataset). We observe that RM perturbation can boost the performance substantially in few-shot settings and the best choice of i is 8. Under fully-supervised settings, Table 9 presents the ablation of RM perturbation. It also shows that a large number of [MASK] inserted harms the performance, especially when the backbone is RoBERTa.

4.4 A2T Perturbation.

We investigate how the minimum cosine similarity between normal input s and perturbed input s' of A2T perturbation affects the results. We run experiments with minimum cosine similarity from $\{0.2, 0.4, 0.6, 0.8\}$ and show results in Table 10. It indicates small similarity may cause damage to the standard performance because the perturbation can be too large in this case.

5.0 Conclusions

In this paper, we first investigated the training instability issues on prompt tuning, which has a precipitous loss landscape in its visualization. To tackle the problem, we proposed PTP-RN and PTP-ADV algorithms, which include four different perturbations (RG, RM, ADV, A2T) on both discrete and continuous spaces, to smooth the loss landscape and make the training stable. Furthermore, our algorithms are also capable of boosting the performance of prompt tuning. The extensive experiments validate the effectiveness of our proposed algorithms on NLU tasks under both fully-supervised and few-shot settings.

APPENDIX

A.1 Details of learning rate and prompt length

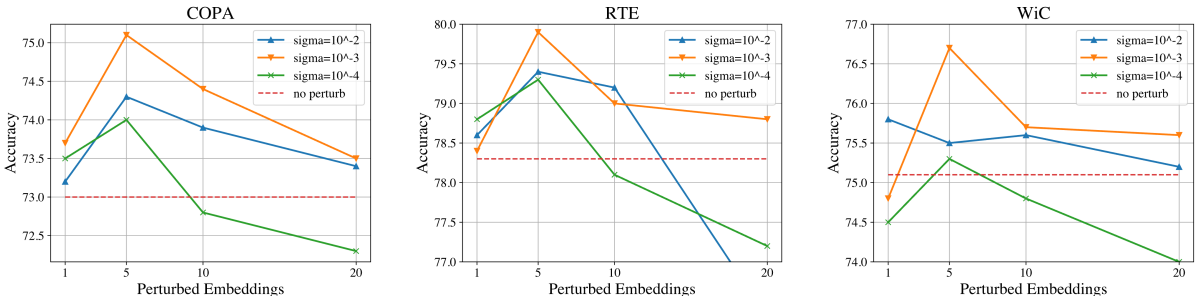


Figure 4: Performance of PTP+RG on SuperGLUE (WiC, RTE, COPA datasets) with σ from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and perturbed embeddings from {1, 5, 10, 20}. The dashed red line represents the performance of baseline method PT2 with BERT-large as backbone LM.

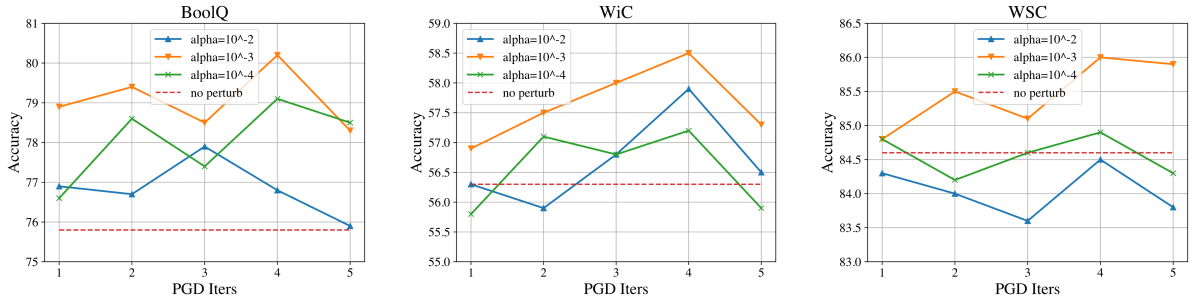


Figure 5: Performance of PTP+PGD on FewGLUE (WiC, BoolQ, WSC datasets) with different α and PGD iterations. The dashed red line represents the performance of the baseline method PT [27]. It shows the best α and PGD iterations are 10^{-3} and 4, respectively.

Under fully-supervised settings, the prompt length and learning rate details are presented as Table 4. Under few-shot learning settings, we report it as Table 5. The prompt length is exactly the same as the PT [27], thus we ignore it here.

Table 4: Prompt Length and Learning Rate details for 8 tasks on SuperGLUE. LR1, PL1: learning rate and prompt length for continuous prompts with BERT-large backbone. LR2 and PL2: learning rate and prompt length for prompts with RoBERTa-large backbone.

Tasks	LR1	LR2	PL1	PL2
BoolQ	1e-3	5e-3	40	16
COPA	5e-3	7e-3	16	16
RTE	1e-2	5e-3	20	128
WSC	3e-3	7e-3	16	8
CB	7e-3	9e-3	16	16
MultiRC	1e-4	3e-3	40	20
ReCoRD	3e-4	5e-3	16	40
WiC	1e-4	5e-3	20	16

A.2 Supplement for Experiment

This section includes the tables and figures as a supplement to our experiment and ablations. Table 6 demonstrates the comparison of the variance of different training methods on FewGLUE benchmark. Table 7 presents the ablation of our algorithm with RG perturbation on MultiRC [18] task. We show the ablation of PGD perturbation in Table 8. The ablation of RM perturbation is presented as Table 9. Table 10 shows the ablation of our proposed PTP+A2T training algorithm.

Figure 4 shows the ablation of RG perturbations on WiC [31], RTE [7], and COPA [13] datasets under fully-supervised learning setting while Figure 5 presents the ablation of our PTP+PGD training method on WiC [31], BoolQ [5] and WSC [22] datasets under few-shot learning settings. We show the ablation of our proposed PTP+RM algorithm as Figure 6 on FewGLUE benchmark.

All experiments are conducted on servers with RTX A6000 GPUs, each having 48GB of memory.

Table 5: Prompts' Learning Rate details for 7 tasks in FewGLUE.

Tasks	Learning Rate
BoolQ	5e-5
RTE	5e-5
WiC	1e-5
WSC	5e-5
COPA	1e-5
MultiRC	1e-4
CB	1e-5

Table 6: The variance of the scores on the dev sets of RTE, COPA and BoolQ from the FewGLUE benchmark. We compute it on 5 runs with different random seeds (other hyper-parameter are the same). We employ bold font to denote the smallest deviation in each task and blue font to denote the decrease when compared to PT.

Tasks	RTE	WSC	WiC	BoolQ
PT	1.89	1.68	1.77	1.45
+RG	0.81	0.78	0.91	0.56
+A2T	0.45	0.43	**0.39**	0.59
+RM	0.68	0.95	0.87	0.65
+PGD	0.35	0.31	0.47	**0.43**

Table 7: Results of different σ in RG perturbation (MultiRC task). The baseline method is PT2 and its performance is 75.0. E denotes number of embeddings that are perturbed. We mark the best and the worst.

MultiRC	E=1	E=5	E=10	E=20
σ=1e-2	75.2	74.3	75.8	74.7
σ=1e-3	77.3	**78.1**	77.1	76.6
σ=1e-4	76.9	77.2	76.8	75.5

Table 8: Results of PTP+PGD on fully-supervised COPA dataset. The baseline method is PT2 [26], whose accuracy is 73.0 . The backbone LM employed is BERT-large. α is the perturbation size while t is PGD iterations (see Eq. (2.8)).

COPA	t=1	2	3	4	5
α=1e-2	70.3	72.1	72.0	71.0	69.0
1e-3	73.4	70.8	72.9	73.5	73.2
1e-4	73.1	73.4	**74.7**	73.8	72.5

Table 9: The results of different i in Eq. (2.7) (number of [MASK] randomly inserted into input sequence as perturbation) . We select BoolQ with fully-supervised settings to report. The reported increase or decrease is compared to the baseline method PT2.

Dataset	LM	1	2	3	4	5	6	7	8	9	10
BoolQ	BERT	-0.12	+0.50	**+1.57**	+0.65	+1.35	+1.14	+1.12	+0.24	+0.25	-0.38
(Full)	RoBERTa	+0.31	+0.76	**+1.10**	+0.88	+0.69	-0.36	+0.19	-0.43	-0.67	-0.36

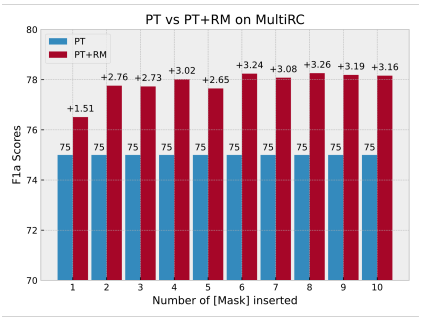


Figure 6: The results of different i in Eq. (2.7) (number of [MASK] randomly inserted into input sequence as perturbation). We select MultiRC and RTE tasks with Few-shot setting to report.

Cosine Sim	0.2	0.4	0.6	0.8
BoolQ(Full)	-0.83	-0.46	+0.61	+0.52
BoolQ(Few)	-0.96	-0.13	+0.78	+0.84
MultiRC(Full)	-0.77	-0.35	+0.25	+0.71
MultiRC(Few)	-0.59	-0.54	+1.03	+0.68

Table 10: The results of different minimum cosine similarity in A2T perturbation. Full: fully-supervised learning setting. Few: few-shot learning setting.

Bibliography

[1] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 2018.

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

[3] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1554–1565. PMLR, 2020.

[4] Minhao Cheng, Pin-Yu Chen, Sijia Liu, Shiyu Chang, Cho-Jui Hsieh, and Payel Das. Self-progressing robust training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7107–7115. AAAI Press, 2021.

[5] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[6] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019.

[7] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.

[8] Joe Davison, Joshua Feldman, and Alexander M. Rush. Commonsense knowledge mining from pretrained models. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1173–1178. Association for Computational Linguistics, 2019.

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

[10] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[11] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6616–6628. Curran Associates, Inc., 2020.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[13] Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. Semeval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, 2012.

[14] Adi Haviv, Jonathan Berant, and Amir Globerson. Bertese: Learning to speak to BERT. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*, pages 3618–3623. Association for Computational Linguistics, 2021.

[15] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[16] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438, 2020.

[17] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932, 2019.

[18] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. 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 252–262, 2018.

[19] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Robust optimization as data augmentation for large-scale graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 60–69. IEEE, 2022.

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[21] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[22] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[23] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online, November 2020. Association for Computational Linguistics.

[24] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics, 2021.

[25] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021.

[26] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[27] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[30] John Morris, Jin Yong Yoo, and Yanjun Qi. TextAttack: Lessons learned in designing python frameworks for NLP. In *Proceedings of Second Workshop for NLP Open*

Source Software (NLP-OSS), pages 126–131. Association for Computational Linguistics, November 2020.

[31] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*, 2018.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research(JMLR)*, 2020.

[33] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics.

[34] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[35] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*, pages 255–269. Association for Computational Linguistics, 2021.

[36] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2339–2352. Association for Computational Linguistics, 2021.

[37] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019.

[38] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In

6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[40] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS 2019*, 2019.

[41] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 816–825. Computer Vision Foundation / IEEE, 2020.

[42] Jin Yong Yoo, John Morris, Eli Lifland, and Yanjun Qi. Searching for a search method: Benchmarking search algorithms for generating NLP adversarial examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 323–332, Online, November 2020. Association for Computational Linguistics.

[43] Jin Yong Yoo and Yanjun Qi. Towards improving adversarial training of NLP models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 945–956. Association for Computational Linguistics, 2021.

[44] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.