When Gradient Descent Meets Derivative-Free Optimization: A Match Made in Black-Box Scenario

Chengcheng Han[◇*] **Liqing Cui**[◇*] **Renyu Zhu**[◇♠]
Jianing Wang[◇] **Nuo Chen**[◇] **Qiushi Sun**[◇♡] **Xiang Li**[◇] **Ming Gao**[◇♣†]
[◇]School of Data Science and Engineering,[‡] East China Normal University
[♠]NetEase Fuxi AI Lab
[♡]Department of Mathematics, National University of Singapore
[♣]KLATASDS-MOE, School of Statistics, East China Normal University
`{chengchenghan,liqingcui,jianingwang,nuochen,qiushisun}@stu.ecnu.edu.cn`
`zhurenyu@corp.netease.com`
`{xiangli,mgao}@dase.ecnu.edu.cn`

Abstract

Large pre-trained language models (PLMs) have garnered significant attention for their versatility and potential for solving a wide spectrum of natural language processing (NLP) tasks. However, the cost of running these PLMs may be prohibitive. Furthermore, PLMs may not be open-sourced due to commercial considerations and potential risks of misuse, such as GPT-3. The parameters and gradients of PLMs are unavailable in this scenario. To solve the issue, black-box tuning has been proposed, which utilizes derivative-free optimization (DFO), instead of gradient descent, for training task-specific continuous prompts. However, these gradient-free methods still exhibit a significant gap compared to gradient-based methods. In this paper, we introduce gradient descent into black-box tuning scenario through knowledge distillation. Furthermore, we propose a novel method GDFO, which integrates gradient descent and derivative-free optimization to optimize task-specific continuous prompts in a harmonized manner. Experimental results show that GDFO can achieve significant performance gains over previous state-of-the-art methods.

1 Introduction

Large pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Raffel et al., 2020) have attracted considerable attention for their versatility and potential for solving a wide spectrum of Natural Language Processing (NLP) tasks. Especially, through prompt-based learning (PL) (Liu et al., 2021a; Gu et al., 2022), PLMs have consistently demonstrated impressive performance on various downstream tasks with a few labeled samples. However, it is a challenge to extend the benefits of these large PLMs to a broader audience. For users, the cost of running these models may be prohibitive; for service providers, they may not open source the model parameters due to commercial considerations and potential risks of misuse[1]. One possible solution is to deploy PLMs as a service, enabling users to access the advanced capabilities of PLMs through their inference APIs, such as GPT-3 (Brown et al., 2020), ERNIE (Sun et al., 2021) and Yuan (Wu et al., 2021b).

In this scenario, the large pre-trained language model provided by the server is considered as a black box. In order to perform various downstream tasks, users are required to construct task-specific prompts or select training samples (Brown et al., 2020) to input into the black box. We can manually construct discrete prompts, which are simple and effective but may not fully utilize training data,

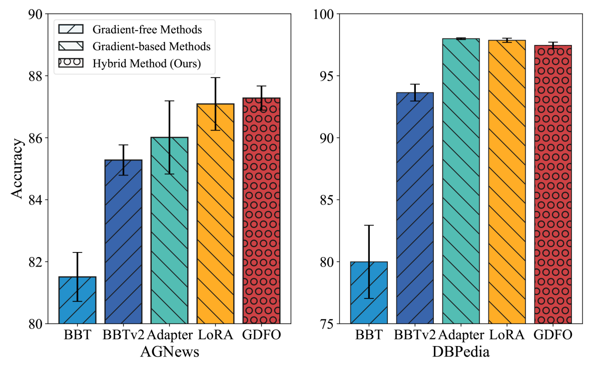

Figure 1: Accuracy (%) on the AG's News and DBPedia datasets. Experimental setup is detailed in Section 4.3. It is noted that prior gradient-based approaches, such as Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021), are unable to be used in black-box scenarios. GDFO is the first to introduce gradient descent in the black-box tuning scenario.

[*]Equal contribution.
[†]Corresponding author.
[‡] Shanghai Engineering Research Center of Big Data Management

[1]https://openai.com/blog/openai-api/

potentially resulting in suboptimal performance on some tasks. Instead of designing hand-crafted discrete prompts, there have been an increasing number of studies on continuous prompt tuning (Lester et al., 2021; Liu et al., 2021a; Ding et al., 2022), which aim to train continuous prompts and add them to the original samples. Trainable continuous prompts have also shown remarkable success on various tasks, but most existing methods optimize the continuous prompts through back-propagation, which is unavailable in the black-box scenario. To solve the issue, Sun et al. (2022b) have recently proposed Black Box Tuning (BBT), which utilizes random projection matrices and derivative-free optimization (DFO) (Kolda et al., 2003; Conn et al., 2009; Rios and Sahinidis, 2013), instead of gradient descent, for training continuous prompts in the black-box scenario. Built upon BBT, BBTv2 (Sun et al., 2022a) prepends continuous prompts to each layer of the PLM and further presents a divide-and-conquer gradient-free algorithm to alternately optimize the prompts at different layers. Both BBT and BBTv2 have shown their superiority against other gradient-free methods. Despite the success, there remains a significant gap compared to gradient-based methods on certain tasks. For example, compared against BBTv2, Adapter (Houlsby et al., 2019), a gradient-based method, leads by 4.35% on the DBPedia dataset (as shown in Figure 1). Therefore, we consider that the incorporation of gradient descent into the black-box scenario may potentially enhance the performance of the model.

Based on the insights discussed above, in this paper, we introduce gradient descent into the black-box scenario through knowledge distillation techniques. In particular, we propose a novel approach named **GDFO** to combine **G**radient descent with **D**erivative-**F**ree **O**ptimization, allowing them to jointly optimize task-specific continuous prompts. First, we adopt the technique of knowledge distillation, where a *student model* is trained to emulate the knowledge of the black-box model, referred to as the *teacher model*. Then, a prompt generator is trained by gradient descent through the *student model*, while utilizing derivative-free optimization algorithms to optimize continuous task-specific prompts. The continuous prompts generated by the prompt generator and the prompts optimized by the derivative-free algorithm are further integrated to serve as the final prompts. Finally, we perform extensive experiments on seven benchmark datasets to show that GDFO can achieve significant performance gains over other state-of-the-art methods. The main contributions of the paper are summarized as follows:

- To the best of our knowledge, we are the first to utilize gradient descent to optimize task-specific continuous prompts in the black-box scenario through knowledge distillation.
- We propose a novel method GDFO, which integrates gradient descent and derivative-free optimization to optimize task-specific continuous prompts in a harmonized manner.
- We conduct comprehensive experiments on seven benchmark datasets under the black-box scenario. Empirical results demonstrate the superiority of GDFO over other competitors.

2 Related Work

2.1 Prompt-based Learning

Prompt-based learning, in which the PLM is adapted to various tasks by task-specific prompts, has emerged as a promising framework. Brown et al. (2020) shows that PLM can perform excellently in few-shot learning by using manual prompts concatenated with samples. However, designing prompts in a hand-crafted fashion requires substantial time and experience and may not find the optimal ones (Jiang et al., 2020; Shin et al., 2021). To solve the problem, researchers attempt to use automated prompts. LM-BFF (Gao et al., 2021) uses prompt-based fine-tuning with automatically searched prompts and generates task demonstrations to be a part of the input context. P-tuning (Liu et al., 2021b) optimizes the continuous prompts using gradient descent as an alternative to discrete prompt searching. P-tuning v2 (Liu et al., 2021a) adopts continuous prompts for each layer of the PLMs to improve the model performance. Prefix-tuning (Li and Liang, 2021) optimizes continuous task-specific vectors and prepends them to the input texts. Input-tuning (An et al., 2022) fine-tunes both the continuous prompts and the input representations, leading to a more effective way to adapt unfamiliar inputs to frozen PLMs.

2.2 Black-box Tuning

Due to commercial considerations, large PLMs such as GPT-3 (Brown et al., 2020) are only pro-

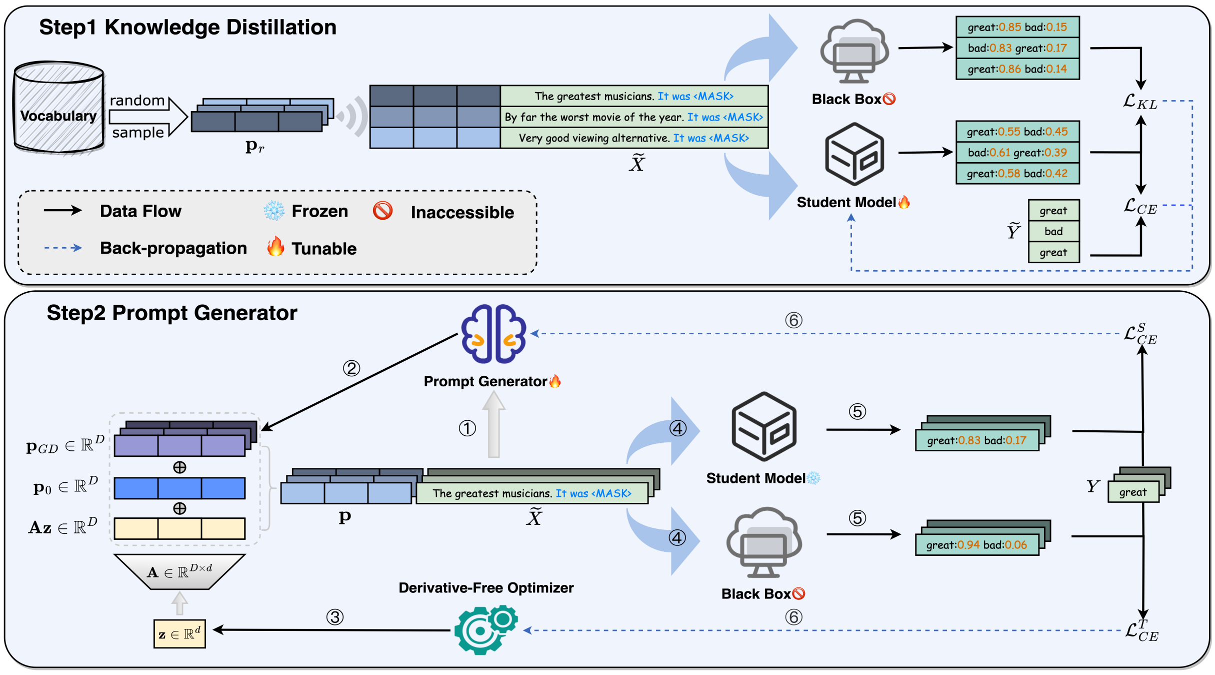

Figure 2: The overall architecture of GDFO. The details of the model are described in Section 3. The training procedure is shown in Algorithm 1.

vided as a service in the cloud, resulting inaccessible parameters and gradients of PLMs. To tackle this issue, BBT (Sun et al., 2022b; Diao et al., 2022) has been proposed to optimize the continuous prompts via derivative-free optimization (DFO). As an improved version of BBT, BBTv2 (Sun et al., 2022a) inserts prompts to each layer of the PLMs instead of optimizing the prompt merely in the input layer. Furthermore, GrIPS (Prasad et al., 2022) proposes a gradient-free search approach to generate discrete prompts. Besides, RLPrompt (Deng et al., 2022) optimizes discrete prompts through reinforcement learning and utilizes a continuous policy network which is highly parameter-efficient to generate prompts. PALP (Cho et al., 2022) combines linear models and in-context learning (Brown et al., 2020) to augment training samples with the templates for better contextualization. To improve the computational efficiency, PromptBoosting (Hou et al., 2022) constructs a pool of prompts via a gradient-free approach and ensembles many weak learners using the ADABOOST algorithm to enhance the model performance. Despite the success of the above approaches, all of them do not optimize continuous prompts through gradient descent (GD) in the black-box scenario, our method introduces GD to the scenario through knowledge distillation and combines GD and DFO to jointly optimize continuous prompts, which provides a novel insight for future black-box tuning approaches.

2.3 Knowledge Distillation

As a representative method of model compression, knowledge distillation transfers the knowledge from a larger deep neural network (*teacher*) to a smaller network (*student*) (Hinton et al., 2015; Kim and Rush, 2016). There have been different distillation algorithms being proposed to face more complex settings of transferring knowledge, including adversarial distillation (Ma et al., 2020; Wang et al., 2022), multi-teacher distillation (Guan et al., 2020; Yuan et al., 2021) and data-free distillation (Fang et al., 2022; Binici et al., 2022). Furthermore, the superior success of PLMs has also spurred researchers to distill PLMs into smaller models while retaining performance. DistilBERT (Sanh et al., 2019) introduces a triple loss combining language modeling and cosine-distance losses to leverage the inductive biases learned by large models during pre-training. TinyBERT (Jiao et al., 2020) performs a Transformer distillation method at both the pre-training and task-specific learning stages. NewsBERT (Wu et al., 2021a) designs a collaborative learning framework where the *student model* can learn from the experience of the *teacher model*. In this paper, we consider knowledge distillation to transfer knowledge from a black-box *teacher* to a *student*, which is used for training a prompt generator by gradient descent.

3 Method

In this section, we describe our approach GDFO. We first give an overview of GDFO, which is illustrated in Figure 2. GDFO first trains a *student model* by aligning its outputs to that of the *teacher model* (i.e., the black-box model). Then, GDFO trains the prompt generator by gradient descent while simultaneously optimizing the continuous prompts via DFO. Finally, the final prompts are obtained by integrating the prompts generated by the prompt generator with those optimized by DFO, which are then fed into the black-box model together with query instances to obtain predictions. Next, we describe each component of GDFO in detail.

3.1 Knowledge Distillation

Given a student model S and a teacher model T, the objective of knowledge distillation (KD) is to enhance the performance of S by aligning its outputs with those of T, which is accomplished by reducing the divergence between the probability distributions generated by S and T. In the black-box scenario, the black-box model is considered as T. We utilize T's outputs as soft targets for S to learn. Given a training instance, we randomly select n tokens from the PLM vocabulary to construct a random prompt $\mathbf{p}_r$, which is concatenated to the beginning of the instance. Additionally, a hand-crafted template[2] is appended to the end of the instance. We use the concatenated sentence as the input x. We denote $S(x)$ and $T(x)$ as the output logits of S and T for input x, respectively. The KD can be conducted by minimizing the Kullback-Leibler (KL) divergence distance between the *student* and *teacher* predictions:

$$\mathcal{L}_{KL} = \mathrm{KL}(\sigma(S(x)/\tau) \| \sigma(T(x)/\tau)), \tag{1}$$

where $\sigma(\cdot)$ denotes the `softmax` function and τ is a temperature hyper-parameter. The *student* parameters are updated according to $\mathcal{L}_{KL}$ and the cross-entropy loss $\mathcal{L}_{CE}$ over the ground-truth y:

$$\mathcal{L} = (1-\lambda)\,\mathcal{L}_{CE} + \lambda \mathcal{L}_{KL}, \tag{2}$$

where λ is a weight and $\mathcal{L}_{CE}$ is defined as:

$$\mathcal{L}_{CE} = -y \log \sigma(S(x)). \tag{3}$$

[2]The details of templates are shown in Table 1.

Algorithm 1 Training Procedure

```
Input: Training data $\{\mathcal{X}_{train}, \mathcal{Y}_{train}\}$; Black-box model $T$; Student model $S_\theta$; Prompt generator $G_\mu$; Epochs for knowledge distillation $E_{kd}$; The number of API calls $N$; The PLM vocabulary $\mathcal{V}$; Hand-crafted template $t$;
    # Knowledge Distillation
 1: for each $i \in E_{kd}$ do
 2:     for each $x \in \mathcal{X}_{train}$ do
 3:         $\mathbf{p}_r \leftarrow$ Random_Sample($\mathcal{V}, n$);
 4:         $\hat{y}_T, \hat{y}_S \leftarrow T([\mathbf{p}_r; x; t]), S_\theta([\mathbf{p}_r; x; t])$;
 5:         Calculate $\mathcal{L}$ by Equation 2;
 6:         Update $\theta$ by $\mathcal{L}$;
 7:     end for
 8: end for
    # Prompt Generator
 9: for each $i \in N$ do
10:     for each $x \in \mathcal{X}_{train}$ do
11:         $\mathbf{p}_{GD} \leftarrow G_\mu(x)$;
12:         Get $\mathbf{z}$ by CMA-ES;
13:         Get $\mathbf{p}$ by Equation 4;
14:         $\hat{y}_T, \hat{y}_S \leftarrow T([\mathbf{p}; x; t]), S([\mathbf{p}; x; t])$;
15:         Calculate $\mathcal{L}_{CE}^T, \mathcal{L}_{CE}^S$ by Equation 3;
16:         Update $\mu$ by $\mathcal{L}_{CE}^S$;
17:         Optimize CMA-ES by $\mathcal{L}_{CE}^T$;
18:     end for
19: end for
```

3.2 Prompt Generator

Upon the completion of training S via knowledge distillation, the *student* parameters are frozen and a prompt generator is optimized by gradient descent with the purpose of generating continuous prompts $\mathbf{p}_{\text{GD}} \in \mathbb{R}^D$ for given samples. Meanwhile, following BBT (Sun et al., 2022b), we optimize intermediate vector $\mathbf{z} \in \mathbb{R}^d$ through CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen and Ostermeier, 2001; Hansen et al., 2003), which is a widely used evolutionary algorithm for non-convex black-box optimization in continuous domain. Then a random projection matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$ is utilized to project $\mathbf{z}$ into the high-dimensional space. Finally, we randomly sample n tokens from the PLM vocabulary as initial prompt $\mathbf{p}_0$ and get final continuous prompt $\mathbf{p} \in \mathbb{R}^D$:

$$\mathbf{p} = \alpha \mathbf{p}_{\text{GD}} + (1-\alpha)(\mathbf{p}_0 + \mathbf{A}\mathbf{z}), \tag{4}$$

where α is a balancing weight. Further information regarding the initialization of $\mathbf{A}$ and the specifics of the optimization procedure of CMA-ES can be found in Sun et al. (2022b). Given a training instance, $\mathbf{p}$ is concatenated to the beginning of it and a hand-crafted template[2] is appended to the end of it. The concatenated sample is fed into S and T. Then the output logits are obtained and used to compute $\mathcal{L}_{CE}$, which is utilized to update the parameters of the prompt generator and optimize $\mathbf{z}$ through CMA-ES. The overall training procedure of GDFO is summarized in Algorithm 1.

3.3 Inference

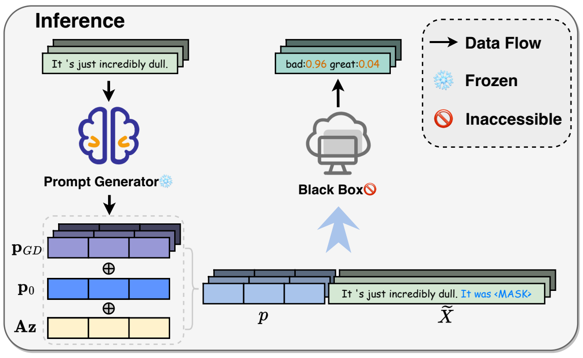

Figure 3: The inference procedure of GDFO.

During the inference stage, given a query instance, we first input it into the prompt generator to generate $\mathbf{p}_{\mathrm{GD}}$. Subsequently, we combine $\mathbf{p}_{\mathrm{GD}}$, $\mathbf{p}_0$, and $\mathbf{Az}$ that have been optimized through CMA-ES to obtain the final continuous prompt $\mathbf{p}$ through the Equation 4. Next, similar to the training stage, we concatenate $\mathbf{p}$ to the front of the query instance and append the hand-crafted template[2] to the end of it. Finally, we input the concatenated sample to the black-box model to obtain the prediction. The overall inference procedure is shown in Figure 3.

4 Experiments

In this section, we perform comprehensive experiments to compare our proposed model with twelve competitive baselines on seven downstream tasks.

4.1 Datasets

We perform experiments on a variety of language understanding tasks, including sentiment analysis, topic classification, natural language inference (NLI), and paraphrasing. Statistics of these datasets are given in Table 1. Specifically, we utilize the following datasets:

Sentiment analysis: SST-2 (Socher et al., 2013) and Yelp polarity (Zhang et al., 2015) consist of text samples with assigned sentiment labels (e.g. positive or negative).

Topic classification: AG's News (Zhang et al., 2015) and DBPedia (Zhang et al., 2015) contain text samples with pre-defined topics.

NLI: SNLI (Bowman et al., 2015) and RTE (Wang et al., 2018) are composed of sentence pairs and the objective is to determine the relationship between the two sentences, such as entailment, contradiction and neutral.

Paraphrase: MRPC (Dolan and Brockett, 2005) contains sentence pairs and the goal is to recognize semantic equivalence between the two sentences.

4.2 Baselines

We compare GDFO with twelve competitive methods, which can be grouped into two categories: gradient-based methods and gradient-free methods.

For gradient-based methods, we consider six baselines: **(1) Model Tuning** fine-tunes the entire PLM through training data. **(2) Adapter** (Houlsby et al., 2019) is a new module added between layers of a PLM. The parameters of the original network remain fixed, yielding a high degree of parameter sharing. **(3) BitFit** (Zaken et al., 2022) is a sparse-finetuning method where most of the network parameters are frozen and only the bias-terms of the model (or a subset of them) are being modified. **(4) LoRA** (Hu et al., 2021), an efficient adaptation strategy, allows us to train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change, while keeping the pre-trained weights frozen. **(5) Prompt Tuning** (Lester et al., 2021) freezes the entire PLM and only allows additional tunable tokens to be prepended to the input text. **(6) P-Tuning v2** (Liu et al., 2021a) applys continuous prompts for every layer of the PLM instead of the mere input layer.

For gradient-free methods, we also consider six baselines: **(1) Manual Prompt** conducts subsequent experiments using hand-crafted prompts following the pre-defined templates in Table 1. **(2) In-Context Learning** (Brown et al., 2020) provides a few training examples for the model to improve its capability of few-shot learning. **(3) Feature-MLP** trains a two-layered MLP classifier provided with embeddings encoded by the PLM. **(4) Feature-BiLSTM** trains a bidirectional LSTM on the word representations and connects it to a

Category	Datasets	# Classes	# Training samples	# Test samples	Templates	Label words
Single sentence	SST-2	2	32	0.9k	<Sentence>. It was [MASK].	great, bad
	Yelp P.	2	32	38k	<Sentence>. It was [MASK].	great, bad
	AG's News	4	64	7.6k	[MASK] News: <Sentence>	World, Sports, Business, Tech
	DBPedia	14	224	70k	[Category: [MASK]] <Sentence>	Company, Education, Artist, Athlete, Office, Transportation, Building, Natural, Village, Animal, Plant, Album, Film, Written
Sentence pair	MRPC	2	32	0.4k	<Sentence$_1$> ? [MASK] , <Sentence$_2$>	Yes, No
	RTE	2	32	0.3k	<Sentence$_1$> ? [MASK] , <Sentence$_2$>	Yes, No
	SNLI	3	48	9.8k	<Sentence$_1$> ? [MASK] , <Sentence$_2$>	Yes, Maybe, No

Table 1: Statistics, hand-crafted templates and label words of datasets.

linear classifier. **(5) BBT** (Sun et al., 2022b) optimizes the continuous prompt prepended to the input text via derivative-free optimization (DFO). **(6) BBTv2** (Sun et al., 2022a) proposes a divide-and-conquer algorithm to alternately optimize the prompt at each layer of the PLM. Compared with BBT, BBTv2 inserts prompts to each layer of the PLM instead of optimizing the prompt merely in the input layer.

4.3 Implementation

Few-shot setting We adopt the same procedure as described in previous studies (Zhang et al., 2020; Sun et al., 2022a) to establish a true few-shot learning environment. Specifically, we randomly select k samples per class to create a k-shot training set $\mathcal{D}_{train}$, and form a development set $\mathcal{D}_{dev}$ by randomly selecting another k samples from the original training set, resulting in $|\mathcal{D}_{train}| = |\mathcal{D}_{dev}|$. We use the original development sets as our test sets $\mathcal{D}_{test}$. For datasets that do not have development sets, we use the original test sets. It is noted that $|\mathcal{D}_{test}| \gg |\mathcal{D}_{train}| = |\mathcal{D}_{dev}|$.

Experimental settings To compare with BBTv2 (Sun et al., 2022a), we mainly use RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019) as the black-box model. For hyper-parameters, we use the grid search to find the best for our model. For knowledge distillation, we use BERTLARGE (Devlin et al., 2019) as our *student model*. We set the temperature τ to 1 and the balancing weight λ to 0.5. We fine-tune the *student model* for 2,000 epochs with the learning rate $1e-4$. For prompt generator, we use a fully connected layer and set the dimensionality of the fully connected layer to 1024. The learning rate of the prompt generator is $1e-5$. For CMA-ES, following Sun et al. (2022b), we set the prompt length n to 50. The dimensionality of $\mathbf{z}$ is set to 500 and the population size of CMA-ES is set to 20. The balancing weight α is set to 0.5. We train our prompt generator and run CMA-ES for 8,000 API calls. All baseline results are recorded in Sun et al. (2022a). We run all the experiments on a single NVIDIA v100 GPU.

4.4 Main Results

The results of 16-shot setting on various downstream tasks are shown in Table 2. From the table, GDFO consistently outperforms all the baselines on the average performance. Specifically, in the gradient-based comparison, GDFO achieves an average accuracy of 81.85%, which outperforms the runner-up gradient-based model, LoRA, by a notable 3.84% improvement. When compared against the gradient-free methods, GDFO leads BBTv2 by 5.26% and 3.89% on the SNLI and RTE datasets, respectively. Our model generates a continuous prompt for each sample, rather than using an optimized continuous prompt for all samples, such as BBT and BBTv2. Furthermore, the incorporation of both DFO and gradient descent during the training stage allows GDFO for more comprehensive and efficient training of continuous prompts, resulting in a notable improvement in the model performance.

4.5 Ablation Study

We conduct an ablation study to investigate the characteristics of the main components of GDFO. As illustrated in Figure 4, the results[3] demonstrate that GDFO outperforms GDFO-w/o-KD. For instance, on the SNLI dataset, the accuracy of GDFO is 62.53%, whereas that of GDFO-w/o-KD is only 58.51%. This indicates that the knowledge distillation module, which transfers the knowledge of the *teacher model* to the *student model* by aligning

[3]The evaluation metric used in the ablation study is F1 score for MRPC and accuracy for other datasets.

	Methods	SST-2 acc	Yelp P. acc	AG's News acc	DBPedia acc	MRPC F1	SNLI acc	RTE acc	Average
Gradient-based	Model Tuning	85.39$_{\pm 2.84}$	91.82$_{\pm 0.79}$	86.36$_{\pm 1.85}$	97.98$_{\pm 0.14}$	77.35$_{\pm 5.70}$	54.64$_{\pm 5.29}$	58.60$_{\pm 6.21}$	78.88
	Adapter	83.91$_{\pm 2.90}$	90.99$_{\pm 2.86}$	86.01$_{\pm 2.18}$	**97.99$_{\pm 0.07}$**	69.20$_{\pm 3.58}$	57.46$_{\pm 6.63}$	48.62$_{\pm 4.74}$	76.31
	BitFit	81.19$_{\pm 6.08}$	88.63$_{\pm 6.69}$	86.83$_{\pm 0.62}$	94.42$_{\pm 0.94}$	66.26$_{\pm 6.81}$	53.42$_{\pm 10.63}$	52.59$_{\pm 5.31}$	74.76
	LoRA	88.49$_{\pm 2.90}$	90.21$_{\pm 4.00}$	87.09$_{\pm 0.85}$	97.86$_{\pm 0.17}$	72.14$_{\pm 2.23}$	61.03$_{\pm 8.55}$	49.22$_{\pm 5.12}$	78.01
	Prompt Tuning	68.23$_{\pm 3.78}$	61.02$_{\pm 6.65}$	84.81$_{\pm 0.66}$	87.75$_{\pm 1.48}$	51.61$_{\pm 8.67}$	36.13$_{\pm 1.51}$	54.69$_{\pm 3.79}$	63.46
	P-Tuning v2	64.33$_{\pm 3.05}$	92.63$_{\pm 1.39}$	83.46$_{\pm 1.01}$	97.05$_{\pm 0.41}$	68.14$_{\pm 3.89}$	36.89$_{\pm 0.79}$	50.78$_{\pm 2.28}$	70.47
Gradient-free	Manual Prompt	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
	In-Context Learning	79.79$_{\pm 3.06}$	85.38$_{\pm 3.92}$	62.21$_{\pm 13.46}$	34.83$_{\pm 7.59}$	45.81$_{\pm 6.67}$	47.11$_{\pm 0.63}$	60.36$_{\pm 1.56}$	59.36
	Feature-MLP	64.80$_{\pm 1.78}$	79.20$_{\pm 2.26}$	70.77$_{\pm 0.67}$	87.78$_{\pm 0.61}$	68.40$_{\pm 0.86}$	42.01$_{\pm 0.33}$	53.43$_{\pm 1.57}$	66.63
	Feature-BiLSTM	65.95$_{\pm 0.99}$	74.68$_{\pm 0.10}$	77.28$_{\pm 2.83}$	90.37$_{\pm 3.10}$	71.55$_{\pm 7.10}$	46.02$_{\pm 0.38}$	52.17$_{\pm 0.25}$	68.29
	BBT	89.56$_{\pm 0.25}$	91.50$_{\pm 0.16}$	81.51$_{\pm 0.79}$	79.99$_{\pm 2.95}$	61.56$_{\pm 4.34}$	46.58$_{\pm 1.33}$	52.59$_{\pm 2.21}$	71.90
	BBTv2	90.33$_{\pm 1.73}$	92.86$_{\pm 0.62}$	85.28$_{\pm 0.49}$	93.64$_{\pm 0.68}$	77.01$_{\pm 4.73}$	57.27$_{\pm 2.27}$	56.68$_{\pm 3.32}$	79.01
Hybrid	**GDFO (ours)**	**92.41$_{\pm 1.03}$**	**93.17$_{\pm 0.37}$**	**87.19$_{\pm 0.51}$**	96.92$_{\pm 0.71}$	**80.13$_{\pm 1.97}$**	**62.53$_{\pm 1.31}$**	**60.57$_{\pm 1.02}$**	**81.85**

Table 2: Results (%) of 16-shot setting on various downstream tasks. Following Sun et al. (2022a), we report mean and standard deviation of performance over 3 different splits.We highlight the best results in **bold**.

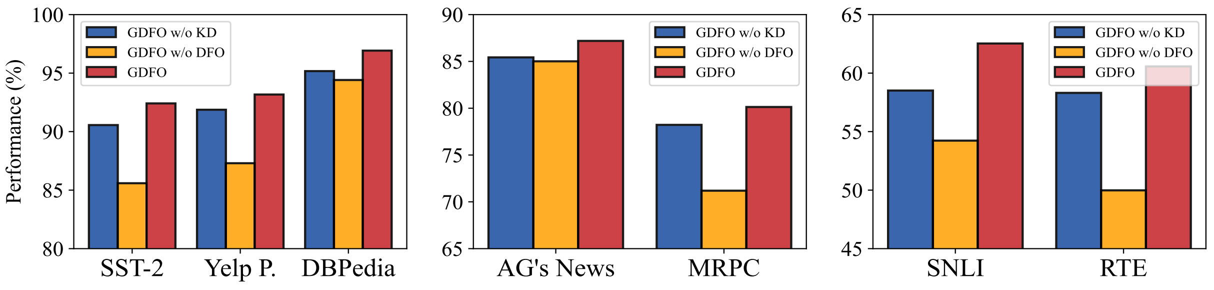

Figure 4: Ablation study: Results (%) of 16-shot problems over seven datasets. **w/o KD** denotes removing knowledge distillation and **w/o DFO** denotes removing derivative-free optimization. When removing the prompt generator, our method degrades to BBT (Sun et al., 2022b). The comparison of GDFO and BBT is shown in Table 2. The detailed analysis are described in Section 4.5.

the outputs of the *student model* with those of the *teacher model*, effectively improves the model performance. Additionally, when removing derivative-free optimization, a significant decline is observed across all datasets, with an average decrease of 6.5%. This demonstrates the effectiveness of incorporating derivative-free optimization in the black-box scenario. It is worth noting that when removing the prompt generator, the student model will not function, which means that gradient descent is eliminated. In this case, our method degrades to a gradient-free method BBT. The results, as shown in Table 2, reveal that GDFO achieves significant performance gains over BBT across all datasets, which demonstrates the effectiveness of training the prompt generator through gradient descent in the black-box scenarios.

4.6 Analysis

Different Black-Box Models To evaluate the universality of GDFO across PLMs with varying architectures, in addition to encoder-only PLMs (e.g., RoBERTa$_{\text{LARGE}}$), we conduct experiments using decoder-only (e.g., GPT-2$_{\text{LARGE}}$) and encoder-decoder PLMs (e.g., BART$_{\text{LARGE}}$ and T5$_{\text{LARGE}}$) as black-box models. As shown in Figure 5, GDFO achieves superior performance over other competitors across all the settings. For example, When using GPT-2 as the black-box model, GDFO achieves 87.5% and 85.2% on the SST-2 and DBPedia datasets, respectively. In particular, it outperforms BBT by a notable 11.9% and 15.5% improvements in both cases. When considering BART as the black-box model, GDFO leads BBTv2 by 8.12% on the DBPedia dataset. All the results clearly show the generalizability of our model across various PLMs.

Different Student Models We next conduct an in-depth experiment for student models on three datasets. The results are shown in Table 3. From the results, different student models have a impact on the performance of GDFO (approximately 2% on average). Furthermore, we observe that student

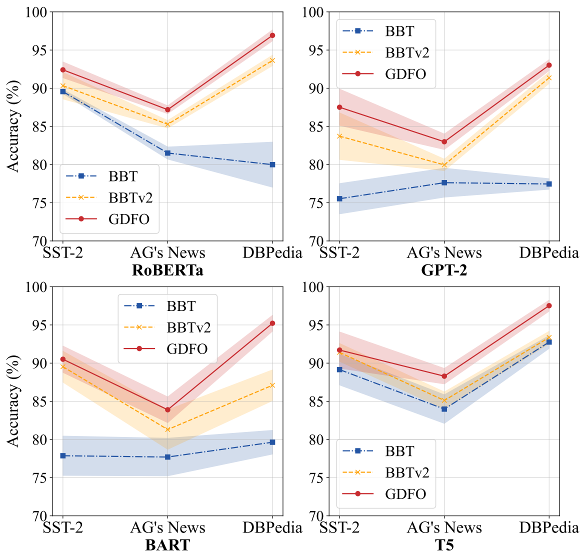

Figure 5: Accuracy (%) on different black-box models. We report mean and standard deviation of performance over 3 different splits. The results of BBT and BBTv2 are reported in (Sun et al., 2022a).

Stu. Models	SST-2	AG's News	DBPedia
	Encoder-only PLMs		
$\text{BERT}_{\text{LARGE}}$	$92.41_{\pm 1.03}$	$87.19_{\pm 0.51}$	$96.92_{\pm 0.71}$
$\text{RoBERTa}_{\text{LARGE}}$	$\mathbf{93.17}_{\pm \mathbf{0.39}}$	$\mathbf{88.91}_{\pm \mathbf{0.47}}$	$\mathbf{97.56}_{\pm \mathbf{0.53}}$
	Decoder-only PLMs		
$\text{GPT-2}_{\text{LARGE}}$	$91.12_{\pm 1.72}$	$85.98_{\pm 1.28}$	$95.91_{\pm 2.01}$
	Encoder-Decoder PLMs		
$\text{BART}_{\text{LARGE}}$	$91.19_{\pm 0.93}$	$87.07_{\pm 0.57}$	$96.13_{\pm 0.82}$
T5_{LARGE}	$93.03_{\pm 0.31}$	$88.87_{\pm 0.47}$	$97.73_{\pm 0.98}$

Table 3: Accuracy (%) of GDFO based on different student models. The black-box model is $\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019). We report mean and standard deviation of performance over 3 different splits. We highlight the best results in bold.

models whose architectures are similar to the black-box model tend to exhibit superior performance. For instance, when both the black-box model and the student model are $\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019), GDFO achieves the best performance. When comparing models with identical architectures, such as $\text{BART}_{\text{LARGE}}$ (Lewis et al., 2020) and T5_{LARGE} (Raffel et al., 2020), T5 exhibits superior performance, which may be due to the fact that the T5 model has twice the number of parameters as the BART model. The increased capacity allows the T5 model to better capture and represent the relationships within the input data, resulting in improved performance.

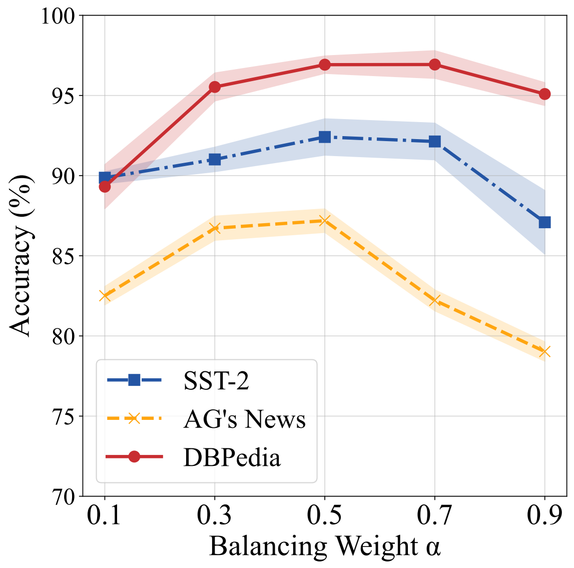

Figure 6: Effect of the balancing weight α on three datasets. We report mean and standard deviation of performance over 3 different splits.

Effect of Balancing Weight The balancing weight α plays a crucial role in determining the performance of the model by controlling the influence of $\mathbf{p}_{\text{GD}}$ and $\mathbf{Az}$. As the value of α increases, the influence of $\mathbf{p}_{\text{GD}}$ becomes more prominent, while conversely, as the value of α decreases, the influence of $\mathbf{Az}$ becomes more pronounced[4]. As illustrated in the Figure 6, when α is set to an extreme value, either too large or too small, it tends to have a negative impact on the model performance. We observe that the average performance of the model across three datasets is optimal when α is set to 0.5, further emphasizing the importance of the combination of derivative-free optimization and gradient descent in improving the performance of the model.

5 Conclusion

In this paper, we introduced gradient descent into the black-box tuning scenario through knowledge distillation for the first time, which provided a novel insight for future black-box tuning approaches. Furthermore, we proposed a novel method, GDFO, which integrates gradient descent and derivative-free optimization for jointly training continuous prompts. GDFO first trains a *student model* to enhance the performance by aligning its outputs with those of the *teacher model* (i.e., the black-box model). After that, GDFO trains a prompt generator using gradient descent while

[4] $\mathbf{p}_0$ is fixed, thus its effect on the model performance is disregarded in the analysis.

simultaneously optimizing a continuous prompt using DFO algorithm. Experimental results on various datasets show that GDFO can achieve significant performance gains over other gradient-free and gradient-based methods.

Limitations

We summarize the limitations of this work as follows: (1) We conduct experiments on 7 language understanding tasks across 4 types (i.e., sentiment analysis, topic classification, natural language inference and paraphrasing). However, the effectiveness of GDFO on tasks such as sequence labeling and generation tasks has yet to be fully examined. (2) Our proposed method uses a student model and a prompt generator, thereby resulting in a higher computational resource requirement in comparison to gradient-free methods. Therefore, it may not be suitable for implementation on certain edge devices, but it is more appropriate for personal or enterprise users who have access to a certain degree of computational resources and have stringent requirements for the model performance. (3) We only focus on the few-shot setting in this paper. It is possible to extend our work to other scenarios such as semi-supervised learning and we will further explore it in the future research.

Ethics Statement

The proposed method has no obvious potential risks. All the scientific artifacts used/created are properly cited/licensed, and the usage is consistent with their intended use.

Acknowledgements

This work has been supported by the National Natural Science Foundation of China under Grant No. U1911203, and the National Natural Science Foundation of China under Grant No.61977025.

References

Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. 2022. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.

Kuluhan Binici, Shivam Aggarwal, Nam Trung Pham, Karianto Leman, and Tulika Mitra. 2022. Robust and resource-efficient data-free knowledge distillation by generative pseudo replay. *arXiv preprint arXiv:2201.03019*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *NIPS*, 33:1877–1901.

Hyunsoo Cho, Hyuhng Joon Kim, Junyeob Kim, Sang-Woo Lee, Sang-goo Lee, Kang Min Yoo, and Taeuk Kim. 2022. Prompt-augmented linear probing: Scaling beyond the limit of few-shot in-context learners. *arXiv preprint arXiv:2212.10873*.

Andrew R Conn, Katya Scheinberg, and Luis N Vicente. 2009. *Introduction to derivative-free optimization*. SIAM.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Shizhe Diao, Xuechun Li, Yong Lin, Zhichao Huang, and Tong Zhang. 2022. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP*.

Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. 2022. Up to 100x faster data-free knowledge distillation. In *AAAI*, volume 36, pages 6597–6604.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423.

Yushuo Guan, Pengyu Zhao, Bingxuan Wang, Yuanxing Zhang, Cong Yao, Kaigui Bian, and Jian Tang. 2020. Differentiable feature aggregation search for knowledge distillation. In *ECCV*, pages 469–484. Springer.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Bairu Hou, Joe O'Connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2022. Promptboosting: Black-box text classification with ten forward passes. *arXiv preprint arXiv:2212.09257*.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*, 8:423–438.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *EMNLP*, pages 4163–4174.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*, pages 1317–1327.

Tamara G Kolda, Robert Michael Lewis, and Virginia Torczon. 2003. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

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

Xinyin Ma, Yongliang Shen, Gongfan Fang, Chen Chen, Chenghao Jia, and Weiming Lu. 2020. Adversarial self-supervised data-free distillation for text classification. In *EMNLP*, pages 6182–6192.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67.

Luis Miguel Rios and Nikolaos V Sahinidis. 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen Jr, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *EMNLP*, pages 7699–7715.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Tianxiang Sun, Zhengfu He, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022a. Bbtv2: Pure black-box optimization can be comparable to gradient descent for few-shot learning. *arXiv preprint arXiv:2205.11200*.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

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*.

Yixu Wang, Jie Li, Hong Liu, Yan Wang, Yongjian Wu, Feiyue Huang, and Rongrong Ji. 2022. Black-box dissector: Towards erasing-based hard-label model stealing attack. In *ECCV*, pages 192–208. Springer.

Chuhan Wu, Fangzhao Wu, Yang Yu, Tao Qi, Yongfeng Huang, and Qi Liu. 2021a. Newsbert: Distilling pre-trained language model for intelligent news application. In *EMNLP*, pages 3285–3295.

Shaohua Wu, Xudong Zhao, Tong Yu, Rongguo Zhang, Chong Shen, Hongli Liu, Feng Li, Hong Zhu, Jiangang Luo, Liang Xu, et al. 2021b. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning. *arXiv preprint arXiv:2110.04725*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *NIPS*, 32.

Fei Yuan, Linjun Shou, Jian Pei, Wutao Lin, Ming Gong, Yan Fu, and Daxin Jiang. 2021. Reinforced multi-teacher selection for knowledge distillation. In *AAAI*, volume 35, pages 14284–14291.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. In *ICLR*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *NIPS*, 28.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The Section of Limitation

☑ A2. Did you discuss any potential risks of your work?
The Section of Ethics Statement

☑ A3. Do the abstract and introduction summarize the paper's main claims?
The Section of Abstract and Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4 and The Section of Ethics Statement

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The Section of Ethics Statement

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.