Hit the Nail on the Head: Parameter-Efficient Multi-task Tuning via Human Language Intervention

Wenxuan Lu, Songhao Jiang, Yijing Wang, Tianning Zang
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China[1]
School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China [2]
`{luwenxuan, jiangsonghao, wangyijing, zangtianning}@iie.ac.cn`

Abstract

Parameter-Efficient Fine-Tuning (PEFT) on small Pre-trained Language Models (PLMs) has emerged as a promising approach to enhance their multi-tasking capabilities. Prevalent methods simultaneously train additional modules (i.e., one task-shared module and multiple task-specific modules) for adapting PLMs to downstream tasks. However, their adaptability to new tasks is constrained, as the task-specific modules independently adapt to each task, overlooking the potential for knowledge transfer across tasks. In this paper, we propose a novel multi-task learning framework, Inspirational Pointer (IP), that enables the transfer of prior knowledge across tasks through human language intervention. Specifically, we attach task descriptions to the input samples, which are then mapped to corresponding task embeddings. Based on those embeddings, we adapt PLMs for downstream tasks. Similar tasks share akin descriptions, allowing new task samples close to similar trained tasks in the task embedding space, hitting the memory about trained tasks of the model. Our experiments on the T5 model demonstrate performance improvements of our method in multi-task learning and few-shot transfer learning. Further, we implemented the IP in decoder-only models including GPT2 and large language models (LLMs), and the results show that IP enhances the capabilities of decoder-only models.

1 Introduction

The goal of multi-task learning is to equip a single model with the capability to address multiple tasks simultaneously. Recent large language models (LLMs) have demonstrated outstanding performance in the field. However, deploying LLMs necessitates significant computational resources, which is inefficient for scenarios requiring rapid processing of massive amounts of data. Consequently, enhancing the multitasking abilities of

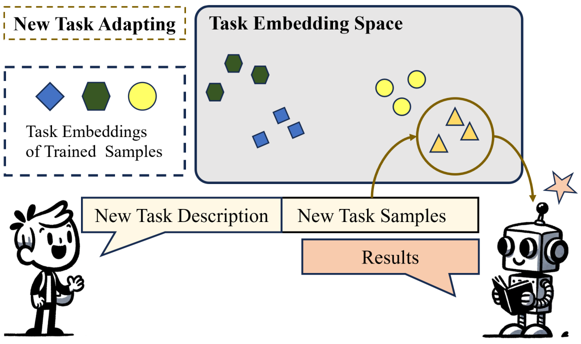


Figure 1: Inspirational Pointer (IP) initially modifies all input samples by appending task descriptions. Then, it generates the alterations to the hidden states of PLM based on the task embeddings of these samples.

smaller Pre-trained Language Models (PLMs), especially those operating on one consumer-grade GPU, is essential. Recent studies indicate that Parameter-Efficient Fine-Tuning (PEFT) on small PLMs significantly improves their multitasking capabilities [Asai et al., 2022; Mahabadi et al., 2021]. The PEFT method freezes all parameters of PLM and trains additional compact modules per task, allowing PLM to adapt to multiple downstream tasks [Houlsby et al., 2019; Li and Liang, 2021].

Recent developments in PEFT on samll PLMs for multi-task learning focus on a crucial challenge: effectively capturing both task-specific and shared information across tasks. A potential solution is the integration of PEFT modules, typically involving two training phases [Wang et al., 2023; Asai et al., 2022]. Initially, individual task-specific PEFT modules are trained separately. Subsequently, an integration module is trained to consolidate those task-specific modules for downstream tasks. Prior works show the effectiveness of this approach in multi-task learning and few-shot transfer learning. However, the additional integration training consumes more resources, which limits its practicality.

An alternative solution provides a more streamlined training process, which is called the hypernetwork-based method [Zhao et al., 2023; Ivison and Peters, 2022; Mahabadi et al., 2021]. It optimizes a hypernetwork and multiple task-specific embeddings from multi-task tuning. Guided by distinct task embeddings, this hypernetwork generates parameters for various task-specific adapters, a kind of PEFT module. During training, the hypernetwork captures shared information across tasks, while each task embedding learns the task-specific information. This one-time training method achieves a better balance between resource consumption and performance. However, this method often shows limited transfer learning effectiveness, particularly in few-shot scenarios. This is due to two main reasons: (1) Adapting to a new task requires training a task-specific embedding independently, without leveraging previously trained task embeddings. (2) The limited number of samples fails to provide enough task-specific information for task embedding.

This paper primarily aims to improve the few-shot transfer learning in one-time training methods. We introduce the Inspirational Pointer (IP), a multi-task learning framework that leverages human language intervention as a knowledge transfer bridge between various tasks. As shown in Figure 1, it first incorporates task descriptions into input samples and then projects the modified data into the task embedding space. Finally, these task embeddings are transformed into modification vectors, which are used to adapt the hidden states of PLMs for downstream tasks. Due to similar tasks having comparable descriptions, this approach enables new samples to align closely with trained tasks in the task embedding space, thereby leveraging the existing knowledge of the model. Additionally, task descriptions also offer task-specific prior knowledge to PLMs [Weller et al., 2020]. Furthermore, we **moved away** from the 'hypernetwork-adapter' structure and explored more parameter-efficient methods. Specifically, we infuse task-specific knowledge into the PLM by directly performing a Hadamard product between the modification vectors and the hidden states of the PLM. Previous work [He et al., 2021] has demonstrated that the Hadamard product can convey more information with fewer parameters, which leads to the parameter efficiency of IP (Shown in Figure 2).

Following previous works, we evaluated the performance of IP on T5 model [Raffel et al., 2019] with 13 NLP datasets. The results show that IP re-

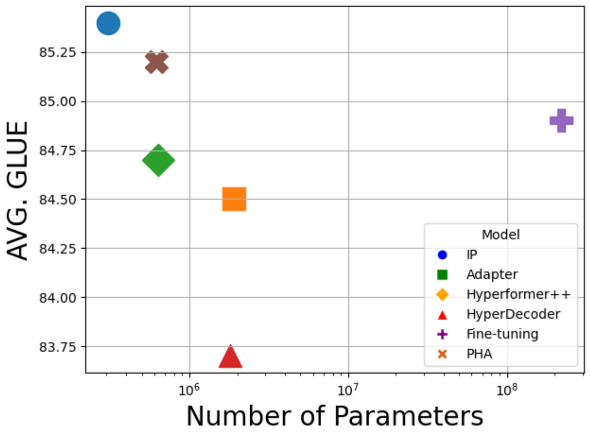


Figure 2: Parameter efficiency on GLUE. The IP demonstrates exceptional precision (depicted on the y-axis) while requiring minimal parameter adjustments for each specific task (shown on the x-axis).

quires only a small number of samples to adapt to new tasks, and it also has considerable advantages in traditional multi-task learning performance. Further, we implemented the IP in decoder-only models, and the results showed that IP enhances their capabilities (in Appendix A.5).

Contributions can be listed as follows:

- This paper finds the limitations of existing methods in adapting to new tasks: the inability to transfer knowledge from trained tasks to new ones.
- We propose a multi-task learning framework Inspirational Pointer (IP), that facilitates knowledge transfer between tasks through human language intervention.
- We evaluated IP performance under various settings, including different tasks, model sizes, and architectures (in appendix A.5), demonstrating its wide applicability.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) on Pre-trained Language Models (PLMs) has become an active research area. This approach trains additional small-scale parameter modules to enhance the adaptability of PLMs to downstream tasks. Some PEFT methods inject trainable modules into the architecture of PLMs. Adapter [Houlsby et al., 2019] and its derivative works [J. et al., 2021; Karimi Mahabadi et al., 2021] insert the trainable modules between each transformer layer of

PLMs. Another choice is prompt tuning [Lester et al., 2021], which attaches the learnable modules (i.e., soft prompts) to the input. The prefix-tuning method [Li and Liang, 2021] tries to optimize the soft prompts for natural language generation tasks. PEFT requires only the training of compact modules for each downstream task, eliminating the need to adjust the parameters of the entire PLM. This has brought significant advancements in the field of multi-task learning.

2.2 Multi-task Learning with PEFT

In implementing PEFT methods in multi-task scenarios, a key challenge is efficiently harnessing information unique to each task while leveraging common elements across tasks. A potential solution is the integration of PEFT modules, generally entailing a dual-phase training process. First, PEFT modules for individual tasks are trained independently. This is succeeded by the training of a unifying integration module that amalgamates these task-specific PEFT modules for downstream tasks. Previous integration methods were based on the attention modules to integrate PEFT modules (e.g., [Pfeiffer et al., 2020], prompts [Asai et al., 2022]) or the knowledge distillation for PEFT modules [Wang et al., 2023]. Yet, this extra step of training demands more resources, posing a constraint on its real-world applicability. An alternative solution is hypernetwork-based method [Zhao et al., 2023; Ivison and Peters, 2022; Mahabadi et al., 2021; Ye and Ren, 2021], which uses hypernetwork to generate parameters for various task-specific adapters via task embeddings. During training, the hypernetwork captures the cross-task information, and the task embeddings learn the specific information for each task. However, this approach has limitations in adapting to new tasks. Our approach mitigates this issue by using task descriptions.

3 Method

3.1 Overview

IP is a universal framework suitable for both encoder-decoder and decoder-only structured PLMs. The core concept of IP is to enable knowledge transfer between tasks via the human language Intervention. As shown in Figure 3, IP includes three main steps: (1)Sample modification: Combining the input sample x with a manual task description s to form the modified sample m, i.e., $m = s + x$. (2) Task embedding generation: The modified sample m is processed through a form module f, resulting in the task embedding t, i.e., $t = f(s + x)$. (3) Model hidden state modification: The task embedding t is utilized to modify the model's hidden states. For the encoder-decoder model M, such as T5, the task embedding form module f is the model's encoder m_1, and the hidden state of the model's decoder m_2 is modified. For the decoder-only model L, such as GPT2 or LLMs, the task embedding form module f is an auxiliary fixed encoder e, and the hidden states of L's each layer are modified. During the training process, the parameters of the PLM are fixed.

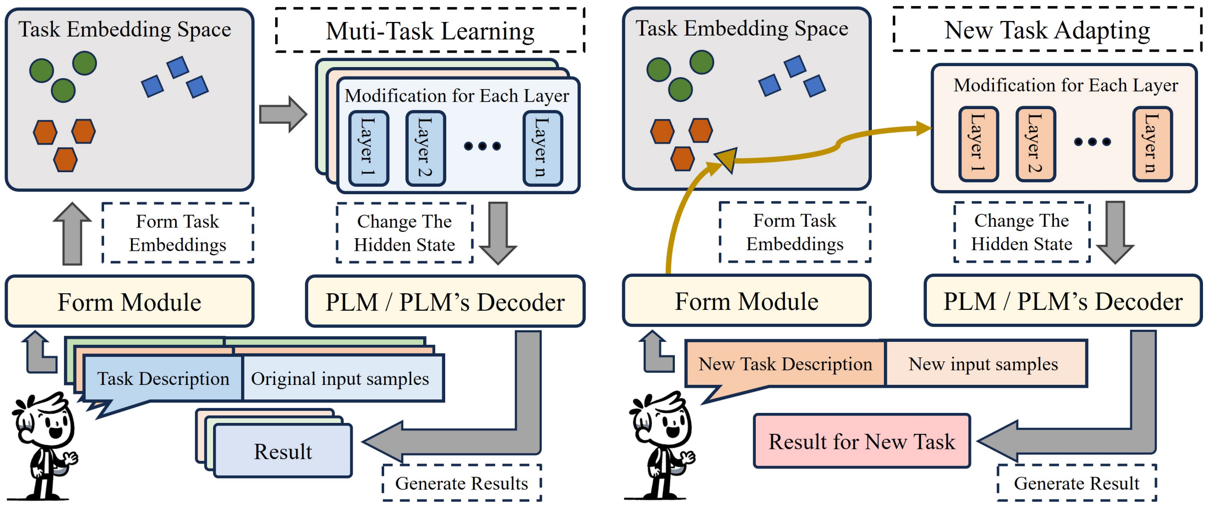
3.2 Details of Sample Modification

In this step, we manually construct and integrate task descriptions with input samples. The task descriptions consist of three parts: the dataset name, task-specific sentence patterns, and dataset-specific words. For task-specific sentence patterns, we have crafted fixed structures tailored to various tasks, such as text classification tasks (Tell me the _ of the _ :), similarity assessment tasks (Evaluate the _ similarity of _ :), and inference tasks (Determine the _ between the _ from _ :). For the dataset-specific words, we filled in the task-specific sentence patterns with dataset-specific words, such as ({sentiment}) for SST2. For example, we constructed the descriptive sentences for the Natural Language Inference (NLI) datasets SciTail and MNLI and the text classification dataset SST2, as follows:

- SciTail: Determine the relationship between the following statements from science materials:
- MNLI: Determine the relationship between the following sentences from a variety of genres:
- SST2: Tell me the sentiment of the film review:

Our task description construction adheres to human language characteristics, where similar tasks have similar descriptive patterns and unrelated tasks exhibit significant differences in their descriptions. We also tried various ways to construct task descriptions, which included automatic generation by GPT, and intuitive manual construction. Please refer to Appendix A.6 for ABLATION results on different ways for constructing task descriptions.

Each input sample will be augmented with a corresponding task description before being fed into the model. Due to the human language intervention, the model relates samples from a new task to previously trained task samples using human language as a bridge. Furthermore, the task descriptions contain task-level semantic information, providing the



(a) The process of IP in multi-task learning. (b) The process of IP in new task adapting.

Figure 3: Figures (a) and (b) illustrate the process of IP in multi-task learning and new task adaptation. During multi-task learning, a suitable task embedding space is established. In adapting to new tasks, influenced by the task descriptions, samples of the new task gravitate closer to similar trained tasks in the task embedding space, thereby enabling PLM to generate suitable results. For the encoder-decoder PLM, the task embedding form module is the encoder of PLM, and the hidden state of the model's decoder is modified. For the decoder-only PLM, the task embedding form module is an auxiliary encoder model, and the hidden states of the entire PLM are modified.

model with sufficient prior knowledge.

3.3 Task Embeddings Form module

After sample modification, we use the form module to construct a task embedding $\tau \in \mathbb{R}^t$, guiding the modification of the PLM's hidden states in the next step. For encoder-decoder models, the encoder acts as the form module. For decoder-only models, we use an additional parameter-fixed encoder model to construct the task embeddings. We collected the output vectors $\{V\} = \{v_1, v_2, \ldots, v_{n_1}\}, v_{n_1} \in \mathbb{R}^{d_1}$, from each layer's mean-pooled hidden state of the form module f, where d_1 is dimension of f and n_1 is the layer number of f. These vectors are combined via a linear summation layer and a projection network T to form a task embedding τ, where T is a neural network consisting of a feed-forward layer and a ReLU non-linearity. We describe this process as:

$$\tau = T(\sum_{i=1}^{n_1} w_i v_i) \tag{1}$$

where w_i are the weights of the linear composition.

Notably, the task embedding we construct is the "sample-level task embedding," meaning that every sample generates its distinct task embedding. Prior work [Ivison and Peters, 2022] demonstrates that sample-level task embeddings enable the model to leverage similarities between samples across datasets while avoiding potential interference within the same dataset. Our experiments revealed that the similarity scores between task embeddings become more reasonable after introducing samples with task descriptions. This suggests the transfer of knowledge across various tasks.

3.4 Hidden State Modification

In this step, we transform task embeddings into modifications of the PLM. For encoder-decoder PLMs, we modify the hidden states of its decoder. For decoder-only PLMs, we alter the hidden states of the entire model. Distinct from the previous work, our method forgoes the 'hypernetwork-adapter' architecture, establishing more efficient connections between PLM and task embeddings. Inspired by prior work [He et al., 2021], we reframe the hypernetwork-based method as the modification of the PLM's hidden states h under the guidance of task embeddings.

$$h \sim h + \Delta h; \quad \Delta h = A_\tau(h) \tag{2}$$

The main idea of the hypernetwork-based method is to ensure that the Δh contains task-specific information. To achieve this, the hypernetwork generates the parameters of adapters A_τ based on the task embeddings τ. Consequently, the task embedding τ can influence the Δh through A_τ.

Guided by such insights, we adopt a unique approach where task-specific information is effectively transmitted to the modification vector of the hidden state by computing the Hadamard product with the hidden state h. The reason for using the Hadamard product is that previous work [Hyeon-Woo et al., 2021] has demonstrated that, with the involvement of the Hadamard product, the model can transmit more information while reducing the number of parameters. Compared to the hypernetwork-based method, our method demonstrates superior parameter efficiency, as shown in Figure 2.

Initially, to generate specific modification vectors for the hidden state of each PLM layer and reduce the number of training parameters, we introduce the layer id embeddings inspired by previous work [Mahabadi et al., 2021]. The layer id embeddings are denoted as $\mathcal{L} = \{L_m\}_{m=1}^{n_2}$, where n_2 and m represents the layer number and the m-th layer of the PLM. Each layer id embedding is combined with the task embedding to generate the corresponding modification vector for the PLM layer's hidden states. Specifically, we combine the task embedding $\tau \in \mathbb{R}^t$ with the layer embedding $L_m \in \mathbb{R}^t$ into a mixed modification vector $I_m \in \mathbb{R}^t$ by feeding a concatenation of (τ, L_m) to a multi-layer neural network C, consisting of two feed-forward layers (i.e., projections $c_1 \in \mathbb{R}^{t\times t}$ and $c_2 \in \mathbb{R}^{t\times d_2}$, where d_2 is dimension of the PLM) and a ReLU non-linearity. Finally, we compute the Hadamard product of these modification vectors $\mathcal{I} = \{I_m\}_{m=1}^{n_2}$ with each hidden state, thus forming the final modification for the hidden state of PLM. This process can be described as follows:

$$\Delta h = \{h_m \circ I_m\}_{m=1}^{n_2}; \quad I_m = C(\tau, L_m) \tag{3}$$

, where h_m means the hidden state of the m-th PLM layer. Ultimately, the model's hidden state h will be updated to $h + \Delta h$.

3.5 Loss Function

The training method for IP is consistent in both multi-task learning and new task adaptation. We consider a general multi-task learning problem. Given a set of target tasks $\{D\} = \{D_1, D_2, \ldots, D_t\}$, where t is the total number of tasks and $\{D_i\} = \{x_i^n, y_i^n\}_{n=1}^{N_i}$ shows the training data of the $i-$th task with N_i samples. Also given a PLM M_θ with parameters θ. We focus on a multi-task setup, where a model is trained on multiple tasks simultaneously. Standard multi-task loss on the training set can be listed as follows:

$$L(\theta, \{D_\tau\}_{i=1}^{T}) = \sum_{\tau=1}^{T} \sum_{(x_i^\tau, y_i^\tau)\in D_\tau} w_\tau l(M_\theta(x_i^\tau), y_i^\tau) \tag{4}$$

where l represents the loss function for each task, typically the cross-entropy loss, and w_τ is the weight for the t-th task.

In our approach, we incorporate the task descriptions into the input samples and transform the encoder output vectors into task embeddings. Guided by task embeddings, we adjust the hidden states of the PLM to adapt to downstream tasks. Let S_τ denote the description of the t-th task and χ represent the PLM with IP. The final loss formula can be articulated:

$$\begin{aligned} &L(\theta, \{S_\tau\}_{i=1}^{T}, \{D_\tau\}_{i=1}^{T}) \\ &= \sum_{\tau=1}^{T} \sum_{\substack{s_i^\tau \in S_\tau \\ (x_i^\tau, y_i^\tau)\in D_\tau}} w_\tau l(\chi_\theta(x_i^\tau, s_i^\tau), y_i^\tau) \end{aligned} \tag{5}$$

3.6 Parameter-efficiency

For IP, we introduce a one-layer neural network T, the weights of the linear composition W, a set of learnable layer id embeddings $\mathcal{L} = \{L_m\}_{m=1}^{n_2}$, and a two-layer neural network C, resulting in $(d_1 \times t) + n_1 + (t \times n_2) + (t \times t + t \times d_2)$ parameters, where n is the number of PLMs layers, d is the PLM dimension and t is the task embedding dimension we set. Therefore, during the training process, a total of $(t \times (d_1 + d_2 + t + n_2) + n_1)$parameters need to be adjusted, which is significantly less than the number of parameters required by traditional fine-tuning and Adapter methods. We list and compare various methods in terms of the number of the trainable parameters in Table 1.

4 Experiment

4.1 Model Choice

Our primary PLM is the T5-Base (220M) [Raffel et al., 2019]. That choice is based on the conventions of prior multi-task works [Wang et al., 2023; Zhao et al., 2023; Asai et al., 2022] and the need for fair comparison. Moreover, enhancing the multi-task capabilities of small-scale model T5, which can operate on one consumer-grade GPU, holds great practical value given the ubiquitous issue of limited computational resources. Additionally, we incorporate T5-Small (60M) and T5-Large (770M)

Method	Tunable	GLUE									SuperGLUE				
	Params	CoLA	SST-2	STS-B	MRPC	QQP	MNLI	QNLI	RTE	Avg	BoolQ	WiC	CB	WSC	Avg
Fine-tuning	220M	61.8	**94.6**	89.7	90.2	**91.6**	86.8	93.0	71.9	84.9	81.1	**70.2**	85.7	59.6	74.2
Adapter	1.9M	64.0	93.2	**90.7**	85.3	90.2	86.5	93.2	71.9	84.5	**82.5**	67.1	85.7	**67.3**	75.7
Fine-tuning-m	220M	54.9	92.5	88.8	**90.2**	91.1	85.7	92.0	75.4	83.8	78.5	69.5	85.2	66.7	75.0
Adapter-m	1.9M	61.5	93.0	89.9	90.2	90.5	86.3	93.2	70.3	84.4	78.4	67.3	85.2	64.7	73.9
Prompt-tuning	76.8k$*n$	10.6	90.9	89.5	68.1	89.7	81.3	92.8	54.7	72.2	61.7	48.9	67.9	51.9	57.6
ATTEMPT	96k$*n$	57.4	93.2	89.7	85.7	90.3	84.3	93.0	73.4	83.4	78.8	66.8	78.6	53.8	69.5
MPT	76.8k$*n$	**62.9**	93.1	90.1	89.6	89.9	85.5	93.2	77.8	85.2	79.4	69.1	79.3	67.5	73.8
PHA	616k	60.6	94.0	88.2	89.2	90.3	86.3	93.4	**80.4**	85.3	81.3	65.3	94.1	63.2	75.9
Hyperformer++	638K	59.0	93.7	90.3	88.6	89.9	85.0	93.3	77.5	84.7	75.8	68.9	81.5	52.9	69.8
HyperDecoder	1.8M	55.9	94.0	90.5	87.7	90.5	86.0	93.4	71.7	83.7	77.8	66.0	92.6	66.7	75.8
Inspirational Pointer	**347K**	60.2	93.9	90.3	88.7	90.8	**86.8**	**93.6**	79.6	**85.5**	79.2	67.1	**94.6**	66.5	**76.8**

Table 1: The result of the multi-task learning. T5-base serves as the PLM backbone of all methods. We also report **Tunable Params**, which represent the number of parameters that need to be fine-tuned for each task. The best result on each block is in **bold**. It is noteworthy that due to prompt-tuning, ATTEMPT, and MPT, which require learning prompt parameters separately for each task, their parameter number has to be multiplied by n, where n represents the number of training tasks. Fine-tuning and adapter denote those methods individually adapt to each task with no parameter sharing, while Fine-tuning/adapter-m denote these methods adapt to multiple tasks simultaneously.

models to investigate the influence of model scale on IP performance A.3. To demonstrate the generality of the IP, we have also applied it to decoder-only models in appendix A.5. The results showed that IP can also improve the performance of these models, consistent with the trend of T5 experiments.

4.2 Datasets

Following previous works in multi-task learning, we evaluate IP using 8 datasets from the GLUE benchmark [Wang et al., 2018] and 4 datasets from the SuperGLUE benchmark [Wang et al., 2019]. For the few-shot transfer learning test, we train IP on the GLUE tasks and then evaluate our method using three datasets: CB, BoolQ from SuperGLUE, and an additional dataset SciTail [Khot et al., 2022]. More details are seen in AA.1

4.3 Implementation Details

Our multi-task adaptation experiment involved conducting multi-task learning across 8 datasets from the GLUE benchmark and 4 datasets from the SuperGLUE benchmark. Following prior work [Zhang et al., 2020], we use the validation set as the testing set in the absence of a dedicated testing set. In our few-shot adaptation experiments, we select k = 4, 16, and 32 examples from the training dataset, while utilizing the full test set for evaluation. The task embedding dimension size is 200. T5 model is finetuned using the AdamW optimizer [Loshchilov and Hutter, 2017], employing a 3e-4 learning rate with linear decay and a warmup of 500 steps. More details are seen in A.2.

4.4 Baselines

To evaluate the effectiveness of our method, we compare IP with the following baselines: (1) Full fine-tuning (FT). (2) Adapters [Houlsby et al., 2019]. (3) Prompt tuning (PT) [Lester et al., 2021], the prompt tuning introduces task-specific embeddings to the input layer, initializing them by randomly sampled top vocabularies. (4) Prompt transfer method, which is the advanced method for PEFT module integration. we select ATTEMPT [Asai et al., 2022], MPT [Wang et al., 2023], which adapts to the target tasks using shared prompts, which are derived by distilling knowledge from the source tasks. (5) Hypernetwork-based method, we select state-of-the-art models: Hyperformer [Mahabadi et al., 2021], Hyperdecoder [Ivison and Peters, 2022] and PHA [Zhao et al., 2023], which use the hypernetwork to generate the parameters of adapters and integrate them into PLM layers.

4.5 Result

4.5.1 Multi-task Learning

As shown in Table 1, the results demonstrate that IP surpasses all other methods in terms of enhancing performance, while also efficiently managing parameters. Note that ATTEMPT and MPT require iterative training. They rely on pre-training prompts to retain knowledge from source tasks, and this

k-shot		FT	Adapter	PT	HF	ATP	PHA	MPT	HD	IP
4	BoolQ	50.5	53.4	61.6	48.0	61.8	66.7	62.2	54.4	**67.8**
	SciTail	79.6	79.5	57.7	82.0	80.2	82.5	80.2	75.4	**83.1**
	CB	57.7	51.1	53.5	60.7	**82.1**	76.5	73.6	69.1	73.2
16	BoolQ	56.5	51.4	61.9	50.6	60.0	**71.3**	63.3	64.6	69.3
	SciTail	80.0	83.2	60.8	71.9	79.5	86.7	87.3	85.4	**87.9**
	CB	77.0	74.8	63.5	64.3	78.5	**79.6**	78.6	75.3	77.8
32	BoolQ	58.4	54.5	61.7	58.3	65.3	71.3	68.9	68.3	**72.2**
	SciTail	81.9	85.0	60.2	85.8	80.2	**88.6**	86.3	85.1	84.2
	CB	80.0	74.8	67.8	81.4	85.7	82.7	82.1	79.6	**86.1**

Table 2: Result of few-shot transfer learning with shot counts k = 4, 16, 32, detailing the accuracy across all tasks based on 10 random seeds. The best result on each block is in **bold**. All models are trained using the GLUE tasks, with the T5-Base model serving as the backbone.

knowledge is transferred to target tasks through the training of an integration module. Therefore, the computation of their trainable parameters should include the training of pre-training prompts for multiple source tasks.

Compared to the traditional adapter method, the IP approach utilizes only one-sixth of the trainable parameters while achieving approximately a 1% improvement in performance. In comparison with vanilla fine-tuning methods, our approach significantly reduces the number of training parameters from 220m to 347k while maintaining comparable performance. When compared to the state-of-the-art models of hypernetwork-based methods Hyperformer++, Hyperdecoder, and PHA, IP shows superior accuracy improvements on both the GLUE and SuperGLUE benchmarks, while utilizing the lower number of trainable parameters. Similar observations were made when comparing IP with prompt transfer methods, which require iterative training.

4.5.2 Few-shot Transfer Learning

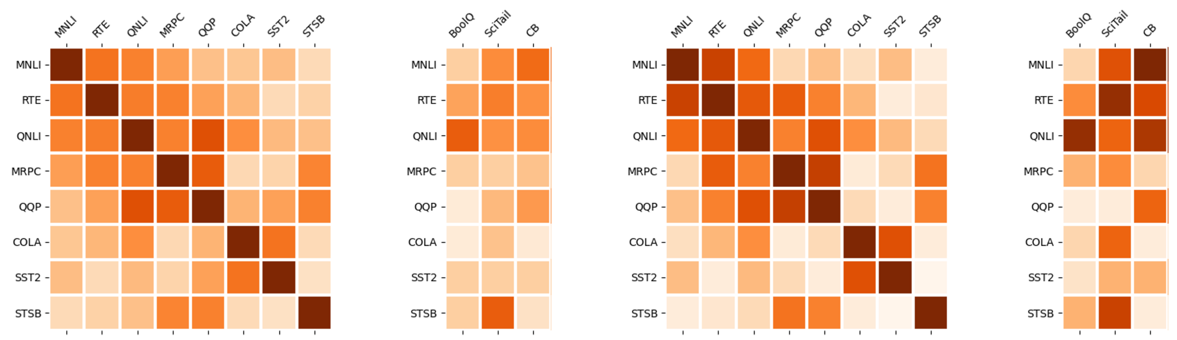
In our research, we assess the effectiveness of our proposed approach in adapting to new tasks. Followed by [Wang et al., 2023], we conduct few-shot transfer experiments on datasets BoolQ, CB, and SciTail. Initially, we randomly selected k=4, 16, and 32 samples from three datasets to construct few-shot training sets. Subsequently, all models were first trained on the Glue tasks, followed by adaptation training for new tasks on the few-shot datasets. Finally, the adapted models were evaluated on the complete test set to obtain the final results. Our comparison includes several established baselines such as Fine-tuning, Adapter, Prompt tuning, Hyperformer, ATTEMPT, HyperDecoder, MPT, and PHA. The findings presented in Table 2 stem from training an 8-task adaptation model for GLUE, followed by fine-tuning using a limited number of samples from BoolQ, CB, and SciTail.

Table 2 summarizes the results of the few-shot transfer learning experiments. Our method demonstrates significant improvements over the traditional adapter approach across various few-shot datasets. Compared to hypernetwork-based methods, our approach benefits from effectively facilitating knowledge transfer between tasks, and IP achieves significant improvements. Additionally, despite the multiple training iterations involved in prompt transfer methods, the performance of IP is on par with them. These results indicate that despite the limited number of training samples, our method effectively adapts to new tasks. Comparing with the prompt transfer approach, the IP method demonstrated superior performance, despite the prompt transfer method requiring multiple training iterations. After a comprehensive comparative analysis, we found that the IP method achieved superior performance with fewer trainable parameters.

4.6 Ablation Study

4.6.1 Components Remove

We conduct an ablation study using the GLUE benchmark to assess the impact and efficacy of the introduced components. The task description and the Hadamard Product are removed independently for this purpose. Specifically, to assess the impact of task descriptions on the model, we eliminated modifications to the input samples. Then, instead of using the Hadamard product, we evaluated its impact on the model by employing the traditional hypernetwork approach to generate adapter parameters for task embedding processing. As shown in Table 3, we observed a significant decrease in the final performance of the model when the task



(a) Training model without task description

(b) Training model with task description

Figure 4: Visualization of the similarity scores, which are calculated by task embeddings. Darker colors indicate higher scores.

descriptions were removed. This indicates that the original samples, when used as inputs, had limited capacity for information transfer between tasks. Our method addresses this shortcoming by directly incorporating task-specific prior knowledge into the model through the task descriptions. Moreover, as similar tasks have similar descriptions, this facilitates knowledge transfer between tasks. Additionally, we noted that the performance of methods modifying the pre-trained language models (PLM) with task embedding, using either the Hadamard product or hypernetworks, was comparable. Given that our method requires fewer parameters than the hypernetwork-based approach, this demonstrates the advantages of our method in terms of parameter efficiency.

	Task Description	GLUE Avg
Hadamard Product	✓	85.5
	×	84.9
Hypernetwork	✓	85.2
	×	84.3

Table 3: Result of ablation study on task description and Hadamard Product. "GLUE Avg" means average performance in GLUE.

4.6.2 Effect of Task Description

We conducted a qualitative analysis of IP to study whether cross-task knowledge is indeed captured by task embeddings after adding task descriptions to the samples. We measured the average task embeddings produced by the model for eight datasets in the GLUE benchmark and calculated their similarities. The same procedure was applied to a few-shot transfer learning dataset. Specifically, we first fully trained IP on the GLUE benchmark, then randomly sampled 200 input examples from the test sets of various tasks, extracted the task embedding generated by the model for each task, and calculated their averages. Finally, we recorded the similarities between the average task embeddings. We experiment on both scenarios, IP with and without task descriptions, to reveal the important role of task descriptions.

The results are shown in Figure 4. Panel B indicates that under the influence of task descriptions, tasks that are similar (e.g., MNLI and RTE) have higher task embedding similarities. This demonstrates that IP effectively clusters similar tasks together. In terms of adapting to new tasks, we observed that IP effectively links trained tasks with similar tasks, as evidenced by the relationships between CB and RTE, MNLI. Comparing Panels A and B, t is observed that, guided by task descriptions, embeddings of similar tasks are more concentrated. Moreover, this alignment results in new tasks deriving enhanced benefits from previously trained tasks.

5 Conclusion

This paper introduces the Inspirational Pointer (IP), a multi-task learning framework. IP modifies input samples by attaching task descriptions, then influences the PLM hidden states based on the ask embeddings of these samples. The experimental results show that our method achieves superior performance, applicable in multi-task and few-shot transfer learning scenarios, and IP with various sizes PLMs also perform well. Finally, the ablation study highlights the significance of task descriptions. For future work, we aim to explore more effective methods for constructing task descriptions.

Limitation

This paper proposes a multi-task learning framework applicable to various models. Although it performs well across multiple tasks and models, it still has limitations when applying it to decoder-only models: an additional form module is required. Although the parameters of the form module are fixed, thus maintaining parameter efficiency, the additional module reduces the overall computational efficiency. In future work, we will attempt to directly extract certain states from the decoder-only model itself to construct task embeddings, thereby improving computational efficiency.

References

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. *The PASCAL Recognising Textual Entailment Challenge*, page 177–190.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *Cornell University - arXiv,Cornell University - arXiv*.

WilliamB. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases.

Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2021. Kronecker decomposition for gpt compression. *arXiv preprint arXiv:2110.08152*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzębski, Bruna Morrone, Quentinde Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *International Conference on Machine Learning,International Conference on Machine Learning*.

Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. 2021. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098*.

Hamish Ivison and Matthew E Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task nlp. *arXiv preprint arXiv:2203.08304*.

HuEdward J., Yulong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv: Computation and Language,arXiv: Computation and Language*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2022. Scitail: A textual entailment dataset from science question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

HectorJ. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. *Principles of Knowledge Representation and Reasoning,Principles of Knowledge Representation and Reasoning*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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)*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *Learning,Learning*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*.

Marie-Catherinede Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung,Proceedings of Sinn und Bedeutung*.

George Mihaila. 2021. Gpt2 for text classification using hugging face. `https://gmihaila.github.io/tutorial_notebooks/gpt2_finetune_classification/`.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

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

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

Richard Socher, Alex Perelygin, JeanY. Wu, J.C.-I. Chuang, ChristopherD. Manning, AndrewY. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *Empirical Methods in Natural Language Processing,Empirical Methods in Natural Language Processing*.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. *Preprint*, arXiv:2305.08377.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and SamuelR. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv: Computation and Language,arXiv: Computation and Language*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*.

Alex Warstadt, Amanpreet Singh, and SamuelR. Bowman. 2018. Neural network acceptability judgments. *arXiv: Computation and Language,arXiv: Computation and Language*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E Peters. 2020. Learning from task descriptions. *arXiv preprint arXiv:2011.08115*.

Adina Williams, Nikita Nangia, and SamuelR. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *Cornell University - arXiv,Cornell University - arXiv*.

Qinyuan Ye and Xiang Ren. 2021. Learning to generate task-specific adapters from task description. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 646–653.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, KilianQ. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv: Computation and Language,arXiv: Computation and Language*.

Hao Zhao, Jie Fu, and Zhaofeng He. 2023. Prototype-based HyperAdapter for sample-efficient multi-task tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4603–4615, Singapore. Association for Computational Linguistics.

A Appendix

A.1 Details in Dataset

we evaluate IP using 8 datasets from the GLUE benchmark [Wang et al., 2018] and 4 datasets from the SuperGLUE benchmark [Wang et al., 2019]. Specifically, those benchmarks comprise a range of text classification tasks, including CoLA [Warstadt et al., 2018] for testing sentence acceptability, SST-2 [Socher et al., 2013] focused on sentiment analysis, and various natural language inference tasks (NLI) like MNLI [Williams et al., 2017], QNLI [Demszky et al., 2018], CB [Marneffe et al., 2019], and RTE [Dagan et al., 2006]. It also includes STS-B [Cer et al., 2017] for assessing sentence similarity, and tasks like MRPC [Dolan and Brockett, 2005] and QQP [Wang et al., 2018] for evaluating paraphrasing similarity. Coreference resolution is tested through WSC [Levesque et al., 2012], BoolQ [Clark et al., 2019] for question answering, and WiC [Pilehvar and Camacho-Collados, 2018] for word sense disambiguation.

A.2 Implementation Details

Following prior work [Zhang et al., 2020], we use the validation set as the testing set in the absence of a dedicated testing set. For datasets with fewer than 100,000 entries, the validation set is split into half: one for validation and the other for testing. In contrast, for larger datasets, 1000 samples from the training set are repurposed for validation purposes, and the existing validation set is used for testing. The task embedding dimension size is 200. T5 model is finetuned using the AdamW optimizer [Loshchilov and Hutter, 2017], employing a 3e-4 learning rate with linear decay and a warmup of 500 steps. Unless specified, we train for 65k steps with an effective batch size of 128, and evaluations are conducted at every 1000-step interval on the development set. All tests are conducted five times using distinct random seeds, and the average of each result is reported.

A.3 Model scaling

To investigate the impact of scaling pre-trained model sizes on the performance of IP, we replicate the experiments of [Wang et al., 2023] on GLUE tasks. The result illustrated in Figure 5, compares IP against full fine-tuning (FT), prompt-tuning(PT), ATTEMPT, MTP, Hyperformer++ (HF), and Hyperdecoder(HD). using three sizes of the T5 model: T5-Small(60m), T5-Base(220m), and T5-Large(770m). In three different scales of T5 models, we observe that IP achieves superior and competitive performances. The result reveals its effectiveness across a spectrum of models, from 60 million to 770 million parameters.

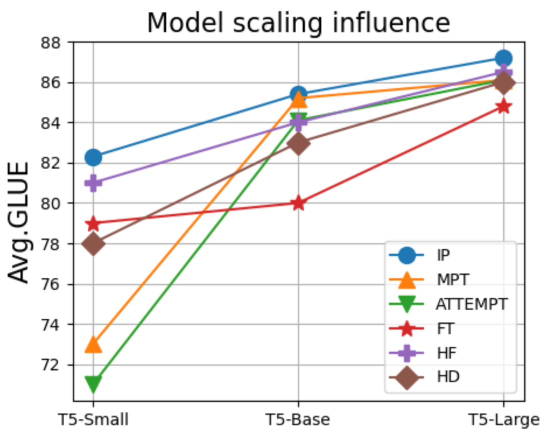


Figure 5: Performance comparison of different baseline models ranging from T5-Small to T5-Large, illustrated as a function of model size.

A.4 Samples of Task Descriptions

Table 6 shows an example of a set of task descriptions we constructed. Our principle for constructing task descriptions adheres to the characteristics of human language: (1) similar tasks have comparable descriptive sentences; (2) unrelated tasks exhibit low similarity in their descriptions. Whether multi-task learning or adapting to new tasks, we only use a set of task descriptions to train IP.

A.5 Decoder-Only Models

A.5.1 Overview

This section introduces how to apply IP to decoder-only PLMs. As shown in Figure 6, we utilize an additional encoder model, serving as the form module, to construct task embeddings from the modified samples. We then leverage these embeddings to modify the hidden states of the PLM. The specific procedure is given in Chapter 3. Notably, after modifying the hidden states of the PLM, we input the combination of {task description + sample} into the modified PLM to obtain the final results. To demonstrate the generality of IP, we verified its effectiveness on both the classical small model GPT-2 and the widely-used LLM Llama2[Touvron et al., 2023].

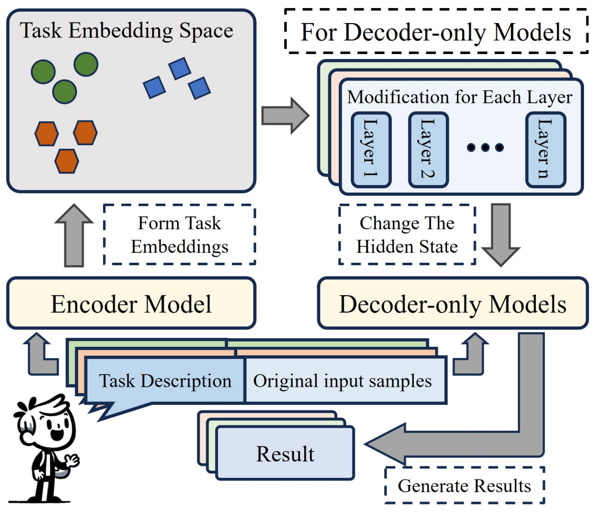


Figure 6: Apply IP to decoder-only models. We introduce an additional parameter-fixed encoder model, serving as the form module.

A.5.2 GPT2

We adopted GPT-2 Medium (355M) as the backbone model for IP. The details of data processing and parameter settings have been elaborated in the previous sections. We selected the Roberta-base model as the form module and fixed its parameters. Similar to previous works[Edalati et al., 2021; Mihaila, 2021; Sun et al., 2023], we conducted experiments on the GPT-2 model using the GLUE dataset. It is worth noting that since the aforementioned PEFT method does not natively support decoder-only models, it is not included in the comparative experiments.

As shown in Table 4, although our method exhibited some disadvantages on certain datasets, it still achieved the best average performance. This indicates that IP can improve the multi-task capability of decoder-only architectures, while also demonstrating the generality of our approach.

A.5.3 Llama2

In this section, we choose the popular open-source language model Llama2-7b as our backbone model. We use RoBERTa-base and RoBERTa-large as the form module respectively, to observe the impact of different-sized form modules on IP performance.

We evaluate the effect of IP from the perspectives of open-ended generation tasks, rather than traditional natural language understanding tasks, such as GLUE or SuperGlUE. This is because conventional NLU tasks are overly simplistic, consisting of single-choice questions with extremely short target lengths, typically requiring only a single token output. Such tasks are too easy for large language models and can be readily solved even under zero-shot conditions. Therefore, we focus more on more complex tasks.

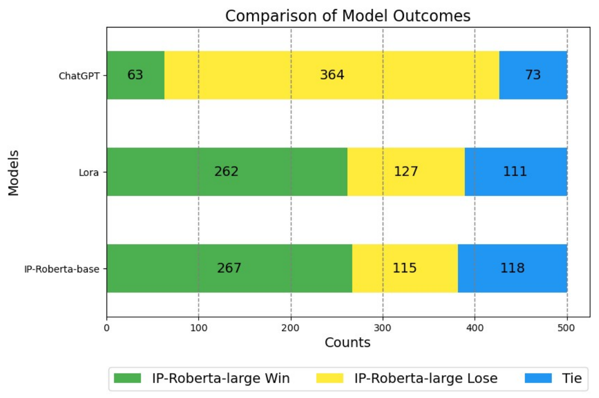


Figure 7: The GPT4 evaluation results between IP with RoBERTa-large and baselines (IP with RoBERTa-base, Lora, chatgpt).

Considering that {task description + input sample} is a core step of the IP method, we select 5,500 samples containing {instruction + input} combinations from the Alpaca dataset, with 5,000 for training and 500 for testing. This selection stems from the characteristic of the Alpaca dataset that similar inputs often have similar expressions, which aligns well with the principle of IP. For example, instructions for mathematical formula inputs are typically related to math problems, while instructions for literary topic inputs often pertain to writing. We perform instruction tuning on all methods using the training set, then generate text on the test set, with the text quality ultimately evaluated by GPT-4.

As shown in figure 7, compared to the Llama with Lora, the output quality of IP (based on the RoBERTa-large form module) is higher. Although our model still lags behind ChatGPT, this is mainly due to ChatGPT's larger model scale and higher-quality training data. Nevertheless, our model still produces higher-quality outputs in some samples. Overall, IP plays a positive role in LLM. In terms of selecting the form module, we found that IP based on RoBERTa-large provides higher answer quality than models using RoBERTa-base, as larger-scale form modules perform better in extracting task embeddings.

A.5.4 Summary

The primary purpose of IP is to enhance the multi-task learning capabilities of small-scale models. Whether T5 or GPT-2, which can run on a single

Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	Avg.
Fine-tuning	74.8	82.1	78.4	**84.5**	87.6	**65.3**	91.2	80.5
Adapter	**75.4**	**83.2**	75.2	83.1	86.7	64.2	92.3	79.9
Fine-tuning-m	72.1	82.8	78.6	81.8	83.2	61.2	89.8	78.5
Adapter-m	68.7	81.8	77.1	84.2	85.8	63.9	91.5	79.0
IP	75.2	81.9	**80.2**	84.1	**88.2**	62.2	**92.7**	**80.6**

Table 4: Performance comparison of different methods. Fine-tuning and adapter denote those methods individually adapt to each task with no parameter sharing, while Fine-tuning/adapter-m denote these methods adapt to multiple tasks simultaneously.

consumer-grade GPU, IP can effectively improve its multi-task learning performance. This demonstrates the powerful capabilities and broad applicability of IP. Additionally, in experiments of LLM, we have observed that IP also improves the output quality of LLM models.

A.6 Construct Task Descriptions

To design IP, we tried various ways to construct task descriptions, which included automatic generation by GPT, intuitive manual construction, and the construction rules described in our paper. We also attempted to modify task descriptions through data augmentation (i.e., synonym replacement) during training. The effects of IP on the construction methods for these four task descriptions are shown in the table 5. The first two approaches (GPT generation and human construction) exhibited slight disadvantages. We speculate this is due to their higher randomness in generating task descriptions. Additionally, data augmentation didn't statistically improve the final results. To simplify the IP process, we **remove** the data augmentation step. Moreover, this also indicates that our rules for constructing task descriptions are already effective.

Method	GLUE Avg.
GPT Generation	84.9
Human construction	84.7
Based on rules (DA)	85.4
Based on rules	85.5

Table 5: The impact of different task description construction methods. The method "based on rules" is the one used in the main paper, and "DA" refers to the inclusion of data augmentation techniques (such as synonym substitution).

Dataset	Task Description
	GLUE
MNLI	MNLI: Determine the relationship between the following sentences from a variety of genres:
SST2	SST2: Tell me the sentiment of the film review:
CoLA	CoLA: Tell me the grammatical acceptability of the film statement:
STS-B	STS-B: Evaluate the level of semantic similarity for these sentences:
MRPC	MRPC: Judge whether the following pair of sentences are semantically equivalent:
QQP	QQP: Assess if the following pair of questions are semantically equivalent:
QNLI	QNLI: Determine if the second statement provides an answer to the question statement:
RTE	RTE: Determine the logical relationship between these two sentences, if the second logically follows from the first:
	SuperGLUE
BoolQ	BoolQ: Tell me whether the statement is true or false:
WiC	WiC: Check whether the word in focus has a consistent meaning in the following contexts:
CB	CB: Determine the relationship between these two statements:
WSC	WSC: Interpret the following sentence and decide what the pronoun is referring to:
	Other
SciTail	SciTail: Determine the relationship between the following statements from science materials:

Table 6: Task description of IP for all datasets.