# Exploring Universal Intrinsic Task Subspace for Few-Shot Learning via Prompt Tuning

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, *Member, IEEE*, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, and Jie Zhou

***Abstract*—Why can pre-trained language models (PLMs) learn universal representations and effectively adapt to broad NLP tasks differing a lot superficially? In this work, we empirically find evidence indicating that the adaptations of PLMs to various few-shot tasks can be reparameterized as optimizing only a few free parameters in a unified low-dimensional *intrinsic task subspace*, which may help us understand why PLMs could easily adapt to various NLP tasks with small-scale data. To find such a subspace and examine its universality, we propose an analysis pipeline called *intrinsic prompt tuning* (IPT). Specifically, we resort to the recent success of prompt tuning and decompose the soft prompts of multiple NLP tasks into the same low-dimensional nonlinear subspace, then we learn to adapt the PLM to unseen data or tasks by only tuning parameters in this subspace. In the experiments, we study diverse few-shot NLP tasks and surprisingly find that in a 250-dimensional subspace found with 100 tasks, by only tuning 250 free parameters, we can recover 97% and 83% of the full prompt tuning performance for 100 seen tasks (using different training data) and 20 unseen tasks, respectively, showing great generalization ability of the found intrinsic task subspace. Besides being an analysis tool, IPTcould further help us improve the prompt tuning stability.**

***Index Terms*—Intrinsic dimension, pre-trained language model, prompt tuning, task unification.**

## I. Introduction

PRE-TRAINED language models (PLMs), such as BERT [1], GPT [2], and T5 [3], have shown dominant performances on various natural language processing (NLP) tasks [4], [5]. After pre-training huge model parameters on massive data, a PLM can easily adapt to diverse downstream NLP tasks with small-scale data through full-parameter fine-tuning or parameter-efficient tuning methods [6], [7]. While NLP tasks widely benefit from the universality of PLMs, the mechanisms behind their success remain unclear. Why can PLMs learn universal representations through task-irrelevant pre-training objectives and easily adapt to diverse NLP tasks differing a lot? A deep understanding of this question may impact the designing of pre-training and downstream adaption algorithms profoundly. To provide a possible answer to this important question, in this paper, we hypothesize that the adaptations of PLMs to various downstream tasks can be reparameterized as optimizing only a few free parameters in a unified low-dimensional parameter subspace, which is dubbed as *intrinsic task subspace* (Fig. 1), and we find certain empirical evidence supporting the hypothesis.

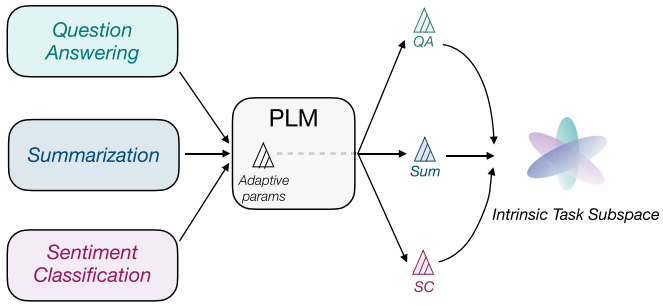


Fig. 1. An illustration of a common low-dimensional intrinsic task subspace for diverse tasks. PLMs tune adaptive parameters to adapt to each task.

During adaptation to a specific downstream task, PLMs optimize the tunable *adaptive parameters*. This is typically a high-dimensional optimization problem. For instance, in conventional fine-tuning, the adaptive parameters are all the PLM parameters, which may exceed hundreds of millions. Even in parameter-efficient tuning methods like prompt tuning [6], there are still tens of thousands of adaptive parameters. Interestingly, [8] show that the adaptation to a single task of a PLM can be reparameterized into only optimizing hundreds of free parameters in a low-dimensional subspace and then randomly projecting the tuned parameters back to the full model parameter space. This motivates our hypothesis that adaptations to multiple tasks can be reparameterized into optimizations in **a unified** low-dimensional intrinsic task subspace. If this hypothesis holds, then (1) the existence of a common task

Manuscript received 26 July 2023; revised 6 February 2024 and 4 May 2024; accepted 10 July 2024. Date of publication 18 July 2024; date of current version 29 July 2024. This work was supported by the National Key R&D Program of China under Grant 2020AAA0106502. The associate editor coordinating the review of this article and approving it for publication was Dr. Boyang Li. *(Yujia Qin and Xiaozhi Wang contributed equally to this work.) (Corresponding authors: Zhiyuan Liu; Juanzi Li.)*

Yujia Qin, Xiaozhi Wang, Yusheng Su, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, and Maosong Sun are with the Department of Computer Science, Tsinghua University, Beijing 100084, China (e-mail: qyj20@mails.tsinghua.edu.cn; wangxz20@mails.tsinghua.edu.cn; yushengsu.thu@gmail.com; ningding.cs@gmail.com; 18715465095@163.com; chen weize1998@gmail.com; liuzy@tsinghua.edu.cn; lijuanzi@tsinghua.edu.cn; houlei@tsinghua.edu.cn; sms@tsinghua.edu.cn).

Yankai Lin is with the Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China (e-mail: mrlyk423@gmail.com).

Peng Li is with the Institute for AI Industry Research (AIR), Tsinghua University, Beijing 100084, China (e-mail: lipeng@air.tsinghua.edu.cn).

Jie Zhou is with the Pattern Recognition Center, WeChat AI, Tencent Inc., Beijing 100084, China (e-mail: withtomzhou@tencent.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASLP.2024.3430545, provided by the authors.

Digital Object Identifier 10.1109/TASLP.2024.3430545

reparameterization subspace provides a possible explanation for the universality of PLMs and (2) the low dimensionality of the intrinsic task subspace explains why the adaptations can be done with relatively little data and computation in view of the tremendous model parameters. From this perspective, the PLMs serve as general *task compression frameworks*, which compress the learning complexity of various tasks from very high dimensionalities to low dimensionalities.

To find empirical evidence for the hypothesis, we need to develop methods for finding the common intrinsic task subspaces of PLMs. Naturally, the subspace should contain the solutions of PLM adaptations to various tasks (i.e., the tuned adaptive parameters for different tasks). Therefore, an intuitive way to approximate the intrinsic task subspace is to train a low-dimensional decomposition of the adaptive parameters of multiple tasks and then examine whether we can learn solutions to unseen tasks in the found subspace. However, training a decomposition for all the PLM parameters (i.e., the case of full-parameter fine-tuning) is computationally unaffordable since the required parameters of the decomposition would be hundreds of times of PLM parameters. Fortunately, prompt tuning (PT) provides an effective parameter-efficient alternative, whose number of adaptive parameters (*soft prompts*) are only tens of thousands. PT can also achieve close performance to fine-tuning on both understanding [6] and generation [9] tasks.

In experiments, we explore the common intrinsic subspace through PT under the few-shot learning setting, which ensures the data scales of various tasks are balanced. We name the analysis pipeline used in this paper as **I**ntrinsic **P**rompt **T**uning (**IPT**), which consists of two phases: multi-task subspace finding (MSF) and intrinsic subspace tuning (IST). During MSF, we first obtain trained soft prompts for multiple tasks as the initial adaptive parameters and then learn an auto-encoder for them. The auto-encoder defines the desired intrinsic task subspace with a projection function to project the soft prompts into the subspace and a back-projection function to project vectors in the subspace back to prompts. Then we train the two projection functions with the vector reconstruction loss and the task-specific loss to ensure the learned subspace contains adaptation solutions for each task. During IST, to adapt the PLM to unseen data and tasks, we only train the few free parameters in the low-dimensional subspace found by MSF and project the trained low-dimensional vectors into adaptive parameters (soft prompts) with the fixed back-projection function.

Surprisingly, we find that the intrinsic task subspace may not only exist but also is extremely low-dimensional. We study diverse few-shot NLP tasks and find that in a 250-dimensional subspace found by MSF using 100 training tasks, IST can recover 97% and 83% of the full PT performance for 100 seen tasks (using different training data) and 20 unseen tasks, respectively. Furthermore, we analyze the effect of training task types, the number of training tasks, and training data scales for IPT. We also show that IPTand the intrinsic task subspace could help us analyze task differences and improve PT training stability. We encourage future work to explore how to better find the intrinsic task subspace and develop techniques taking inspiration from universal reparameterizations of PLM adaptations.

## II. Related Work

### A. Pre-Trained Language Model, Fine-Tuning, Prompt Tuning

Since the success of BERT [1], pre-trained language models bring a new paradigm to NLP, that is to pre-train a massive model as the universal backbone and then adapt the PLMs to specific downstream tasks. The mainstream way of downstream adaptation is fine-tuning, which adds task-specific classification heads and tunes all the PLM parameters with supervised data.

Recently, researchers found that promising results can be achieved by casting downstream tasks into the form of pre-training tasks and adding some *prompt* tokens into the input, including human-designed explainable prompts [10], [11], [12] and automatically searched prompts [13], [14], [15]. Following this line of study, the prompts are extended from real tokens to trainable embeddings, i.e., soft prompts [16], [17], [18]. Furthermore, some works [6], [9] demonstrate that only tuning soft prompts and keeping PLMs frozen can achieve excellent performance in various tasks, especially for large-scale PLMs. Especially, [6] show that with the growth of PLM's size, the gap between prompt tuning and fine-tuning becomes narrower and finally disappears. Prompt tuning is further proven to be effective for both data-rich settings [19], [20], [21], [22] and few-shot learning settings [23], [24], [25], [26], [27]. In this work, we try to understand these phenomena, i.e., why PLMs can learn universal abilities to adapt to various tasks with a few tunable parameters and a few data points compared to the large volume of parameters, especially in the few-shot setting. Furthermore, as shown in Section V-E, IPT can also alleviate the stability issue of prompt tuning, which is significant in the few-shot learning setting.

### B. Intrinsic Dimensionality

*Intrinsic dimension* (ID) is the minimal number of variables needed to represent some data or approximate a function. [28] propose to measure the IDs of objective functions optimized by neural networks by randomly projecting all the trainable parameters into linear subspaces and finding the minimal dimensions that satisfactory solutions appear. Following this, [8] show that the IDs of PLM adaptations (via fine-tuning) to a single task can be smaller than thousands and pre-training implicitly lowers the IDs of downstream tasks, which motivates this work. The random linear projection used in their methods does not involve any additional training for finding low-dimensional subspaces, hence successfully finding solutions in the subspaces provides ample evidence for the existence of effective low-dimensional reparameterizations. Considering the existence of individual subspace for each task has been proved, here we aim to study whether the subspace is universal. However, the random linear projections of previous methods inevitably introduce redundant task-irrelevant information and make the investigated subspace not compact for reparameterizing task adaptations. Therefore,

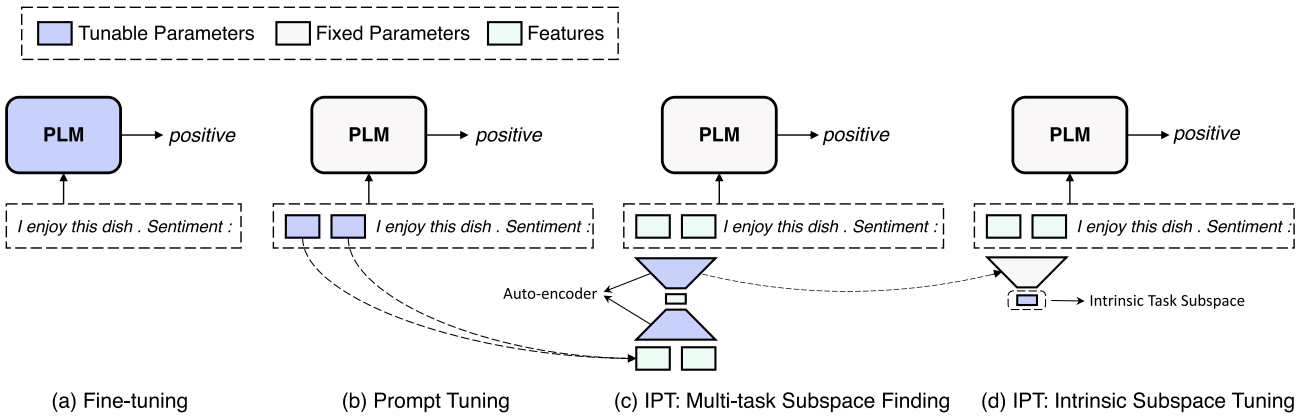


Fig. 2. Illustrations of (a) fine-tuning, (b) prompt tuning and two components of IPT, (c) multi-task subspace finding, and (d) intrinsic subspace tuning. We discriminate tunable parameters, fixed parameters, and intermediate features with different colors.

we resort to more powerful subspace-finding methods and use supervision from diverse tasks to train a nonlinear low-dimensional decomposition for the adaptive parameters. From the point of view of intrinsic dimensionality, the findings of [8] with random projections reveal the upper bound of IDs, while the MSF in our work can be partly seen as finding a lower bound for IDs in prompt tuning for PLMs.

Besides studying the intrinsic dimensionality of objective functions, the other line of research studies the intrinsic dimensionality of data representations [29], [30], [31], [32], [33], [34], [35], which have inspired many practical methods, such as network regularization [36] and network architecture search [37].

### C. Unifying Different NLP Tasks

There has been a long-standing hope to handle broad tasks with a unified general-purpose model. Following the traditional multi-task learning [38] paradigm, various methods have been developed to bring performance improvements by jointly training multiple tasks [39], [40], [41], [42]. Although various NLP tasks differ a lot on the surface, there have been some attempts to unify different NLP tasks into the same task format, such as question answering [43], [44] and natural language inference [45].

Recently, with the development of in-context learning [10], prompting [6], [15], and instruction tuning [46] methods, breaking task boundaries and unifying different tasks have been more and more popular for large language models. Many benchmarks like the CrossFit [47] and MMLU [48] are constructed to evaluate the zero-shot and few-shot generalization abilities of PLMs on extremely multiple tasks. Recent benchmarks, such as PromptSource [49], Natural Instructions [50], BigBench [51], and Super-NaturalInstructions [52], are also equipped with task descriptions, prompts, or instructions to enhance PLMs' zero-shot cross-task generalization abilities. Benefited from this, recently developed large PLMs like T0 [49], FLAN [53], FLAN-T5 [54], and ChatGPT [55] can effectively handle multiple tasks in a unified zero-shot or few-shot manner. The hypothesis and analyses in this paper may help understand these progresses, i.e., why and how PLMs easily adapt to so many tasks.

## III. Methodology

We first introduce essential preliminaries, including PLM, fine-tuning, and prompt tuning in Section III-A. Then we introduce our analysis pipeline **I**ntrinsic **P**rompt **T**uning (**IPT**) in Section III-B, which consists of two stages: (1) Multi-task Subspace Finding (MSF) and (2) Intrinsic Subspace Tuning (IST). In Fig. 2, we visualize the paradigms of fine-tuning, prompt tuning, and our IPT.

### A. Preliminaries

*Pre-trained Language Model:* Assume we are given a series of NLP tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{|\mathcal{T}|}\}$ partitioned into training tasks $\mathcal{T}_{\text{train}}$ and test tasks $\mathcal{T}_{\text{test}}$. Following [3], without loss of generality, we cast each task $\mathcal{T}_i$ $(1 \leq i \leq |\mathcal{T}|)$ into the unified conditional generation format. Specifically, given a training instance $(\mathcal{X}, \mathcal{Y})$ of $\mathcal{T}_i$, where both the input $\mathcal{X}$ and the target $\mathcal{Y}$ consist of a sequence of tokens, i.e., $\mathcal{X} = \{w_1, \ldots, w_{|\mathcal{X}|}\}$ and $\mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$. Our goal is to learn a mapping function $\mathcal{F}_i : \mathcal{X} \rightarrow \mathcal{Y}$, and the de-facto way is to model $\mathcal{F}_i$ with a PLM $\mathcal{M}$, which first converts the input $\mathcal{X}$ into embeddings $\mathbf{E} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{X}|}\} \in \mathbb{R}^{|\mathcal{X}| \times d}$, where $d$ denotes the input embedding dimension. Then $\mathbf{E}$ is sequentially processed by a series of Transformer [56] layers and converted into hidden representations $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{X}|}\} \in \mathbb{R}^{|\mathcal{X}| \times d}$. Finally, the PLM decodes $\mathcal{Y}$ conditioned on $\mathbf{H}$. Formally, the goal is to optimize the following objective:

$$\mathcal{L}_{\text{LM}} = -\sum_{j=1}^{|\mathcal{Y}|} \log p(y_j | w_1, \ldots, w_{|\mathcal{X}|}, y_1, \ldots, y_{j-1}).$$

*Fine-tuning and Prompt Tuning:* In standard fine-tuning, all parameters of $\mathcal{M}$ $(\theta_{\mathcal{M}})$, including the parameters in the embedding layer and the Transformer layers, are tuned when optimizing $\mathcal{L}_{\text{LM}}$. Rather, prompt tuning (PT) prepends some task-specific embeddings (i.e., *soft prompts*) $\mathbf{P}_i = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ parameterized by $\theta_P$ before $\mathbf{E}$, and thus modify the input embeddings into $\mathbf{E}^* = \{\mathbf{p}_1, \ldots, \mathbf{p}_n; \mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{X}|}\} \in \mathbb{R}^{(n+|\mathcal{X}|) \times d}$. Then we keep $\theta_{\mathcal{M}}$ frozen and only tune $\theta_P$ to adapt $\mathcal{M}$ to $\mathcal{T}_i$ during PT. The training objective of PT is essentially the same as $\mathcal{L}_{\text{LM}}$ and

denoted as $\mathcal{L}_{\text{LM}}(\mathbf{P}_i)$. PT tunes substantially fewer parameters than fine-tuning, i.e., $\theta_P \ll \theta_{\mathcal{M}}$.

### B. Intrinsic Prompt Tuning

To verify our hypothesis that the adaptations of PLMs to various downstream tasks can be reparameterized as optimization within a unified low-dimensional *intrinsic task subspace*, we propose a two-phase analysis pipeline IPT. The first phase MSF aims to find the intrinsic task subspace with multiple tasks' prompts, which are defined by an auto-encoder consisting of a projection function and a back-projection function. The second phase IST tunes a low-dimensional vector in the subspace found by MSF and then recovers the vector to soft prompts through the back-projection function. We then introduce both phases in detail.

*Multi-task Subspace Finding:* We first conduct prompt tuning for each downstream task $\mathcal{T}_i$ and obtain the trained soft prompts $\mathbf{P}_i \in \mathbb{R}^{n\times d}$. During MSF, we try to find a satisfactory intrinsic task subspace of a low dimension $d_I$ by learning a decomposition for the matrix $\mathbf{P}_i$. Inspired by text autoencoders [57], the decomposition consists of a projection function $\mathbf{Proj}(\cdot)$ to project $\mathbf{P}_i$ into the $d_I$-dimensional subspace and a back-projection function $\mathbf{Proj}_b(\cdot)$ to project the $d_I$-dimensional vectors back to soft prompts of $\mathcal{T}_i$, and we optimize the reconstruction loss $\mathcal{L}^i_{\text{AE}}$:

$$\mathbf{P}^*_i = \mathbf{Proj}_b(\mathbf{Proj}(\mathbf{P}_i)),$$
$$\mathcal{L}^i_{\text{AE}} = ||\mathbf{P}^*_i - \mathbf{P}_i||^2_2,$$

where $\mathbf{Proj}(\cdot)$ is implemented with a single-layer feed-forward network and $\mathbf{Proj}_b(\cdot)$ is parameterized by a two-layer nonlinear perceptron as follows:

$$\mathbf{Proj}_b(\mathbf{d}_i) = \mathbf{W}_2(\tanh(\mathbf{W}_1\mathbf{d}_i + \mathbf{b}_1)) + \mathbf{b}_2,$$

where $\mathbf{W}_1 \in \mathbb{R}^{d'_I\times d_I}$, $\mathbf{b}_1 \in \mathbb{R}^{d'_I}$, $\mathbf{W}_2 \in \mathbb{R}^{n\times d\times d'_I}$ and $\mathbf{b}_2 \in \mathbb{R}^{n\times d}$ are trainable parameters. $d_I$ denotes the intrinsic dimension investigated in this paper.

Since the desired intrinsic task subspace should work for broad tasks, we introduce multi-task training and also take the task-oriented language modeling losses (using the reconstructed soft prompts) as objective functions. It should be noted that finding the joint decomposition of multiple tasks' prompts is non-trivial. By jointly optimizing the reconstruction losses and the task-oriented losses, the subspace could gain the ability to reparameterize various task adaptations. The overall training objective of MSF is as follows:

$$\mathcal{L}^{\text{MSF}}_{\theta_{\text{proj}}} = \frac{1}{|\mathcal{T}_{\text{train}}|}\sum_{i=1}^{|\mathcal{T}_{\text{train}}|}(\mathcal{L}_{\text{LM}}(\mathbf{P}^*_i) + \alpha\mathcal{L}^i_{\text{AE}}),$$

where $\alpha$ denotes the hyper-parameter controlling the ratio between the two losses, and $\theta_{\text{proj}}$ denotes the parameters of both $\mathbf{Proj}$ and $\mathbf{Proj}_b$. During MSF, we only optimize $\theta_{\text{proj}}$ while keeping other parameters fixed. By introducing downstream task supervision and nonlinearity, we could find more irredundant and effective subspaces than the random linear subspaces [8], and more detailed comparison and discussion are described in Section V-A.

*Intrinsic Subspace Tuning:* In this stage, we want to evaluate if the subspace found by MSF is generalizable to unseen training data of $\mathcal{T}_{\text{train}}$ as well as unseen tasks $\mathcal{T}_{\text{test}}$. If both conditions are satisfied, we can conclude that we successfully find an intrinsic task subspace reparameterizing the adaptations of PLMs to various tasks to some extent. Specifically, we only retain $\mathbf{Proj}_b$ learned during MSF and keep both $\mathbf{Proj}_b$ and the PLM $\mathcal{M}$ frozen. Then for each task $\mathcal{T}_i$, instead of conducting vanilla prompt tuning, we randomly initialize an *intrinsic vector* $\mathbf{V}_i \in \mathbb{R}^{d_I}$ and only tune $d_I$ free parameters in the found subspace. $\mathbf{V}_i$ is projected back to soft prompts with the fixed $\mathbf{Proj}_b$, then the obtained soft prompts are inputted into the PLM to compute the losses. The objective function for training a specific task $\mathcal{T}_i$ during IST is formulated as:

$$\mathcal{L}^{\text{IST}}_{\mathbf{V}_i} = \mathcal{L}_{\text{LM}}(\mathbf{Proj}_b(\mathbf{V}_i)).$$

## IV. Experiment and Analysis

We first describe the experimental settings in Section IV-A, including the tasks and corresponding datasets, evaluation pipeline, evaluation metrics and training details. Then we introduce the experimental results and analyses in IV-B. We explain some of the notations used in this paper in Table I, so that readers could easily navigate through this section.

### A. Experimental Settings

*Tasks and Datasets:* To ensure good coverage for broad NLP tasks, we randomly choose 120 few-shot NLP tasks ($\mathcal{T}_{\text{all}}$, see supplementary material for details) from *CrossFit Gym* [47], including text classification, question answering, conditional generation, etc. To evaluate the generalization ability of IPT, we randomly split the overall task set $\mathcal{T}_{\text{all}}$ into training tasks $\mathcal{T}_{\text{train}}$ and test tasks $\mathcal{T}_{\text{test}}$. We adopt three task splits as listed in Table II to investigate the influence of task types. Each task $\mathcal{T}_i \in \mathcal{T}_{\text{all}}$ consists of a tuple of ($\mathcal{D}^i_{\text{train}}$, $\mathcal{D}^i_{\text{dev}}$, $\mathcal{D}^i_{\text{test}}$), and the sizes of $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{dev}}$ are both set to $K$ for the few-shot setting. For classification and regression tasks, $K = 16$; while for other categories, $K = 32$. The few-shot setting ensures the data scales of tasks are balanced so that the subspace found by MSF will not be biased towards data-rich tasks.

*Evaluation Pipeline. MSF:* During MSF, we first conduct prompt tuning for each task and obtain the soft prompts. Then we train an auto-encoder by jointly decomposing all the soft prompts of $\mathcal{T}_{\text{train}}$. After obtaining the auto-encoder, we apply it to the soft prompts of different tasks to test its reconstruction ability. (1) We first evaluate the reconstructed prompts of $\mathcal{T}_{\text{train}}$ (denoted as $\mathcal{T}_{\text{train}}$(MSF)) to see how much performance we could retain after reconstruction from a $d_I$-dimensional subspace. This performance provides an empirical upper bound for the generalization to unseen data and unseen tasks in our setting. (2) We also directly reconstruct the soft prompts of $\mathcal{T}_{\text{test}}$ with the learned auto-encoder and test their performance ($\mathcal{T}_{\text{test}}$(MSF)) to see the auto-encoder's reconstruction ability for unseen soft prompts.

*Evaluation Pipeline. IST:* During IST, we investigate whether adaptations to various tasks can be reparameterized into the

TABLE I
DESCRIPTIONS ABOUT THE NOTATIONS USED IN THIS PAPER

| Notation | Description |
|---|---|
| IPT | The analysis pipeline proposed in this paper (Intrinsic Prompt Tuning). |
| MSF | The first stage of IPT (Multi-task Subspace Finding). |
| IST | The second stage of IPT (Intrinsic Subspace Tuning). |
| $\mathcal{D}^i_{\text{train}}$ | The training set of the task $\mathcal{T}_i$. |
| $\mathcal{D}^i_{\text{dev}}$ | The development set of the task $\mathcal{T}_i$. |
| $\mathcal{D}^i_{\text{test}}$ | The test set of the task $\mathcal{T}_i$. |
| $E^{\text{FT}}_{\text{rel}}$ | Average relative performance of IPT compared with fine-tuning (FT). |
| $E^{\text{PT}}_{\text{rel}}$ | Average relative performance of IPT compared with prompt tuning (PT). |
| *random* | Randomly split tasks $\mathcal{T}_{\text{all}}$ into training tasks $\mathcal{T}_{\text{train}}$ and test tasks $\mathcal{T}_{\text{test}}$. |
| *non-cls* | Choose a subset of non-classification tasks as the training tasks $\mathcal{T}_{\text{train}}$. |
| *cls* | Choose a subset of classification tasks as the training tasks $\mathcal{T}_{\text{train}}$. |
| $\mathcal{T}_{\text{train}}$(MSF) | Reconstructing the trained soft prompts of training tasks by training an auto-encoder. |
| $\mathcal{T}_{\text{test}}$(MSF) | Testing the generalization of the trained auto-encoder on test tasks. |
| $\mathcal{T}^{\text{same}}_{\text{train}}$(IST) | Conducting IST for each training task with the same training data used in MSF. |
| $\mathcal{T}^{\text{diff}}_{\text{train}}$(IST) | Conducting IST for each training task with the different training data used in MSF (*unseen-data challenge*). |
| $\mathcal{T}_{\text{test}}$(IST) | Conducting IST for each test task (*unseen-task challenge*). |
| $\mathcal{T}^{\text{in}}_{\text{test}}$(IST) | Conducting IST for each test task belonging to the same categories of training tasks (*unseen-task challenge*). |
| $\mathcal{T}^{\text{out}}_{\text{test}}$(IST) | Conducting IST for each test task belonging to different categories of training tasks (*unseen-task challenge*). |

TABLE II
THE OVERALL 120 TASKS $\mathcal{T}_{\text{ALL}}$ CONSIST OF 43 CLASSIFICATION TASKS (CLS.) AND 77 NON-CLASSIFICATION TASKS (NON-CLS.)

| Shorthand | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}$ |
|---|---|---|
| *random* | 100 random | 20 random |
| *non-cls* | 35 non-cls. | 42 non-cls.($\mathcal{T}^{\text{in}}_{\text{test}}$) / 43 cls.($\mathcal{T}^{\text{out}}_{\text{test}}$) |
| *cls* | 35 cls. | 8 cls.($\mathcal{T}^{\text{in}}_{\text{test}}$) / 77 non-cls.($\mathcal{T}^{\text{out}}_{\text{test}}$) |

Three task splits are evaluated, including *random*, *non-cls* and *cls*, with details listed above, E.G., For *non-cls* partition, 35 non-cls. Are chosen as $\mathcal{T}_{\text{train}}$ and 42 non-cls. / 43 cls. Are chosen as $\mathcal{T}^{\text{in}}_{\text{test}}$ / $\mathcal{T}^{\text{out}}_{\text{test}}$, respectively.

found subspace. We first conduct IST for $\mathcal{T}_{\text{train}}$ using exactly **the same** $\mathcal{D}^i_{\text{train}}$ / $\mathcal{D}^i_{\text{dev}}$ utilized in MSF training, and get the result $\mathcal{T}^{\text{same}}_{\text{train}}$(IST).

Then we evaluate the generalization ability of IPTwith two challenges: (1) *unseen-data challenge* and (2) *unseen-task challenge*.

• For the *unseen-data challenge*, we sample **different** $\mathcal{D}^i_{\text{train}}$ / $\mathcal{D}^i_{\text{dev}}$ for $\mathcal{T}_{\text{train}}$ while keeping test data the same. Then we conduct IST with the new data and test its performance on $\mathcal{T}_{\text{train}}$, which is denoted as $\mathcal{T}^{\text{diff}}_{\text{train}}$(IST). This challenge evaluates whether the learned subspace can also reparameterize optimization on unseen data. This challenge is important in that different training data often leads to distinct optimization trajectories.

• For the *unseen-task challenge*, we evaluate the soft prompts obtained by IST on $\mathcal{T}_{\text{test}}$, which are tasks unseen during MSF. We aim to investigate how well can optimization in the found subspace recover PLM adaptations of unseen tasks, which will provide evidence for our hypothesis that the reparameterization subspaces for different task adaptations are not orthogonal. As shown in the task splits in Table II, for the *random* split, the results of this challenge are denoted as $\mathcal{T}_{\text{test}}$(IST); for the *non-cls* and *cls* splits, we have two test sets with different task types and the corresponding results are denoted as $\mathcal{T}^{\text{in}}_{\text{test}}$(IST) and $\mathcal{T}^{\text{out}}_{\text{test}}$(IST), respectively. $\mathcal{T}^{\text{in}}_{\text{test}}$(IST) means we conduct IST for test tasks belonging to the same categories of $\mathcal{T}_{\text{train}}$, while $\mathcal{T}^{\text{out}}_{\text{test}}$(IST) means we conduct IST for test tasks that belong to different categories of $\mathcal{T}_{\text{train}}$.

*Evaluation Metrics:* Since different tasks have distinct evaluation protocols (e.g., F1 score for discriminative tasks and BLEU for generative tasks typically), following [47], we mainly choose average relative performance ($E_{\text{rel}}$) as the evaluation metric, instead of absolute performance, which is reported in Section A1. Specifically, let $\mathcal{T}=\{\mathcal{T}_1,\ldots,\mathcal{T}_{|\mathcal{T}|}\}$ be the evaluated tasks and $E_{\mathcal{T}_i}$ denotes the test score of $\mathcal{T}_i$ for IPT(either in MSF or IST), $E^*_{\text{rel}}=\frac{1}{|\mathcal{T}|}\sum_{\mathcal{T}_i\in\mathcal{T}}\frac{E_{\mathcal{T}_i}}{E^*_{\mathcal{T}_i}}$, where $E^*_{\mathcal{T}_i}$ denotes the performance of $T_i$ using either prompt tuning ($E^{\text{PT}}_{\mathcal{T}_i}$) or fine-tuning ($E^{\text{FT}}_{\mathcal{T}_i}$). $E^{\text{PT}}_{\text{rel}}$ / $E^{\text{FT}}_{\text{rel}}$ denotes our IPT's relative performance to prompt tuning / fine-tuning.

*Training Details:* Without loss of generality, we follow [47] to use $\text{BART}_{\text{BASE}}$ [58] for the experiments in the main paper, and unify all tasks into the same sequence-to-sequence format following [3] and [59]. We also test $\text{BART}_{\text{LARGE}}$ in Appendix A3.

For prompt tuning / fine-tuning, we perform a grid search on the combination of a series of learning rates ($\{1\times10^{-5}, 2\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}\}$) and batch sizes ($\{2,4,8\}$), choose the best checkpoint using $\mathcal{D}_{\text{dev}}$, and evaluate it on $\mathcal{D}_{\text{test}}$. We set the number of soft prompts to be 100 for all tasks and randomly initialize them. For IPT, we examine the dimension $d_I$ of $\{5,10,50,100,250\}$. We set the inner hidden size $d'_I$ of $\mathbf{Proj}_b$ to 768. Note that for fine-tuning / prompt tuning, 139 M / 76,800 parameters are tuned, while IPTonly tunes $d_I$ free parameters.

We adopt AdamW [60] as the optimizer, and train all models under the same environment of NVIDIA 32 GB V100 GPU. We set the max training step to 10,000 / 100,000 and validate on $\mathcal{D}_{\text{dev}}$ every 100 / 1000 steps for fine-tuning / prompt tuning,

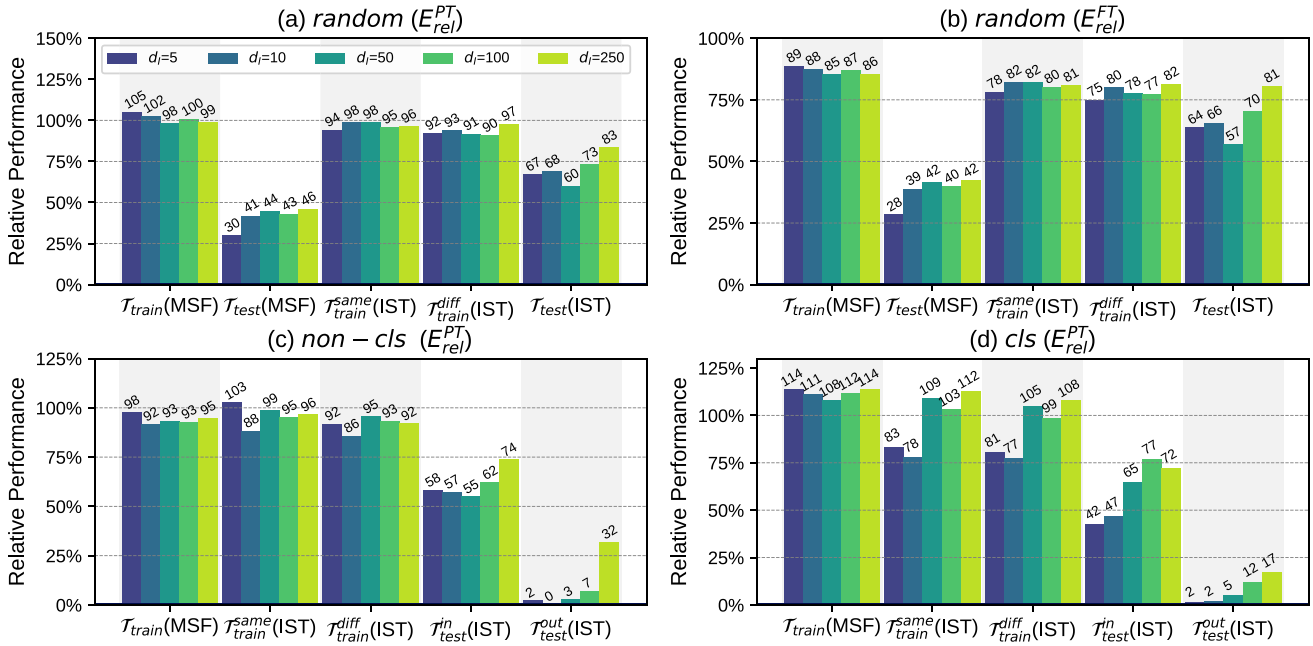


Fig. 3. Relative performance of IPTat different dimension $d_I$ on three task splits (*random*, *non-cls* and *cls*). We report the relative performance of IPTcompared with both prompt tuning ($E_{\text{rel}}^{\text{PT}}$) and fine-tuning ($E_{\text{rel}}^{\text{FT}}$). Different notions are summarized in Table I.

since prompt tuning requires far more training steps than fine-tuning to converge [61]. We conduct fine-tuning / prompt tuning for different tasks individually and then calculate the average performance. The ratio $\alpha$ that is used to balance $\mathcal{L}_{\text{LM}}(\mathbf{P}_i^*)$ and $\mathcal{L}_{\text{AE}}^i$ is set to 200.[1] The hyper-parameters of IST are chosen as the same as prompt tuning for fair comparisons. To avoid possible information leakage, we discard the low-dimensional solutions (intrinsic vectors) found during MSF, and randomly initialize the intrinsic vector for IST.

### B. Main Results

Based on the experimental results shown in Fig. 3, we study the following questions:

**Q1. Do PLMs really reparameterize various task adaptations into a universal task subspace?** From the results in Fig. 3(a), we observe that: (1) for the *unseen-data challenge* ($\mathcal{T}_{\text{train}}^{\text{diff}}$(IST)), IST on unseen i.i.d. data could recover more than 90% of the full prompt tuning performance of the 100 training tasks; (2) for the *unseen-task challenge* ($\mathcal{T}_{\text{test}}$(IST)), we can also achieve 83% performance by only tuning 250 parameters. From these results, we can say that the low-dimensional reparameterizations in the subspaces found by MSF successfully recover the PLM adaptations of $\mathcal{T}_{\text{train}}$ and can also generalize to unseen tasks. Thus non-trivial performance can be achieved by only tuning a few free parameters in these subspaces. This strongly supports our hypothesis that PLMs reparameterize various task adaptations into the same low-dimensional subspace, or at least the low-dimensional reparameterization subspaces for various task adaptations [8] have a substantial intersection, otherwise, the subspace found with $\mathcal{T}_{\text{train}}$ in MSF would be improbable to work for $\mathcal{T}_{\text{test}}$.

**Q2. What limits IPT?** Although strong evidence is observed, the effectiveness of IPT could still be improved, especially at low dimensions. From the results in Fig. 3(a) and (b), we discuss what factors may limit the effectiveness and provide insights for improving the analysis pipeline.

1) *Reconstruction ability of the auto-encoder:* The performance on $\mathcal{T}_{\text{train}}$ when we directly reconstruct soft prompts using the auto-encoder of MSF ($\mathcal{T}_{\text{train}}$(MSF)) can be even better than vanilla prompt tuning (PT). This is because MSF explicitly enforces multi-task knowledge sharing within a unified subspace. Such knowledge sharing equips the subspace with better representational capabilities, making the subspace more universal. Thus the performance of MSF could even exceed conducting vanilla PT for each individual task. However, from the comparisons between $\mathcal{T}_{\text{train}}$(MSF) and $\mathcal{T}_{\text{test}}$(MSF), we can see that directly reconstructing soft prompts of unseen tasks performs poorly. It indicates that the reconstruction ability of the auto-encoders trained in MSF cannot generalize well to unseen soft prompts, which will limit IPTto some extent. This may come from the limited representation ability of the networks used to parameterize $\mathbf{Proj}(\cdot)$ and $\mathbf{Proj}_b(\cdot)$. Nevertheless, IST could find better solutions ($\mathcal{T}_{\text{test}}$(IST)) than MSF reconstructed prompts ($\mathcal{T}_{\text{test}}$(MSF)) with task-specific supervisions on $\mathcal{T}_{\text{test}}$.

2) *Optimization in IST:* The performance of $\mathcal{T}_{\text{train}}$(MSF) is always near 100%, which demonstrates that there exists good enough solutions of $\mathcal{T}_{\text{train}}$ in the found subspace. However, even using exactly the same training data, IST cannot find these good solutions (i.e., the gap between $\mathcal{T}_{\text{train}}$(MSF) and $\mathcal{T}_{\text{train}}^{\text{same}}$(IST)), which shows that the adopted optimization algorithm may limit

[1]Note when calculating the AE loss, we have already divided it by 768 (the hidden size), hence 200 * AE loss is actually smaller than the LM loss after sufficient training.

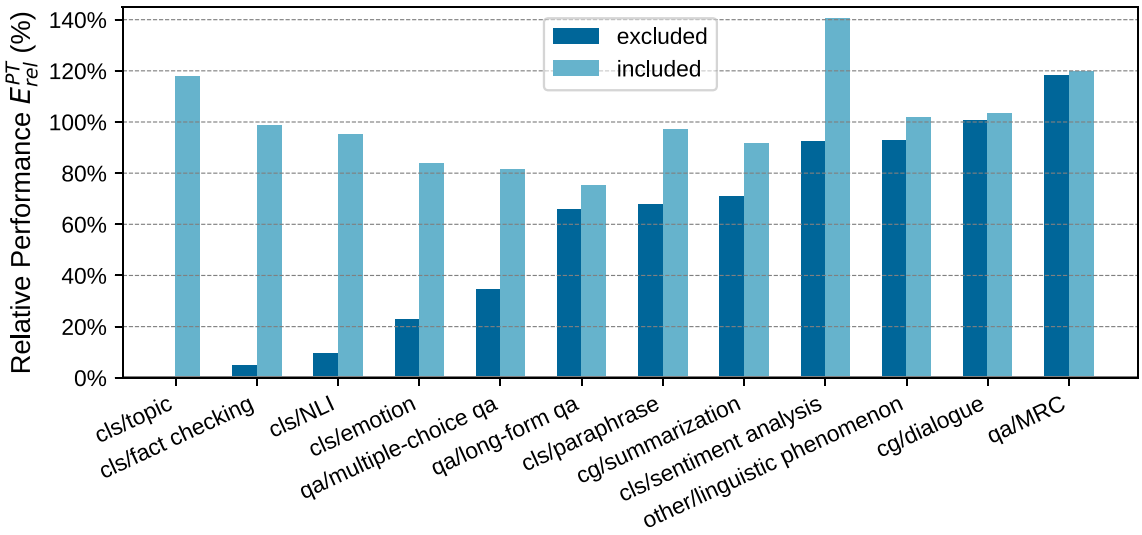


Fig. 4. $E_{\text{rel}}^{\text{PT}}$ performance of IPTon tasks grouped by fine-grained task types. *included* means that MSF is conducted on the training tasks of the *random* split (including all the task types). *excluded* means that MSF is conducted on the training tasks of the *random* split excluding the tasks of the investigated type. Note the performance of IST ($E_{\text{rel}}^{\text{PT}}$) is tested on the tasks of the investigated type.

IST's performance. We also observe that with $d_I$ increasing, the recovering performance of IST generally becomes better, which is because a higher dimension brings larger representational capacities. (3) **Adaptive parameters.** Comparing the results in Fig. 3(a) and (b), we observe that the relative performance of fine-tuning ($E_{\text{rel}}^{\text{FT}}$) is always poorer than that of PT ($E_{\text{rel}}^{\text{PT}}$). Since both $E_{\text{rel}}^{\text{FT}}$ and $E_{\text{rel}}^{\text{PT}}$ have the same numerator, the above phenomenon is because PT is slightly inferior to fine-tuning under the few-shot setting. Since the performance of IPTis bounded by PT, ideally, $E_{\text{rel}}^{\text{FT}}$ could be improved by designing better PT algorithms or selecting more appropriate adaptive parameters (i.e., utilizing other parameter-efficient tuning methods). More detailed comparisons and discussions are in Section V-A. **Q3. What is the influence of task types?** We first divide the studied tasks into *cls.* (classification), which are discriminative tasks, and *non-cls.* (non-classification), which tend to be generative tasks. From the results in Fig. 3(c)-(d), we find that there exists a huge generalization gap between the two coarse-grained task types. When using only one kind of tasks during MSF, the found subspaces work well for the same kind of tasks ($\mathcal{T}_{\text{test}}^{\text{in}}(\text{IST})$) but generalize poorly to the other kind of tasks ($\mathcal{T}_{\text{test}}^{\text{out}}(\text{IST})$). This shows that the found subspace is severely biased by the training task types.

*Fine-grained Analysis on Task Types:* Besides the cls. and non-cls. categorization, we further conduct more fine-grained analyses on task types to fathom their influence. Specifically, we choose 6 cls. task types (cls/topic, cls/fact checking, cls/NLI, cls/emotion, cls/paraphrase, cls/sentiment analysis) and 6 non-cls. task types (qa/multiple-choice qa, qa/long-form qa, cg/summarization, other/linguistic phenomenon, cg/dialogue, qa/MRC). Both $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{test}}$ contain the tasks from the above types. Please refer to the detailed task information in the supplementary material. We report the relative performance of IPTcompared with prompt tuning ($E_{\text{rel}}^{\text{PT}}$) on these fine-grained task types, including two settings:

• *included* means that MSF is conducted on the training tasks $\mathcal{T}_{\text{train}}$ of the *random* split (100 tasks in total, including all the task types), and IST is conducted on the test tasks $\mathcal{T}_{\text{test}}$ of each investigated type;

• *excluded* means that MSF is conducted on the training tasks excluding the tasks of the investigated type. In other words, we exclude the investigated type of tasks from the training tasks to form the new training task set. The experiments of IST are conducted using the tasks of the investigated type. The test tasks of both settings (*included* and *excluded*) are the same.

Intuitively, if we exclude a specific task type from the training tasks, the approximated intrinsic task subspace may fail to include the language skills required by this task type, and would thus result in poor recovering performance during IST on the investigated task type. However, if other tasks in the training set share similar language skills to the investigated task type, the above issue could be mitigated to some extent. In this sense, the performance gap between *excluded* and *included* of a task type reflects the relation (common language skills) of the investigated type to other task types.

From the results shown in Fig. 4, we can observe that: (1) for some task types, there exists a huge gap between *excluded* and *included* performance, such as cls/topic, cls/fact checking, and cls/NLI, which may come from the unique skills required to solve these tasks; (2) instead, some tasks tend to require similar language skills than other task types (reflected in the close performance between *excluded* and *included*), such as other/linguistic phenomenon, cg/dialogue, and qa/MRC. The above results demonstrate the potential of IPTto analyze task similarities and differences; (3) IPTachieves evident improvements compared to vanilla prompt tuning (i.e., $E_{\text{rel}}^{\text{PT}} > 100\%$ for the *included* setting) on some task types such as cls/topic, cls/sentiment analysis, and qa/MRC, which indicates that tuning PLMs within the intrinsic task subspace is promising to obtain performance improvements.

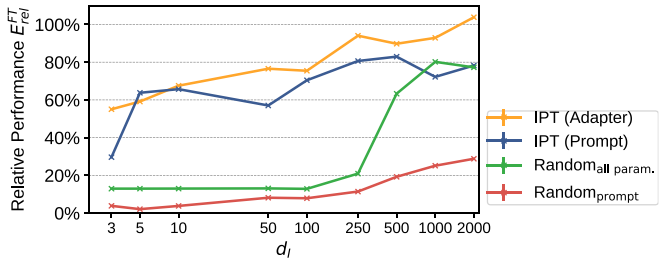


Fig. 5. Comparisons between IPT(IPT (Adapter) and IPT (Prompt)) and other subspace-finding methods (Random$_{\text{prompt}}$ and Random$_{\text{all_param.}}$) on $\mathcal{T}_{\text{test}}$ of *random* task split. We experiment with different intrinsic dimensions ($d_I$). We report the relative performance compared with fine-tuning ($E_{\text{rel}}^{\text{FT}}$).

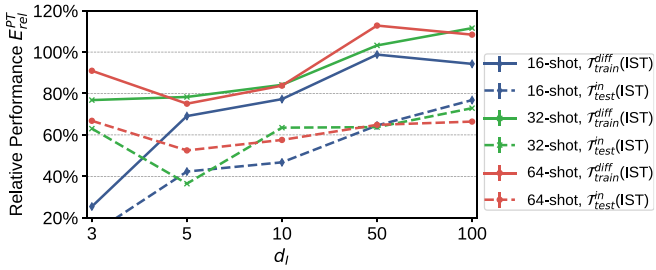


Fig. 6. Impacts of the data scale. We experiment on the task split *cls* and double / quadruple the number of data shots of both training and test tasks. Different intrinsic dimensions ($d_I$) are tested and we report the relative performance of prompt tuning ($E_{\text{rel}}^{\text{PT}}$).

## V. Additional Analyses

### A. Comparison of Subspace-Finding Methods

In Fig. 5, we compare the relative fine-tuning performance ($E_{\text{rel}}^{\text{FT}}$) of subspaces found by IPTwith various subspace-finding methods on the *unseen-task* challenge under $\mathcal{T}_{\text{test}}$ of *random* split. First, we investigate the randomly generated subspaces of [8]: the first trial Random$_{\text{prompt}}$ conducts IST within random subspaces of *soft prompts*. The subspaces are defined by randomly initialized auto-encoders of the same architecture as MSF; the second trial Random$_{\text{all_param.}}$ conducts IST within random subspaces of *all PLM parameters*, which are generated by the efficient Fastfood transform [62]. We can see that the random subspace-finding methods can also find effective unified reparameterization subspaces at large dimensionality, which supports our universal reparameterization subspace hypothesis. In addition, IPTperforms much better than Random$_{\text{prompt}}$ and Random$_{\text{all_param.}}$, especially at low dimensions, which indicates the effectiveness of MSF to exclude redundant task-irrelevant information and find compact subspaces.

After that, we explore another representative parameter-efficient tuning method, Adapter [63], as the backbone of our method, i.e., we conduct both MSF and IST using the adapter parameters instead of soft prompts. For a brief introduction, Adapter plugs in lightweight feed-forward networks between Transformer layers (both after the multi-head attention module and the feed-forward module). Every Adapter module consists of a down-projection matrix $\mathbf{W}_{\text{down}} \in \mathbb{R}^{r_{\text{A}} \times d}$, a non-linear activation function $f(\cdot)$, and a up-projection matrix $\mathbf{W}_{\text{up}} \in \mathbb{R}^{d \times r_{\text{A}}}$, where $r_{\text{A}}$ denotes the bottleneck dimension, and $d$ denotes the hidden size of the PLM. Given an input $\mathbf{h}$, adapter applies a residual connection as follows:

$$\mathbf{h} \leftarrow \mathbf{h} + \mathbf{W}_{\text{up}} f(\mathbf{W}_{\text{down}} \mathbf{h}).$$

The essence of IPTis to define a projector from the intrinsic task subspace to the original parameter space. For Adapter, the parameter space is decided by the newly introduced matrices $\mathbf{W}_{\text{down}}$ and $\mathbf{W}_{\text{up}}$ in each layer. During the first stage MSF, we reparameterize both $\mathbf{W}_{\text{down}}$ and $\mathbf{W}_{\text{up}}$ as the product of a low-dimensional intrinsic vector $\mathbf{V} \in \mathbb{R}^{d_I}$, lying in the intrinsic task subspace, and the corresponding projection matrices as follows:

$$(\mathbf{W}_{\text{up}}; \mathbf{W}_{\text{down}}) = \mathbf{Proj}(\mathbf{V}),$$

where the projection **Proj** is implemented by a two-layer perceptron. During the second stage IST, only a $d_I$-dimensional (randomly initialized) intrinsic vector $\mathbf{V}$ is optimized. Other implementation details are kept the same as IPT(Prompt).

From Fig. 5, we observe that this method (IPT (Adapter)) performs consistently better than the original IPTusing soft prompts (IPT (Prompt)) and can even outperform fine-tuning at 2000 dimensionality (i.e., $E_{\text{rel}}^{\text{FT}} > 100\%$), which may be due to the better performance of adapter than prompt tuning [64]. This further shows that IPTis agnostic to the specific tuning method and provides stronger empirical evidence for our research hypothesis.

### B. Impacts of the Data Scale

Although we adopt the few-shot setup to control the influence of data amount in this paper, it is also interesting to investigate IPT's ability given more training data. Here we take an initial trial using the task split *cls* by doubling / quadrupling the number of data shots $K$ of both seen and unseen tasks (from 16 to 32 / 64), and investigate the performance of IPTunder the *unseen-data* ($\mathcal{T}_{\text{train}}^{\text{diff}}$(IST)) and *unseen-task* ($\mathcal{T}_{\text{test}}^{\text{in}}$(IST)) challenges. Note that with different number of data points, the PT performance (denominator of $E_{\text{rel}}^{\text{PT}}$) is also different. The results are shown in Fig. 6. We can see that when the data scale grows, the performance of IPTgenerally becomes better, especially at low dimensions. This shows that the subspace approximated with more data is more universal. Note in this paper, we mainly focus on the few-shot learning setting, we encourage future work to explore how strong the performance of IPTon data-rich scenarios will be.

### C. Impacts of the Number of Training Tasks

During MSF, the auto-encoder is optimized to reparameterize the adaptive parameters of various training tasks. Ideally, the coverage of $\mathcal{T}_{\text{train}}$ would significantly impact the generalization ability of IPTon unseen tasks $\mathcal{T}_{\text{test}}$. To demonstrate this, we randomly sample $\{20\%, 40\%, 60\%, 80\%\}$ tasks from $\mathcal{T}_{\text{train}}$ of the *random* task split to train the auto-encoder, then evaluate IPT($d_I = \{10, 100\}$) on original $\mathcal{T}_{\text{test}}$ with the *unseen-task* challenge. From the results visualized in Fig. 7, we observe that with the number of training tasks growing, the generalization ability

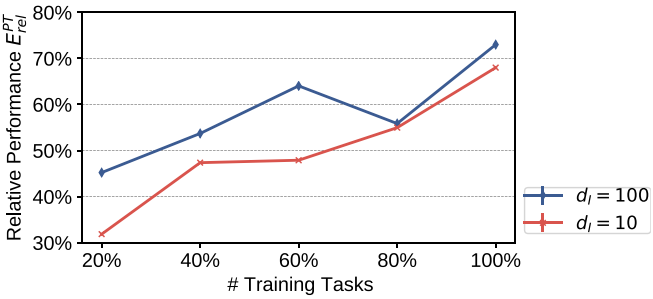


Fig. 7. Impacts of the number of training tasks. $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ tasks are randomly sampled from $\mathcal{T}_{\text{train}}$ of the *random* task split to train the auto-encoder. We report the relative performance of prompt tuning ($E_{\text{rel}}^{\text{PT}}$) for the *unseen-task* challenge. Two intrinsic dimensions are evaluated, i.e., $d_I = 10100$.

TABLE III
STANDARD DEVIATIONS (std) OF TEST SCORES OVER 10 RUNS FOR DIFFERENT TASKS

| **Method** | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}$ | $\mathcal{T}_{\text{all}}$ |
|---|---|---|---|
| Fine-tuning | 2.16 | 2.40 | 2.20 |
| Prompt Tuning | 3.06 | 4.19 | 3.25 |
| IPT | **1.12** | **0.73** | **1.06** |

We compare fine-tuning, prompt tuning, and IPT. $d_I$ of IPT is chosen to be 10.

of the found intrinsic task subspace generally improves. This reflects that increasing the coverage and diversity of seen tasks could help IPTfind more universal subspaces.

### D. Visualization of the Found Intrinsic Subspace

We visualize the intrinsic vectors $\mathbf{V}_i$ (the tunable parameters learned during IST in the found subspace) using PCA in Fig. 8, where we illustrate the data points belonging to different task categories. The conclusions are summarized as follows: (1) from Fig. 8(a), there exists a clear dividing line between the clusters of classification tasks and non-classification tasks, indicating that both task types are highly distinct by nature. This also explains why the subspace learned on one cluster generalizes poorly to the other cluster in Fig. 3; (2) from Fig. 8(b), the points of unseen tasks $\mathcal{T}_{\text{test}}$ are mixed with those of $\mathcal{T}_{\text{train}}$, which demonstrates that the found subspaces universally reparameterize various tasks so that IPTcan generalize well to unseen tasks; (3) from Fig. 8(c) and (d), we find that the points belonging to the same task category exhibit a compact cluster. Given these results, we contend that the learned intrinsic vectors could be viewed as low-dimensional task representations, helping us analyze the similarity and differences of various NLP tasks.

### E. Improving Prompt Tuning Stability With IPT

In Table III, we show the mean standard deviations (std) of test scores for 120 few-shot tasks over 10 runs comparing IPT($d_I = 10$), fine-tuning and prompt tuning (PT). We observe that PT is the most unstable strategy with the highest std, while fine-tuning is far more stable. The instability of PT may influence its practical uses. Intuitively, IPTonly tunes a few free parameters, which will conduce to improving the stability, and IPTis the most stable method in Table III.

TABLE IV
CONDUCTING PROMPT TUNING / FINE-TUNING ON THE RANDOM SPLIT

| **Split** | **Prompt Tuning** | | **Fine-tuning** | |
|---|---|---|---|---|
| | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}$ | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}$ |
| *individual* | 32.6 | 40.1 | 35.2 | 40.7 |
| *all* | 33.4 | 41.2 | 35.5 | 40.9 |

We choose either to train on individual tasks or all tasks.

Furthermore, we propose to use the solutions found by IPTas the initialization for the vanilla PT. Specifically, we conduct vanilla PT of split *random* on $\mathcal{T}_{\text{test}}$ choosing $d_I = 10$ and initialize the soft prompts by back-projecting the IST solutions in the subspace. Other details are kept the same as the PT baseline. We observe that the std achieved in this way is significantly lower than the vanilla PT (1.65 v.s. 4.19) and we can achieve 103.4% of $E_{\text{rel}}^{\text{PT}}$, i.e., the performance could also be improved. This indicates that IPTand vanilla PT can be combined in a two-stage manner to improve the training stability and achieve satisfying performance. This experiment also demonstrates that although our IPTpipeline mainly works as an analytical framework in this paper, it can also bring practical benefits.

### F. FT/PT on All Tasks V.s. FT/PT on Each Individual Task

In all the above experiments, the baseline method (prompting and fine-tuning) is trained for different tasks individually, i.e., we conduct 120 experiments in total. Here we evaluate another baseline method by FT/PT the model on all tasks in a multi-task way. We choose the random split and compare the performance of FT/PT on all tasks v.s. FT/PT on each individual task. The results are shown in Table IV, from which we observe that training on all tasks performs slightly better than the original baseline. The reason is that this facilitates knowledge sharing in the same parameter space. However, considering that the scores are very close to the original numbers, we argue that our conclusions still hold if we choose a different baseline in the above experiments.

## VI. DISCUSSION

*Relation to the scaling law:* Recently, researchers have found that larger PLMs tend to be more sample-efficient [65], parameter-efficient [6], and cross-task generalizable [66]. Our hypothesis may help us understand this phenomenon: the adaptations of larger PLMs can be better reparameterized into a unified subspace so that the cross-task generalization will be easier, and previous work [8] show larger PLMs have lower reparameterization dimensions, hence they should need fewer data and tunable parameters. This also implies that the characteristics of intrinsic task subspaces could be used to examine how well a PLM is trained.

*Utilize and manipulate intrinsic vectors:* The intrinsic vectors obtained during IST depict the adaptations to different tasks and it is worthwhile to explore whether we can (1) utilize them to find the relations among different tasks, and (2) manipulate these

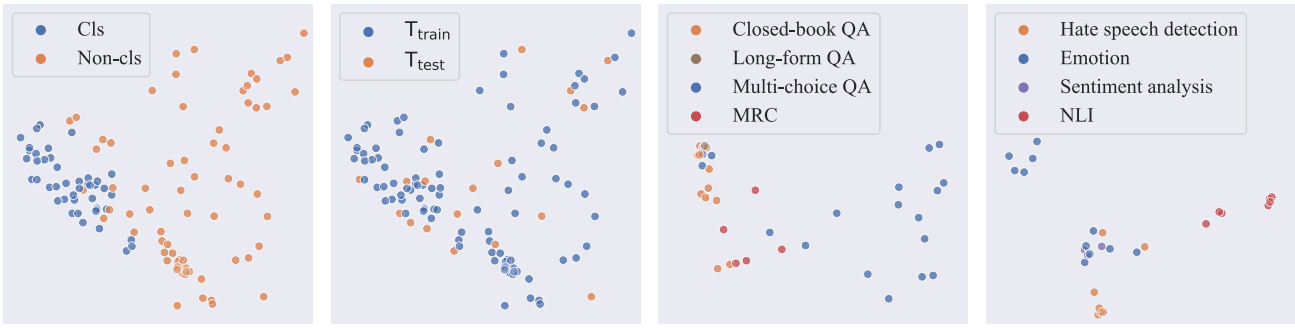


Fig. 8. PCA plots of the intrinsic vectors learned during IST. We label points with different colors to represent their corresponding categories. Specifically, we show the clusters of (a) classification and non-classification tasks, (b) $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{test}}$, (c) question answering tasks, and (d) text classification tasks. Without loss of generality, we choose the task split of *random* and $d_I = 100$.

vectors to achieve cross-task generalization. We also encourage future works to explore more methods to tune PLMs within low-dimensional intrinsic task subspaces, which may have some practical benefits such as avoiding over-parameterization and being more environmentally friendly with fewer tunable parameters.

*Towards a Unified View of Parameter-efficient Tuning:* As shown in Section V-A, our IPTpipeline can not only be applied to prompt tuning, but also works for a representative parameter-efficient tuning (PET) method: Adapter. This shows that IPT-pipeline is agnostic to the specific tunable parameters. From another perspective, this finding implies that different PET methods are intrinsically connected with each other [67]. Inspired by our work, [68] explore the existence of a unified optimization subspace for different PETs and demonstrate that optimizing various PETs can all be seen as finding optimal solutions within a unified subspace. This finding indicates the close connections among various PETs, disclosing the underlying mechanism of PLMs' downstream tuning process.

## VII. Conclusion

In this paper, we study the hypothesis that PLM adaptations to various tasks can be reparameterized as optimizations within a **unified** low-dimensional *intrinsic task subspace*. We develop an analysis tool IPT. It first finds a subspace by jointly decomposing the adaptive parameters of multiple tasks and then tunes parameters within the subspace for unseen data and tasks. In experiments, we study diverse few-shot NLP tasks and demonstrate that the found subspaces contain good solutions for PLM adaptations, which is strong evidence for our hypothesis. In addition, we conduct sufficient analyses to understand the effect of training task types, the number of training tasks, and training data scales for IPT. We also discuss the potential practical use of IPT, such as analyzing task differences and improving training stability. The analyses and conclusions in this paper may help us fathom the inner workings of PLM adaptation and facilitate relevant future explorations.

TABLE V
Average Absolute Performance for Prompt Tuning / Fine-Tuning on the Three Task Splits We Adopted

| Split | Prompt Tuning | | Fine-tuning | |
|---|---|---|---|---|
| | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}^{\text{in}}$ / $\mathcal{T}_{\text{test}}^{\text{out}}$ | $\mathcal{T}_{\text{train}}$ | $\mathcal{T}_{\text{test}}^{\text{in}}$ / $\mathcal{T}_{\text{test}}^{\text{out}}$ |
| *random* | 32.6 | 40.1 ($\mathcal{T}_{\text{test}}$) | 35.2 | 40.7 ($\mathcal{T}_{\text{test}}$) |
| *non-cls* | 23.0 | 28.0 / 49.0 | 24.4 | 29.6 / 52.2 |
| *cls* | 48.6 | 50.9 / 25.7 | 52.5 | 51.1 / 27.2 |

TABLE VI
Average Absolute Performance on the *Random* Task Split

| **Dim** ($d_I$) | 5 | 10 | 50 | 100 | 250 |
|---|---|---|---|---|---|
| *Multi-task Subspace Finding* | | | | | |
| $\mathcal{T}_{\text{train}}$ | 31.8 | 32.2 | 32.0 | 32.6 | **33.0** |
| $\mathcal{T}_{\text{test}}$ | 10.0 | 15.0 | **16.7** | 16.4 | 16.6 |
| *Intrinsic Subspace Tuning* | | | | | |
| $\mathcal{T}_{\text{train}}^{\text{same}}$ | 27.9 | 32.1 | 33.0 | **33.0** | 31.7 |
| $\mathcal{T}_{\text{train}}^{\text{diff}}$ | 28.1 | 31.2 | 31.2 | **32.1** | 32.0 |
| $\mathcal{T}_{\text{test}}$ | 28.0 | 26.8 | 24.0 | 26.3 | **33.7** |

## Appendix

### A. Absolute Performance

In the experiments of the main paper, we report the relative performance ($E_{\text{rel}}$). For reference, we also report the average absolute performance ($E_{\text{abs}}$) in this section. Let $E_{\mathcal{T}_i}$ denote the test score of $\mathcal{T}_i$ for IPT, $E_{\text{abs}} = \frac{1}{|\mathcal{T}|}\sum_{\mathcal{T}_i \in \mathcal{T}} E_{\mathcal{T}_i}$. The $E_{\text{abs}}$ of $\text{BART}_{\text{BASE}}$ for prompt tuning and fine-tuning are shown in Table V, and the $E_{\text{abs}}$ of IPTon three task splits are shown in Tables VI, VII and VIII, respectively.

### B. Relative Performance to Fine-Tuning

In the experiments of the main paper, we report the relative performance to prompt tuning as the main evaluation metric except in Fig. 3(b), which reports the relative performance to fine-tuning on the *random* split for analyses. Here we additionally report $E_{\text{rel}}^{\text{FT}}$ on *non-cls* and *cls* splits in Fig. 9 for reference,

TABLE VII
AVERAGE ABSOLUTE PERFORMANCE ON THE *NON-CLS* TASK SPLIT

| **Dim** ($d_I$) | 5 | 10 | 50 | 100 | 250 |
|---|---|---|---|---|---|
| *Multi-task Subspace Finding* | | | | | |
| $\mathcal{T}_{\text{train}}$ | 23.1 | 21.9 | 22.7 | 22.2 | **22.7** |
| *Intrinsic Subspace Tuning* | | | | | |
| $\mathcal{T}_{\text{train}}^{\text{same}}$ | **27.4** | 23.4 | 27.2 | 26.2 | 26.8 |
| $\mathcal{T}_{\text{train}}^{\text{diff}}$ | 25.8 | 22.8 | 25.4 | 25.8 | **26.6** |
| $\mathcal{T}_{\text{test}}^{\text{in}}$ | 17.4 | 17.7 | 18.6 | 21.5 | **24.2** |
| $\mathcal{T}_{\text{test}}^{\text{out}}$ | 1.0 | 0.8 | 1.4 | 3.9 | **15.8** |

TABLE VIII
AVERAGE ABSOLUTE PERFORMANCE ON THE *CLS* TASK SPLIT

| **Dim** ($d_I$) | 5 | 10 | 50 | 100 | 250 |
|---|---|---|---|---|---|
| *Multi-task Subspace Finding* | | | | | |
| $\mathcal{T}_{\text{train}}$ | **50.0** | 48.0 | 49.5 | 48.7 | 46.0 |
| *Intrinsic Subspace Tuning* | | | | | |
| $\mathcal{T}_{\text{train}}^{\text{same}}$ | 35.2 | 31.9 | **51.0** | 49.1 | 47.9 |
| $\mathcal{T}_{\text{train}}^{\text{diff}}$ | 34.3 | 31.9 | **49.7** | 46.2 | 48.6 |
| $\mathcal{T}_{\text{test}}^{\text{in}}$ | 21.0 | 24.5 | 32.7 | **38.1** | 35.0 |
| $\mathcal{T}_{\text{test}}^{\text{out}}$ | 0.7 | 1.0 | 2.3 | 4.6 | **7.4** |

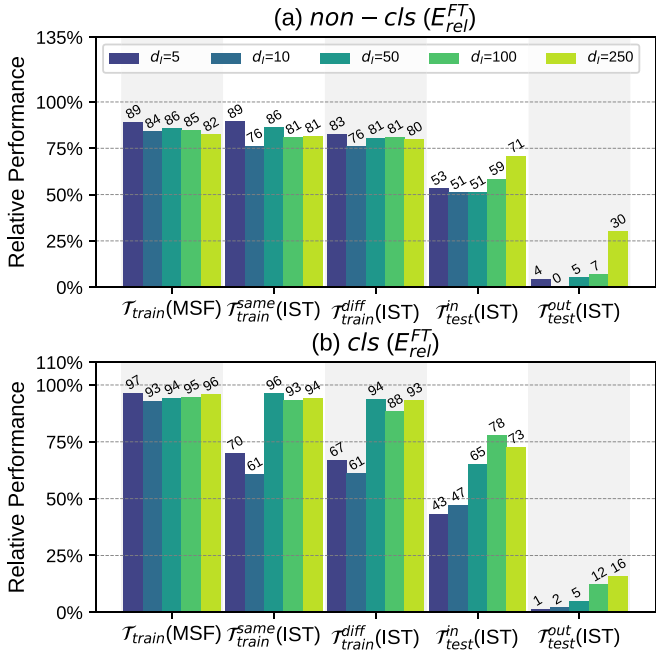


Fig. 9. Relative performance of IPTat different dimension $d_I$ on *non-cls* and *cls* splits, compared with fine-tuning. .

where we can see the general conclusions are consistent with our analyses in Section IV-B.

### C. BART$_{\text{LARGE}}$ Performance

All the experiments in the main paper are conducted with BART$_{\text{BASE}}$ model [58], which is also the main evaluated model of our adopted evaluation platform *CrossFit* [47]. To see whether the conclusions will also hold for larger models, we take a prior

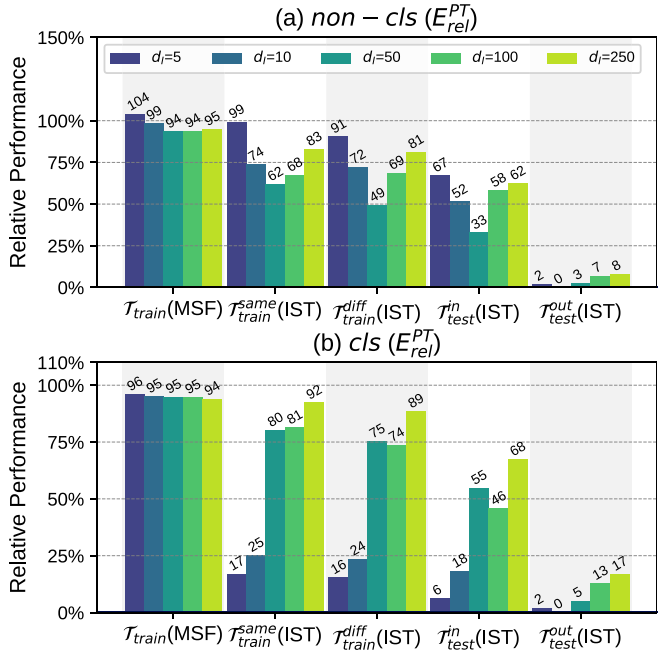


Fig. 10. Relative performance of IPTwith BART$_{\text{LARGE}}$ at different dimension $d_I$ on *non-cls* and *cls* splits, compared with prompt tuning. .

trial by conducting experiments on BART$_{\text{LARGE}}$. As the results in Fig. 10 suggest, the overall conclusions are consistent with those of BART$_{\text{BASE}}$ that non-trivial performance can be recovered in the found subspaces. However, when $d_I$ is extremely low ($5 \sim 10$), the performance is obviously worse than the cases of BART$_{\text{BASE}}$, especially on the *cls* split. This phenomenon may come from the difficulty of finding intrinsic task subspaces for larger PLMs. Considering that [8] indicate that larger PLMs generally have smaller intrinsic dimensions, we encourage future work to investigate how to better find the intrinsic task subspace for PLMs with larger sizes at small intrinsic dimensions. Since our main focus is to demonstrate the existence of the universal intrinsic task subspace and advance further research explorations, in this paper, we do not experiment on other kinds of PLMs or PLMs with extremely large sizes.

### C. Data Availability

Datasets used in this study are freely available at https://github.com/INK-USC/CrossFit and https://huggingface.co/datasets.

### C. Code Availability

The source code of this study is publicly available on GitHub at https://github.com/thunlp/Intrinsic-Prompt-Tuning.

### C. Contributions

The contributions of all authors are listed as follows: X.W., Y.Q., and Y.S. initiated the idea of this paper. Y.Q., J.Y., and W.C. conducted the experiments. Y.Q. and X.W. organized the research and drafted this paper. Y.Q., X.W., Y.L., and N.D. revised

and proofread the whole paper. Y.L., Z.L., J.L., L.H., P.L., M.S., and J.Z. advised the project and provide important suggestions to this paper. All authors participated in the discussion.

### C. Competing Interests

The authors declare no competing interests.

## REFERENCES


[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Computat. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available:https://www.cs.ubc.ca/amuham01/LING530/papers/radford2018improving.pdf

[3] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019. [Online]. Available: https://arxiv.org/abs/1910.10683

[4] X. Han et al., "Pre-trained models: Past, present and future," 2021. [Online]. Available: https://arxiv.org/abs/2106.07139

[5] B. Min et al., "Recent advances in natural language processing via large pre-trained language models: A survey," 2021. [Online]. Available: https://arxiv.org/pdf/2111.01243.pdf

[6] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," 2021. [Online]. Available: https://arxiv.org/abs/2104.08691

[7] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799. [Online]. Available: http://proceedings.mlr.press/v97/houlsby19a/houlsby19a.pdf

[8] A. Aghajanyan, S. Gupta, and L. Zettlemoyer, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.* 2021, pp. 7319–7328. [Online]. Available: https://aclanthology.org/2021.acl-long.568

[9] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353

[10] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst. 33: Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[11] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics: Main Volume*, 2021, pp. 255–269. [Online]. Available: https://aclanthology.org/2021.eacl-main.20

[12] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 2339–2352. [Online]. Available: https://aclanthology.org/2021.naacl-main.185

[13] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 423–438, 2020. [Online]. Available: https://aclanthology.org/2020.tacl-1.28

[14] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "AutoPrompt: Eliciting knowledge from language models with automatically generated prompts," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 4222–4235. [Online]. Available: https://aclanthology.org/2020.emnlp-main.346

[15] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 3816–3830. [Online]. Available: https://aclanthology.org/2021.acl-long.295

[16] K. Hambardzumyan, H. Khachatrian, and J. May, "WARP: Word-level adversarial ReProgramming," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 4921–4933. [Online]. Available: https://aclanthology.org/2021.acl-long.381

[17] Z. Zhong, D. Friedman, and D. Chen, "Factual probing is [MASK]: Learning vs learning to recall," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 5017–5033. [Online]. Available: https://aclanthology.org/2021.naacl-main.398

[18] G. Qin and J. Eisner, "Learning how to ask: Querying LMs with mixtures of soft prompts," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 5203–5212. [Online]. Available: https://aclanthology.org/2021.naacl-main.410

[19] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer, "Spot: Better frozen model adaptation through soft prompt transfer," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 5039–5059.

[20] F. Ma et al., "Xprompt: Exploring the extreme of prompt tuning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 11033–11047.

[21] Y. Gu, X. Han, Z. Liu, and M. Huang, "PPT: Pre-trained prompt tuning for few-shot learning," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 8410–8423.

[22] Y. Su et al., "On transferability of prompt tuning for natural language processing," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2022, pp. 3949–3969.

[23] J. Wang et al., "Towards unified prompt tuning for few-shot text classification," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 524–536.

[24] H. Zhang, X. Zhang, H. Huang, and L. Yu, "Prompt-based meta-learning for few-shot text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 1342–1357.

[25] X. Guo, B. Li, and H. Yu, "Improving the sample efficiency of prompt tuning with domain adaptation," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 3523–3537.

[26] H. Liu et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1950–1965.

[27] Y. Zhang, H. Fei, D. Li, and P. Li, "Promptgen: Automatically generate prompts using generative models," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 30–37.

[28] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the intrinsic dimension of objective landscapes," in *Proc. 6th Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryup8-WCW

[29] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein, "The intrinsic dimension of images and its impact on learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[30] S. Gong, V. N. Boddeti, and A. K. Jain, "On the intrinsic dimensionality of image representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3987–3996.

[31] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan, "Intrinsic dimension of data representations in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[32] T. Birdal, A. Lou, L. J. Guibas, and U. Simsekli, "Intrinsic dimension, persistent homology and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 6776–6789.

[33] F. Camastra and A. Staiano, "Intrinsic dimension estimation: Advances and open problems," *Inf. Sci.*, vol. 328, pp. 26–41, 2016.

[34] F. Camastra and A. Vinciarelli, "Estimating the intrinsic dimension of data with a fractal-based method," *IEEE Trans. Pattern Anal. Mach. Intell., vol.* 24, no. 10, pp. 1404–1407, Oct. 2002.

[35] P. Tempczyk, R. Michaluk, L. Garncarek, P. Spurek, J. Tabor, and A. Golinski, "LIDL: Local intrinsic dimension estimation using approximate likelihood," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 21205–21231.

[36] W. Zhu, Q. Qiu, J. Huang, R. Calderbank, G. Sapiro, and I. Daubechies, "LDMNet: Low dimensional manifold regularized neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2743–2751.

[37] X. He et al., "NAS-LID: Efficient neural architecture search with local intrinsic dimension," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 7839–7847.

[38] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, pp. 41–75, 1997.

[39] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[40] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple NLP tasks," 2016, *arXiv:1611.01587.*

[41] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," 2018, *arXiv:1806.08730*.

[42] A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta, "Muppet: Massive multi-task representations with pre-finetuning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 5799–5811. [Online]. Available: https://aclanthology.org/2021.emnlp-main.468

[43] D. Khashabi et al., "UnifiedQA: Crossing format boundaries with a single QA system," 2020, *arXiv:2005.00700*.

[44] R. Zhong, K. Lee, Z. Zhang, and D. Klein, "Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections," 2021, *arXiv:2104.04670*.

[45] S. Wang, H. Fang, M. Khabsa, H. Mao, and H. Ma, "Entailment as few-shot learner," 2021, *arXiv:2104.14690*.

[46] L. Ouyang et al., "Training language models to follow instructions with human feedback," 2022, *arXiv:2203.02155*.

[47] Q. Ye, B. Y. Lin, and X. Ren, "CrossFit: A few-shot learning challenge for cross-task generalization in NLP," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 7163–7189. [Online]. Available: https://aclanthology.org/2021.emnlp-main.572

[48] D. Hendrycks et al., "Measuring massive multitask language understanding," 2020, *arXiv:2009.03300*.

[49] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," 2021, *arXiv:2110.08207*.

[50] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, "Cross-task generalization via natural language crowdsourcing instructions," 2021, *arXiv:2104.08773*.

[51] A. Srivastava et al., "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," 2022, *arXiv:2206.04615*.

[52] Y. Wang et al., "Super-naturalinstructions: Generalization via declarative instructions on 1600 NLP tasks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 5085–5109.

[53] J. Wei et al., "Finetuned language models are zero-shot learners," 2021, *arXiv:2109.01652*.

[54] H. W. Chung et al., "Scaling instruction-finetuned language models," 2022, *arXiv:2210.11416*.

[55] J. Schulman et al., "ChatGPT: Optimizing language models for dialogue," 2022.

[56] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[57] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 10–21. [Online]. Available: https://aclanthology.org/K16-1002

[58] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[59] D. Khashabi et al., "UNIFIEDQA: Crossing format boundaries with a single QA system," in *Proc. Findings Assoc. Comput. Linguistics*, 2020, pp. 1896–1907. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.171

[60] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[61] N. Ding et al., "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models," 2022. [Online]. Available: https://arxiv.org/pdf/2203.06904.pdf

[62] Q. Le et al., "Fastfood-approximating kernel expansions in loglinear time," in *Proc. Int. Conf. Mach. Learn.*, 2013, p. 8. [Online]. Available: http://proceedings.mlr.press/v28/le13.pdf

[63] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.

[64] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[65] J. Kaplan et al., "Scaling laws for neural language models," 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[66] J. Wei et al., "Finetuned language models are zero-shot learners," 2021. [Online]. Available: https://arxiv.org/abs/2109.01652

[67] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," 2021, *arXiv:2110.04366*.

[68] J. Yi et al., "Different tunes played with equal skill: Exploring a unified optimization subspace for parameter-efficient tuning," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 3348–3366. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.244