Dynamic Task Vector Grouping for Efficient Multi-Task Prompt Tuning

Peiyi Zhang[1], Richong Zhang[1,2]*, Zhijie Nie[1,3], Ziqiao Wang[4],
[1]CCSE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]Shen Yuan Honors College, Beihang University, Beijing, China
[4]School of Computer Science and Technology, Tongji University
{zhangpy,zhangrc,niezj}@act.buaa.edu.cn
ziqiaowang@tongji.edu.cn

Abstract

Multi-task prompt tuning utilizes multiple high-resource source tasks to improve performance on low-source target tasks. Existing approaches transfer the soft prompt trained by combining all source tasks or a single "high-similar" source task one-time-only. However, we find that the optimal transfer performance often comes from a combination of source tasks, which is neither one nor all. Further, we find that the similarity between source and target tasks also changes dynamically during fine-tuning after transfering, making similarity calculation in the initiation stage inadequate. To address these issues, we propose a method called Dynamic Task Vector Grouping (DTVG), whose core ideas contain (1) measuring the task similarity with task vectors instead of soft prompt, (2) grouping the optimal source task combination based on two metrics: *target similarity* and *knowledge consistency*; (3) dynamically updating the combination in each iteration step. Extensive experiments on the 26 NLP datasets under different settings demonstrate that DTVG effectively groups similar source tasks while reducing negative transfer, achieving the start-of-art performance.

1 Introduction

Full parameter fine-tuning (FT) of large pre-trained language models (PLMs) has shown significant success in addressing various natural language processing (NLP) tasks. However, the conventional fine-tuning paradigm requires substantial memory and computational resources. Recently, parameter efficient fine-tuning (PEFT) (Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Zaken et al., 2022; Hu et al., 2022) aims to achieve comparable results of FT by updating a significantly small set of the model parameters.

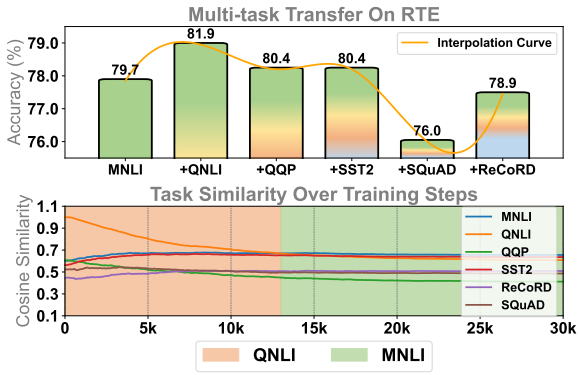

Figure 1: In the upper part, we use performance on the RTE validation set to study potential conflicts of source tasks. We incrementally add source tasks with a random order and train soft prompt by examples-proportional mixing (Raffel et al., 2020). In the bottom part, we calculate the cosine similarity between the average pooled representations of the prompt tokens (Vu et al., 2022). We initialize the RTE soft prompt using the source task's soft prompt with the highest similarity. The legend marker denotes the source task with the highest similarity, which shifts from QNLI to MNLI during fine-tuning.

Soft prompt tuning (PT) (Lester et al., 2021), as an effective PEFT method, achieves a trade-off between effectiveness and efficiency. During training, a series of learnable soft prompt vectors prepended to the input are updated while the original PLMs are frozen. Unlike methods such as LoRA (Hu et al., 2022) and Adapter (Houlsby et al., 2019), PT is independent of the model architecture and can be applied to various models without modification. Although promising, the existing study (Asai et al., 2022) demonstrates PT still underperforms compared to FT, particularly in the case of low-resource tasks. An additional issue with PT is sensitivity to the initialization and needs longer tuning for converge (Lester et al., 2021).

Recent works (Vu et al., 2022; Asai et al., 2022; Feng, 2023; Wang et al., 2023) address the above limitations by transferring soft prompt from high-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2025, pages 26805–26821

resource source task to low-resource target task.

Specifically, they initialize the soft prompt for the target task by either (1) learning a common soft prompt across all source tasks or (2) learning a soft prompt for each source task and selecting one with the task similarity. Subsequently, the soft prompt is tuned exclusively using limited training samples from the target task. These transfer approaches effectively maintain the parameters efficiency of soft prompts and demonstrate superior performance compared to vanilla prompt tuning.

Despite substantial progress, we challenge the rationality of some straightforward ideas in existing approaches. We first check whether existing methods achieve optimal performance. In the upper part of Figure 1, we observe that a subset of source tasks achieves the best transfer performance, neither all source tasks nor a source task. Additionally, Vu et al. (2022) demonstrates MNLI, QNLI, and QQP positively transfer to the RTE dataset, while we find that their gradual addition does not yield a consistent monotonic improvement due to the potential conflicts among source tasks. These observations revealed that we should find a group of source tasks for each target task and consider potential conflicts between source tasks besides the similarity to target tasks.

Further, we check whether *"the most similar source task"* will change in the tuning stage of the target task. We study a single-task version of SPoT (Vu et al., 2022), which transfers the soft prompt from a source task to initiate the target task via similarity measure between their learned soft prompt. In the bottom part of Figure 1, we find that *"the most similar source task"* of RTE shifts from QNLI to MNLI over time. Recall that the low-resource characteristics of the target task hinder sufficient convergence of soft prompt; therefore, it is unsurprising that we cannot select the truly most similar task with an unconverted soft prompt of the target task. This observation suggests that dynamically updating the selected source task during the target task's fine-tuning may enhance the sustainable acquisition of knowledge.

Motivated by these valuable empirical observations, we propose a method called Dynamic Task Vector Grouping (DTVG). Specifically, We first introduce a novel task similarity metric, the dot product between task prompt vectors (TPV), which steadily achieves a better transfer performance than the current metric, the cosine similarity between soft prompts. Based on this metric, we introduce a source task grouping method to select the transfer source task group for each target task with two metrics, including *target similarity* and *knowledge consistency*. Then, a multi-task merging method is used to weighted sum the task vectors from the target task and the selected source tasks, synthesizing the initialization soft prompt for the target task. During the fine-tuning stage of the target task, we track the task similarity changes and dynamically update the source task group, which will effectively improve transfer performance.

In summary, our major contributions are to:

- We present an effective task similarity metric, based on the task prompt vectors, to measure the transfer performance between tasks.
- We propose DTVG, a dynamic task vector grouping method that assembles and updates a source task group for each target task throughout the iterative training process to ensure sustainable acquisition of knowledge.
- We confirm the effectiveness of DTVG on the 26 datasets based on T5 and Llama3 under different settings, surpassing the advanced models and achieving SOTA performance.

2 Background

Soft Prompt Tuning Soft Prompt Tuning (PT) (Lester et al., 2021) proposes strategically inserting the learned soft prompt into the input. Formally, for a task t with the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{|D|}$, we fine-tuning a pre-trained model F_Θ to perform better in the task t with its parameter Θ frozen. Instead, the learnable soft prompt $P \in \mathbb{R}^{d \times r}$ is introduced, where d is the hidden state dimension of F_Θ and r is the soft prompt length. The soft prompt P and the token embedding matrix $E(x_i)$ are spliced as the input of F_Θ. Then, the soft prompt P^* is learned to boost the posterior probability of correct output y_i:

$$P^* = \arg\max_P \mathbb{E}_{(x_i,y_i)\in\mathcal{D}}[\mathrm{P}(y_i|P; E(x_i))] \quad (1)$$

Although the PT method has shown great success in various NLP tasks, it still faces the low-source challenge: Too few training samples prevent the soft prompts from converging, which can result in huge performance differences under different soft prompt initializations.

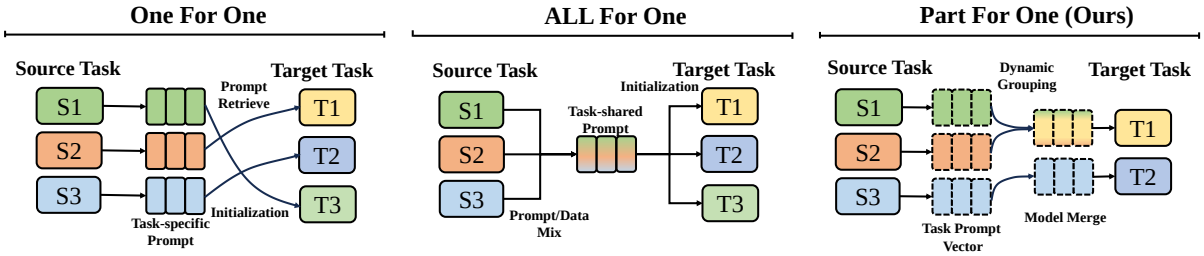

Figure 2: An overview of methods for comparison. One For One, initialize a target task by retrieving the task-specific prompt from one of the most similar source tasks based on task similarity. ALL For One, initialize a target task by learning appropriately across all source tasks based on prompt or data mix. Our Method: Part For One, dynamic group a subset of source tasks and merge their task prompt vectors.

Multi-Task Prompt Tuning Multi-Task Prompt Tuning (Mahabadi et al., 2021; Vu et al., 2022; Asai et al., 2022; Wang et al., 2023) is proposed to address the low-source challenge of PT. Formally, given a high-resource source task set $\mathcal{S} = \{s^1, s^2, \ldots, s^n\}$, where n is the number of source tasks, Multi-Task Prompt Tuning improve the performance of a low-source target task t by transfer learning from $\mathcal{S}$. Current methods usually contain two stages: (1) Learning the transferable soft prompts P_{mix} from $\mathcal{S}$, defined as $P_{\text{mix}} = G(\mathcal{S}, t)$ where G is the learning method; (2) Adopting P_{mix} to t and re-tuning P_{mix} with maximum training steps $N_{\max}$ on the training set of task t.

Multi-task Prompt Tuning does not impose restrictions on G to get P_{mix} and how to adopt P_{mix} on t, excepting that the transfer ones must be soft prompts. Therefore, there are two representative lines of work to be highlighted. One For One: G serves as a retriever and selects the learned soft prompt of the most similar s to initialize for t. SPoT (Vu et al., 2022) regards the soft prompts as the task embeddings and measures task similarity via cosine similarity between soft prompts. Feng (2023) learns G to predict transfer gain by randomly sampling soft prompt pairs. All For One: G serves as a blender and learns the task-shared prompt from source task set $\mathcal{S}$ via different mix strategies. SPoT (Vu et al., 2022) also learns a single soft prompt through multi-task learning by mixing data. ATTEMPT (Asai et al., 2022) trains an attention module and mixes instance-wise prompts from all source tasks $\mathcal{S}$. MPT (Wang et al., 2023) extends the multi-task training method of SPoT by learning task-shared and task-specific modules. TPT (Wu et al., 2023) propose to retrieve token-wise soft prompt from the prompt bank.

Task Arithmetic Task Arithmetic (Ilharco et al., 2023; Zhang et al., 2024; Ortiz-Jimenez et al., 2024) as a newly emerged cost-effective approach demonstrates the effectiveness of multi-task training by operating task vectors derived from different tasks, where task vectors are given as the relative difference between the initialized parameters and those obtained after fine-tuning, capturing the changes induced by the adaptation process in weight space. Our proposed approach is inspired by Task Arithmetic. Similar to task vectors, the task prompt vectors (TPV) $T = [v_1, \ldots, v_r] \in \mathbb{R}^{d \times r}$ are defined as the difference between P_{init} and P^*, i.e. $T = P^* - P_{\text{init}}$, where $v_i \in \mathbb{R}^{d \times 1}$ represent the i-th vector in T. Concurrent work (Belanec et al., 2024) uses TPV to enable generalization to new target tasks without training. In contrast, we introduce TPV to address the issue of potential negative transfer in multi-task prompt tuning.

3 Method

3.1 Overview

We propose a novel multi-task prompt tuning approach, Dynamic Task Vector Grouping (DTVG), which dynamically groups a subset from the source task set to transfer to the target task. Therefore, G in our method serves as a grouper, allowing a specific target task to selectively leverage partially related source tasks, mitigating the risk of negative transfer. As shown in Figure 2, DTVG actually follows the idea of Part For One and distinguishes itself from existing methods.

DTVG consists of two stages: (I) Task Prompt Vector Learning to obtain a tuned TPV for each source and target task and (II) Multi-Task Prompt Transfer to group source tasks' TPV and merge it with the target vector's TPV. Note that the first stage only needs to be performed once, while the second

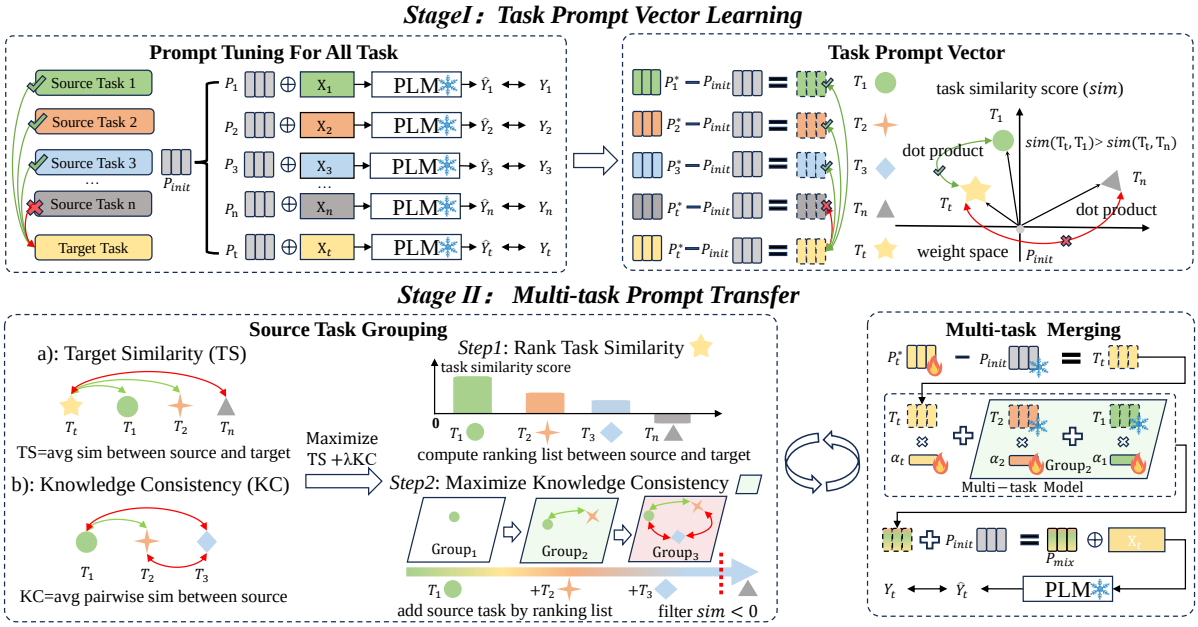

Figure 3: DTVG is to learn dynamic grouping partially related source tasks, including two stages: I) Task prompt vector Learning; II) Multi-task Prompt Transfer. In the first stage, we obtain task prompt vectors via vanilla prompt tuning. In the second stage, Source Task Grouping and Multi-task Merging are executed at each iteration step.

stage is iterative, and the source task group will be dynamically updated during the fine-tuning process of the target task. An algorithm-style process of DTVG is provided in Appendix F.

3.2 Task Prompt Vector Learning

In the first stage, we obtain the soft prompt by individually tuning both the source and target task via the same initialization P_{init} and calculate their task prompt vectors T. Therefore, we have $n+1$ task prompt vectors from $\mathcal{S} \cup \{t\}$.

We propose using the average token-wise task prompt vectors to compute their dot product, allowing us to predict task similarity. This method enables a quantitative assessment of task relationships, as illustrated at the top of Figure 3. Specifically, given two task prompt vectors T_1 and T_2 from s^1 and s^2, we can calculate the similarity between tasks s^1 and s^2. The task similarity scores sim between tasks is defined as follows:

$$sim(T_1, T_2) = \frac{1}{r^2} \left(\sum_{i=1}^{r} v_i^1 \right)^\top \left(\sum_{j=1}^{r} v_j^2 \right) \quad (2)$$

where r denotes the length of soft prompt tokens.

To evaluate the effectiveness of this metric, we conduct transfer experiments on the SuperGLUE benchmark. As shown in Table 1, TPV demonstrates consistent positive transfer, whereas SPoT exhibits negative transfer on WSC and CB, showing the superiority of our metric. Please refer to Appendix C.1 and I for the experiment details and visual analysis, respectively.

	SuperGLUE					
Method	Multi	Bool	WiC	WSC	CB	**Avg.**
PT	72.7	76.0	62.6	67.3	82.1	72.1
SPoT	74.9 ↑	80.6 ↑	65.2 ↑	63.5 ↓	78.6 ↓	72.6
TPV	74.2 ↑	81.3 ↑	66.1 ↑	67.3 -	92.9 ↑	76.4

Table 1: Performance on SuperGLUE benchmark.

3.3 Multi-task Prompt Transfer

In the second stage, we introduce an iterative process for multi-task prompt transfer. As shown in the bottom of Figure 3, for each iteration, Source Task Grouping and Multi-Task Merging are executed sequentially to obtain P_{mix}.

Source Task Grouping Source task grouping aims to group a subset of source tasks $\mathcal{S}' \subseteq \mathcal{S}$. Source tasks in $\mathcal{S}'$ should not only be similar to the target task but also possess consistency of knowledge. We propose two metrics to characterize the source task group quantitatively, including *Target Similarity* and *Knowledge Consistency*.

Target Similarity: To measure the transferability of multiple source tasks to the target task, we define a target similarity score TS as the average of

the similarity between each source and target task prompt vector pair (T_i, T_t), which is formulated as

$$\mathrm{TS}(\mathcal{S}, t) = \frac{1}{|\mathcal{S}|} \sum_{s^i \in \mathcal{S}} sim(T_i, T_t) \tag{3}$$

Knowledge Consistency: In multi-task transfer learning scenarios, conflicts among source tasks are prevalent. For example, in NLP, words crucial for sentiment (e.g., "good") may have varying significance in topic classification, leading to ambiguity and reduced performance in the target task. We propose to quantify the conflicts within a task group by calculating the average pairwise sim between tasks. More formally, we defined the Knowledge Consistency Score (KC):

$$\mathrm{KC} = \begin{cases} \frac{2}{n(n-1)} \sum\limits_{i<j} sim(T_i, T_j) & \text{if } |\mathcal{S}| \geq 2, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Therefore, the objective for selecting a source task group can be defined using TS and KC:

$$\max_{\mathcal{S}' \subseteq \mathcal{S}} \left(\mathrm{TS}(\mathcal{S}', t) + \lambda \mathrm{KC}(\mathcal{S}') \right) \tag{5}$$

$$\text{subject to,} \quad \forall s^i, s^j \in \mathcal{S}', \ s^i \neq s^j$$

where λ is the hyperparameter to achieve the trade-off between TS and KC.

However, the process to find the optimal S' is equivalent to the Set Cover problem, which is the NP-Hard. As the number of tasks increases, the selection from $2^{|\mathcal{S}|}$ subsets becomes infeasible. Therefore, we use a heuristic algorithm to find a suboptimal subset, achieving a balance between efficiency and effectiveness. As shown in Figure 3, the algorithm consists of two steps: (1) $sim(T_i, T_t)$ between each source task s^i and target task t is computed. Then $\{sim(T_i, T_t)\}_{i=1}^{n}$ is ranked in the descending order, obtaining a rank list $\Pi = \{\pi^1, \pi^2, \ldots, \pi^n\}$, where π^j is the source task with j-th highest $sim(T_i, T_t)$; (2) The source tasks with $sim(T_i, T_t)$ are added to the set S' one by one in the order in T until the $\mathrm{KC}(\mathcal{S}')$ is no longer increasing. The implementation details are provided in the Appendix G.

Multi-task Merging Multi-task merging aims to merge the task prompt vectors from the source task group and the target task to get a final soft prompt P_{mix}. Specifically, P_{mix} is obtained by the sum of (1) the rescaled soft prompt task vectors of $\mathcal{S}' \cup \{t\}$ and (2) a common initialization prompt, which can be denoted as

$$P_{\text{mix}} = \underbrace{P_{init}}_{\text{Initialization}} + \underbrace{\alpha_t T_t + \sum_{s \in \mathcal{S}'} \alpha_s T_s}_{\text{Merged Task Prompt Vector}} \tag{6}$$

where $\alpha \in \mathbb{R}^l$ is token level scaling term initialized to all-ones vector. In practice, we employ rescaled task prompt vectors to compute the task similarity score (Equation 2).

Iteration Update In each training step, we sequentially execute the above two steps to ensure the correct source task group $\mathcal{S}'$ selection to compute P_{mix} with in-batch. In practice, we observe that in the early stages of training, the grouping of source tasks exhibits significant fluctuations due to the insufficient convergence of the target task prompt vectors. As the iterations progress, the dynamic grouping gradually stabilizes and ultimately maintains consistency (see Section 4.4 for details).

4 Experiments

4.1 Experiment Setup

Datasets We evaluate the model's natural language understanding capabilities using the GLUE and SuperGLUE benchmarks. In addition, we also use four question-answering datasets from the MRQA 2019 benchmark and four datasets from the "other" benchmark. In the following, we introduce the source tasks and target tasks separately. Further details can be found in Appendix B.

Source Tasks: Following Wang et al. (2023), we set n to 6, and use the same large-scale datasets as source tasks, including MNLI, QNLI, QQP, SST2 from GLUE, ReCoRD from SuperGLUE, and SQuAD from MRQA 2019.

Target Tasks: we use all 8 datasets from GLUE, 5 datasets (excluding ReCoRD) from SuperGLUE, 4 datasets (excluding SQuAD) from MRQA 2019, and 4 datasets from the "other" benchmark.

Models We adopt the model setup from (Lester et al., 2021) for prompt tuning. Our experiments mainly utilize T5-base with the soft prompt of length 100, while in ablation studies, we also explore other scales of T5 in Section 4.4.

Baselines We compare our method with several baseline methods. (1) no transfer learning, which updates model parameters for the each target task without source task, including Finetuning (FT),

Prompt Tuning (PT) (Lester et al., 2021), BitFit (Zaken et al., 2022), Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), DePT (Shi and Lipani, 2024), as well as multi-task versions of FT, Adapter, HyperFomer (Mahabadi et al., 2021), and HyperDecoder (Ivison and Peters, 2022). Note that we exclude ACCEPT (Lin et al., 2024) due to the lack of accessible open-source code, which prevents an evaluation of its ability to address sensitivity to prompt initialization. (2) transfer learning + one for one, where transfer soft prompt from one source task to each target task, such as SPoT (Vu et al., 2022) (3) transfer learning + all for one, where transfer soft prompt from all source tasks to each target task, including ATTEMPT (Asai et al., 2022), MPT (Wang et al., 2023), TPT (Wu et al., 2023) as well as multi-task versions of ATTEMPT, and MPT. For a fair comparison, we directly quote the results of the baselines reported in previous works (Asai et al., 2022; Wu et al., 2023; Wang et al., 2023; Shi and Lipani, 2024) whenever possible, and utilize publicly available source code to ensure consistent experimental settings.

Implementation Details For both the Task Prompt Vector Learning and Multi-task Prompt Transfer stage, we train on high-resource source tasks for 300K steps, following Vu et al. (2022). For the target tasks, we set $N_{\max}$ to 30K. Aligning with standard prompt tuning methods (Lester et al., 2021), we use a default learning rate of 0.3 and select checkpoints with the highest validation set scores to extract task prompt vectors. In the Multi-task Prompt Transfer stage, we apply two-speed learning rates for different modules. We conduct transfer experiments four times and report the average results. Please see Appendix C for details.

Parameter Efficiency For both source and target tasks, we compute the task prompt vector $T \in \mathbb{R}^{r*d}$, where r is the length of the soft prompt and d is the model dimension. For each source task, we introduce a learned scaling term $\alpha \in \mathbb{R}^r$. Our framework enables knowledge transfer from partial source tasks to the target task, therefore, the total number of learned parameters ranges from $r + r * d = r*(d+1)$ to $(n+1)*r+r*d = r*(d+n+1)$, where n is the number of source tasks. We compare different methods' trainable parameters under the least favorable conditions of DTVG in Table 2.

4.2 Main Results

Full-dataset Transfer Table 2 provides the performance and parameter comparison for each dataset on the GLUE and SuperGLUE benchmarks across different baselines. Additionally, we visualize the result on GLUE (see Appendix A). Notably, our proposed method, DTVG, outperforms others by achieving the *highest average performance* on GLUE and SuperGLUE with a *minimal parameter tuning fraction* of 0.035%, in contrast to the fine-tuning. When compared to prompt tuning in terms of low-resource datasets, DTVG significantly improves the performance of the target task, such as CoLA (10.6% vs. 69.1%) and CB (67.9% vs. 97.6%). Simultaneously, our multiple experiments demonstrate that DTVG is robust for addressing inappropriate soft prompt initialization leading to performance degradation. Please see Appendix D for details on MRQA and "Other" benchmarks.

Few-shot Adaptation We compare our method with other baselines on BoolQ, CB, and SciTail in Table 3. On average, our method outperforms the baselines in low-resource settings with only (k= 4,16,32) shots, indicating that our DTVG is adept at harnessing knowledge from multiple source tasks for effective transfer in scenarios with limited training samples. More details about GLUE and SuperGLUE are given in Appendix E.

4.3 Ablation Study

Source Task Grouping Strategy We conduct ablation experiments on the SuperGLUE benchmark to study the impact of two different perspectives for source task grouping. For a) Target Similarity (TS), we only merge TPV with $sim \geq 0$. For b) Knowledge Consistency (KC), we select the source task group with the highest KC among all source task combinations. As shown in Figure 4, these strategies can improve performance consistently. KC improves the average performance on SuperGLUE from 74.8 to 75.1, suggesting that mitigating the conflict among multiple source tasks is critical for effective multi-task prompt tuning, even when the task combinations may not be directly related to the target tasks. ST improves the average performance on SuperGLUE from 74.8 to 75.9, indicating that sim can effectively evaluate and leverage similar source tasks for transferring.

Multi-task Prompt Transfer Strategy We conduct a study to ablate different multi-task prompt

Method	param \ task	GLUE									SuperGLUE					
		MNLI (393K)	QQP (364K)	QNLI (105K)	SST2 (67K)	STS-B (7K)	MRPC (3.7K)	RTE (2.5K)	CoLA (8.5K)	Avg.	Multi (5.1K)	Bool (9.4K)	WiC (6K)	WSC (554)	CB (250)	Avg.
no transfer learning																
Finetuning$_1$	220M	86.8	91.6	93.0	94.6	89.7	90.2	71.9	61.8	84.9	72.8	81.1	70.2	59.6	85.7	73.9
PT$_1$	76.8K	81.3	89.7	92.8	90.9	89.5	68.1	54.7	10.6	72.2	58.7	61.7	48.9	51.9	67.9	57.8
BitFit$_1$	280K	85.3	90.1	93.0	94.2	90.9	86.8	67.6	58.2	83.3	74.5	79.6	70.0	59.6	78.6	72.5
Adapter$_1$	1.9M	**86.5**	90.2	93.2	93.8	90.7	85.3	71.9	64.0	84.5	**75.9**	**82.5**	67.1	67.3	85.7	75.7
LoRA$_4$	3.8M	86.3	89.0	93.2	94.3	90.9	90.1	75.5	63.3	85.3	72.6	81.3	68.3	67.3	92.9	76.5
DePT$_4$	76.8k	85.0	90.4	93.2	94.2	90.8	**90.7**	79.1	63.8	85.9	74.3	79.3	68.7	67.3	92.9	76.5
Finetuning$^*_1$	28M	85.7	91.1	92.0	92.5	88.8	90.2	75.4	54.9	83.8	74.4	81.1	70.0	71.2	85.7	76.1
Adapters$^*_1$	1.8M	86.3	**90.5**	93.2	93.0	89.9	90.2	70.3	61.5	84.4	72.6	82.3	66.5	67.3	89.3	75.6
HyperFomer$^*_1$	638K	85.7	90.0	93.0	94.0	89.7	87.2	75.4	63.7	84.8	72.9	82.5	69.0	67.3	85.7	75.4
HyperDecoder$^*_1$	1.8M	86.0	**90.5**	**93.4**	94.0	90.5	87.7	71.7	55.9	83.7	70.4	78.8	67.1	61.5	82.1	72.0
transfer learning + one for one																
SPoT$_1$	76.8K	85.4	90.1	93.0	93.4	90.0	79.7	69.8	57.1	82.3	74.0	77.2	67.0	50.0	46.4	62.9
transfer learning + all for one																
ATTEMPT$_1$	232K	84.3	90.3	93.0	93.2	89.7	85.7	73.4	57.4	83.4	74.4	78.8	66.8	53.8	78.6	70.5
MPT$_3$	77.6K	85.9	90.3	93.1	93.8	90.4	89.1	79.4	62.4	85.6	74.8	79.6	69.0	67.3	79.8	74.1
TPT$_2$	539K	85.5	90.1	93.2	**94.7**	89.8	89.7	82.3	59.8	85.6	74.4	80.1	69.8	67.3	94.6	77.2
ATTEMPT$^*_1$	96K	83.8	90.0	93.1	93.7	90.8	86.1	79.9	64.3	85.2	74.4	78.3	66.5	69.2	82.1	74.1
MPT$^*_3$	10.5K	84.3	90.0	93.0	93.3	90.4	89.2	82.7	63.5	85.8	74.8	79.2	70.2	67.3	89.3	76.1
transfer learning + part for one																
DTVG (ours)	77.5K	86.0$_{0.2}$	90.3$_{0.1}$	93.1$_{0.0}$	93.2$_{0.0}$	**91.0**$_{0.2}$	90.4$_{0.2}$	**86.3**$_{0.6}$	**69.1**$_{1.0}$	**87.4**	74.5$_{0.7}$	81.4$_{0.1}$	**71.1**$_{0.5}$	**69.9**$_{3.6}$	**97.6**$_{3.4}$	**78.9**

Table 2: Results on GLUE and SuperGLUE benchmark. "param\task" denotes the number of learnable parameters for each task on the GLUE. * denotes multi-task learning on target tasks. $_1$ sourced from Asai et al. (2022), $_2$ sourced from Wu et al. (2023), $_3$ sourced from Wang et al. (2023) and $_4$ sourced from Shi and Lipani (2024). We differentiate high-resource and low-resource tasks using gray and blue, respectively, to highlight our contribution.

Task	k	Method						
		FT	PT	HF	ATP	MPT	DePT	Our
BoolQ	4	50.5	61.6	48.0	61.8	62.2	**62.7**$_{5.4}$	60.6$_{1.5}$
	16	56.5	61.9	50.2	60.0	63.3	66.9$_{4.4}$	**72.3**$_{1.4}$
	32	58.4	61.7	58.3	65.3	68.9	67.2$_{3.4}$	**73.5**$_{1.1}$
CB	4	57.7	53.5	60.7	**82.1**	73.6	75.0$_{5.1}$	**86.9**$_{1.7}$
	16	77.0	63.5	76.3	78.5	78.6	78.6$_{4.3}$	**82.1**$_{2.9}$
	32	80.0	67.8	81.4	**85.7**	82.1	82.1$_{2.3}$	84.5$_{1.7}$
SciTail	4	79.6	57.7	**82.0**	80.2	80.2	78.1$_{2.5}$	78.3$_{1.1}$
	16	80.0	60.8	86.5	79.5	**87.3**	78.5$_{1.4}$	82.1$_{2.9}$
	32	81.9	60.2	85.8	80.2	**86.3**	85.4$_{3.1}$	85.3$_{2.5}$

Table 3: Few-shot adaptation on BoolQ, CB, and SciTail datasets, where FT, HF, ATP denote Finetuning, HyperFomer, and ATTEMPT, respectively.

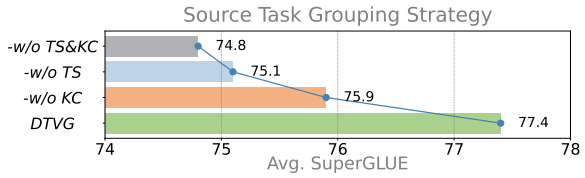

Figure 4: Ablation study for the source task grouping.

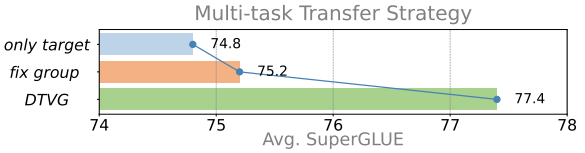

Figure 5: Ablation study for multi-task prompt transfer.

transfer strategies, including 1) *only target*: This strategy focuses solely on learning the target task prompt and its associated scaling term for the task prompt vectors. 2) *fix group*: This strategy fixes the initial source task group, thus eliminating the effect of dynamic grouping, which relies on the specific grouping of source tasks. Figure 5 shows that using a fixed group of source tasks results in a performance drop (77.4 vs. 75.2), suggesting that the choice of the source task group is important. This emphasizes the need for our approach, namely DTVG's ability to efficiently group source tasks by dynamic iteration, thereby improving performance.

4.4 Additional Analysis

We extend our experiments to comprehensively evaluate the performance of DTVG, including model scaling, natural language generation, generalization to other LLMs, and dynamic grouping during training. However, for some experiments without a standard evaluation protocol, we analyze DTVG only against some fundamental baselines.

Model Scaling Figure 6 illustrates the results on three SuperGLUE datasets with different scales of the T5 model. We observe that as the model size increases, performance across different tasks improves. This indicates that our method indeed benefits from a larger model capacity. Please refer

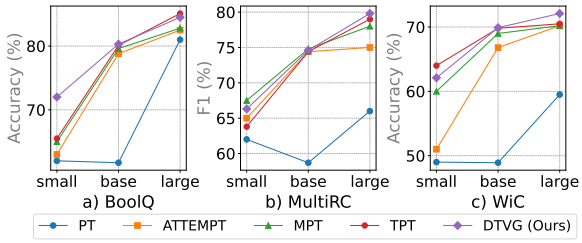

Figure 6: Model Scaling on BoolQ, MultiRC, and WiC.

Task	Method	Metris			
		BLEU	R-1	R-2	R-L
E2E	PT	0.274	62.1	36.3	47.0
	DTVG	0.331	63.6	37.5	47.9
CommonGen	PT	0.056	33.3	9.9	27.6
	DTVG	0.067	36.6	11.0	29.1
WebNLG	PT	0.293	64.4	39.6	52.3
	DTVG	0.363	66.3	41.4	53.4

Table 4: Performance on NLG tasks. R-1, R-2, and R-L denote Rouge-1, Rouge-2, and Rouge-L, respectively.

Method	Task			
	RTE	CoLA	CB	WSC
LLama-3.2-1B				
PT	74.8	59.2	60.7	63.5
SPoT	57.6	67.5	64.3	67.3
DTVG	84.1	63.4	82.1	67.3
LLama-3.2-3B				
PT	60.4	67.2	64.3	67.3
SPoT	63.3	71.7	60.7	67.3
DTVG	89.2	73.1	89.3	69.2
LLama-3-8B				
PT	83.5	69.2	57.1	67.3
SPoT	84.9	70.3	60.7	67.3
DTVG	84.9	72.6	86.3	71.4

Table 5: Results on Llama-3.2-1B, Llama-3.2-3B and Llama-3-8B

to Appendix C.4 for experiment details.

Natural Language Generation As shown in Table 4, we observe that DTVG consistently outperforms PT on three natural language generation tasks (namely, E2E (Dušek et al., 2019), CommonGen (Lin et al., 2020), and WebNLG (Gardent et al., 2017)), suggesting DTVG works not only for NLU but also for NLG. Interestingly, although we transfer TPV from NLU tasks to NLG tasks, DTVG's performance on NLG tasks does not degrade, which aligns with the same observation (Wang et al., 2023). We suspect that this phenomenon might be related to T5's text-to-text framework. Please see Appendix C.5 for details.

Generalization to Other LLMs We experimentally analyze the performance of DTVG on the latest decoder-based models using Llama-3.2-1B, Llama-3.2-3B and Llama-3-8B (Dubey et al., 2024). As shown in Table 5, DTVG outperforms vanilla prompt tuning across various target tasks. When compared with SPoT, DTVG demonstrates consistent positive transfer across various LLMs, whereas SPoT exhibits negative transfer, such as on RTE with Llama-3.2-1B (74.8% vs. 57.6%). These results suggest that DTVG's generalizability to other types of LLMs. Moreover, we observe that DTVG performs better on Llama-3.2-3B than Llama-3.2-1B, indicating that it benefits from more powerful LLMs. Please see Appendix C.6 for experiment details.

Dynamic Grouping Figure 7 illustrates the variations of dynamic grouping for RTE during the training process. Compared to prompt tuning, DTVG achieves better performance on RTE.

From the task grouping perspective, we observe the source task combination shifts from [top1: MNLI, top2: SST2] to [top1: MNLI, top2: QNLI] over time. This result suggests that a) **Target Similarity**: two NLI source tasks become more aligned to the target task RTE (NLI); and b) **Knowledge Consistency**: conflicts exist between MNLI and SST2 (replaced by QNLI) are reduced.

From an iterative training perspective, we observe that source task groups fluctuate frequently during the early stages of training. As training progresses, the task group converges, resulting in a stable selection of tasks in the final stage. This supports our hypothesis that insufficient convergence is attributed to the low-resource characteristics of the target tasks. Additionally, we report the grouping results of MRPC, NQ, and SciTail in Appendix H.

5 Conclusion

In this paper, we present DTVG, a novel approach for addressing potential negative transfer in multi-task prompt tuning based on task prompt vectors. Compared to vanilla transfer of the soft prompt from all source tasks, we dynamically group a subset of source tasks and merge their task prompt vectors to avoid an unrelated source task inducing performance degradation of the target task. Extensive

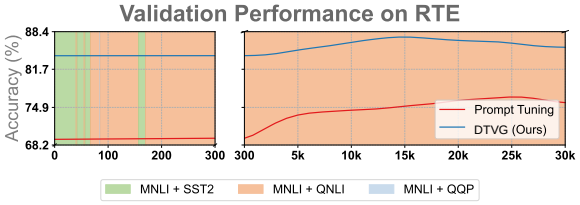

Figure 7: Validation performance on RTE with source task grouping. The source tasks are arranged in each patch legend from left to right, ordered by their similarity to the target task, from highest to lowest.

experiments demonstrate that DTVG effectively groups related source tasks to further optimize the performance of the target task.

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.

Robert Belanec, Simon Ostermann, Ivan Srba, and Maria Bielikova. 2024. Task prompt vectors: Effective initialization through multi-task soft-prompt transfer. *arXiv preprint arXiv:2408.01119*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

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. *arXiv preprint arXiv:1905.10044*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, pages 107–124.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Lingyun Feng. 2023. Learning to predict task transferability via soft prompt. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8829–8844.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 96–120, Online. Association for Computational Linguistics.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea

Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hamish Ivison and Matthew E Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task nlp. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1715–1730.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI conference on artificial intelligence*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1823–1840. Association for Computational Linguistics.

Yu-Chen Lin, Wei-Hua Li, Jun-cheng Chen, and Chu-Song Chen. 2024. Accept: Adaptive codebook for composite and efficient prompt tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15345–15358.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2024. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.

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

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Zhengxiang Shi and Aldo Lipani. 2024. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059.

Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogério Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

A Warstadt. 2019. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Muling Wu, Wenhao Liu, Jianhan Xu, Changze Lv, Zixuan Ling, Tianlong Li, Longtao Huang, Xiaoqing Zheng, and Xuanjing Huang. 2023. Parameter efficient multi-task fine-tuning by learning to transfer token-wise prompts. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 8734–8746. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Frederic Z Zhang, Paul Albert, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. 2024. Knowledge composition using task vectors with learned anisotropic scaling. *arXiv preprint arXiv:2407.02880*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308.

Appendix

A Performance and Parameter Comparison

We visualize the average score (y-axis) and parameter (x-axis) on the GLUE benchmark across various baselines in Figure 8. We observe DTVG surpassing other baselines and achieving SOTA performance with minimal parameters.

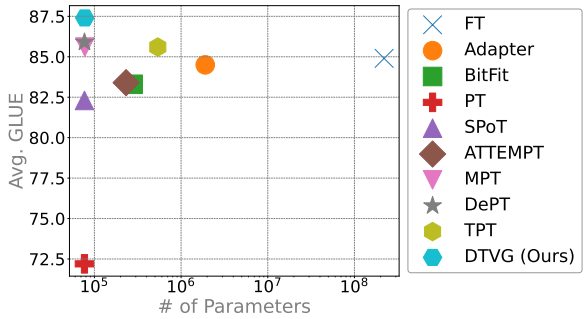

Figure 8: Performance & parameter comparison.

B Dataset Details

We use 26 datasets in total from 5 benchmarks. We use GLUE and SuperGLUE benchmarks to test the model's natural language understanding. MNLI (Williams et al., 2018), QNLI (Demszky et al., 2018), QQP (Wang, 2018), SST2 (Socher et al., 2013), RTE (Giampiccolo et al., 2007), CoLA (Warstadt, 2019), STS-B (Cer et al., 2017), MRPC (Dolan and Brockett, 2005) are derived from GLUE. MultiRC (Khashabi et al., 2018), BoolQ (Clark et al., 2019), WiC (Pilehvar and Camacho-Collados, 2019), WSC (Levesque et al., 2012), and CB (De Marneffe et al., 2019), ReCoRD (Zhang et al., 2018) are from SuperGLUE. We use four question-answering datasets from the MRQA 2019 benchmarks, including Natural Questions (NQ) (Kwiatkowski et al., 2019), HotpotQA (HQ) (Yang et al., 2018), NewsQA (News) (Trischler et al., 2017), and SearchQA (SQA) (Dunn et al., 2017), SQuAD (Rajpurkar, 2016). WinoGrande (WG) (Sakaguchi et al., 2021), YelpPolarity (Yelp) (Zhang et al., 2015), SciTail (Khot et al., 2018) and PAWS-Wiki (PAWS) (Zhang et al., 2019) are from the 'other' benchmark to test model' generalizability across different domains. We also use CommonGen (Lin et al., 2020), E2E (Dušek et al., 2019), and WebNLG (Gardent et al., 2017) sourced from the GEM (Gehrmann et al., 2021) benchmark to test the model's performance on natural language generation. We download all datasets from the huggingface dataset[1]. Table 6 lists more details about each dataset.

C Implementation Details

We use PyTorch[2], huggingface transformers[3] to implement our method. We validate the effectiveness of DTVG based on the open-source repository [4]. All of the experiments are conducted with a single GPU with 32 GB of memory. Following Asai et al. (2022), we use the original T5 checkpoint. We set the batch size for T5-base as 32 for most datasets. We set the batch size to 16 and the gradient accumulation step to 2 for the MRQA benchmark with a long context. Due to the different input lengths of various datasets, we set the maximum token length of 256 for most datasets that have a context of fewer than 200 tokens. We set the maximum token length of 348 for MultiRC and 512 for MRQA datasets. We limit the maximum training data number of YelpPolarity to 100k. We maintain the same hyperparameter settings (Lester et al., 2021) to reinitialize and retrain all tasks, aiming to reconstruct the corresponding soft prompts and task prompt vectors. Similar to (Mahabadi et al., 2021), for datasets lacking publicly available test sets, we use the validation set as the test set or partition it to create separate test and validation sets.

C.1 Comparison of Task Prompt Vectors and Soft prompt

We used the reconstructed soft prompts with the same initialization to compare SPoT (Vu et al., 2022) and TPV. Specifically, we initialize the target task prompt with the soft prompt that obtained the highest metric score from six source tasks (namely, MNLI, QNLI, QQP, SST-2, ReCoRD, and SQuAD). Note that the difference between the implementations of the two methods SPoT and TPV is only in the task similarity metric. SPoT uses the traditional cosine similarity of soft prompts, while TPV uses Eqn.2 to compute task similarity.

C.2 Full-dataset Transfer

We set warmup steps to be 500, weight decay to be $1 * 10^{-5}$, and use Adam (Kingma and Ba, 2015) for optimization with a linear learning rate scheduler.

[1] https://github.com/huggingface/datasets
[2] https://pytorch.org/
[3] https://github.com/huggingface/transformers
[4] https://github.com/AkariAsai/ATTEMPT

Dataset	Source	Target	Benchmark	Task Type	Domain	Metric
MNLI	✓	✓	GLUE	Natural Language Inference	Various	Accuracy
QQP	✓	✓	GLUE	Paraphrase Detection	Social QA	Accuracy & F1
QNLI	✓	✓	GLUE (QA)	Natural Language Inference	Wikipedia	Accuracy
SST2	✓	✓	GLUE	Sentiment Analysis	Movie Reviews	Accuracy
STS-B	×	✓	GLUE	Sentence Similarity	Various	Pearson & Spearman corr.
MRPC	×	✓	GLUE	Paraphrase Detection	News	Accuracy & F1
RTE	×	✓	GLUE	Natural Language Inference	News & Wikipedia	Accuracy
CoLA	×	✓	GLUE	Acceptability	Various	Matthews corr.
ReCoRD	✓	×	SuperGLUE	Question Answering (QA)	News	F1 & EM
MultiRC	×	✓	SuperGLUE	Question Answering (QA)	Various	F1 & EM
BoolQ	×	✓	SuperGLUE	Question Answering (QA)	Wikipedia	Accuracy
WiC	×	✓	SuperGLUE	Word Sense Disambiguation	Lexical databases	Accuracy
WSC	×	✓	SuperGLUE	Common Sense Reasoning	Fiction books	Accuracy
CB	×	✓	SuperGLUE	Natural Language Inference	Various	Accuracy
SQuAD	✓	×	MRQA 2019	Question Answering (QA)	Wikipedia	F1 & EM
NQ	×	✓	MRQA 2019	Question Answering (QA)	Wikipedia	F1 & EM
HotpotQA	×	✓	MRQA 2019	Question Answering (QA)	Wikipedia	F1 & EM
SearchQA	×	✓	MRQA 2019	Question Answering (QA)	Search snippets	F1 & EM
NewsQA	×	✓	MRQA 2019	Question Answering (QA)	News	F1 & EM
WinoGrande	×	✓	'Other'	Common Sense Reasoning	WikiHow	Accuracy
YelpPolarity	×	✓	'Other'	Sentiment Analysis	Yelp reviews	Accuracy
SciTail	×	✓	'Other'	Natural Language Inference	Science exams	Accuracy
PAWS	×	✓	'Other'	Paraphrase Detection	Wikipedia	Accuracy
WebNLG	×	✓	GEM	Data to Text (NLG)	Various	Automated Evaluation
E2E	×	✓	GEM	Data to Text (NLG)	Restaurant	Automated Evaluation
CommonGen	×	✓	GEM	Data to Text (NLG)	Commonsense	Automated Evaluation

Table 6: Details about 26 datasets from 5 Benchmarks in total. GLUE (QA) denotes the QNLI derived from the Question Answering Dataset (SQuAD). Lexical databases contain WordNet, VerbNet, and Wiktionary, Search snippets denote question answering from the search engine. Automated Evaluation includes BLEU, Rouge-1, Rouge-2, and Rouge-L. Following Shi and Lipani (2024), we use the metric marked with an underline as the primary evaluation metric.

C.3 Few-shot Adaptation

In few-shot adaptation experiments, followed by (Mahabadi et al., 2021), we run experiments three times with different random seeds and take the mean of the performance. In each trial, we train 1k steps on the target task for both task prompt vector learning and multi-task prompt transfer stage, which we found to be able to achieve full convergence. We evaluate every 50 steps on the original validation set. For the rest, we report on the original test sets based on the best checkpoint on the validation set.

C.4 Model Scale

For model scaling experiments, we set the batch sizes are 100 and 16 for T5-small and T5-large, respectively.

C.5 Other LLMs

We use Llama-3.2-1B, Llama-3.2-3B and Llama-3-8B to test DTVG's generalizability on other types of LLMs. In our experiment, we use the same 6 source tasks as our main experiments setting on the T5-base and select RTE, CoLA, CB, and WSC as target tasks. We set the length of the soft prompt to 100 for both models and set the batch size to 16, 4 and 2 for Llama-3.2-1B, Llama-3.2-3B and Llama-3-8B, respectively. Compared to encoder-decoder-based models, we observe that decoder-based autoregressive models require a smaller learning rate. Therefore, we set the learning rate of the soft prompt and its corresponding scaling term to 0.001 for Llama-3.2-1B and Llama-3.2-3B, and to 0.0001 for Llama-3-8B.

C.6 Natural Language Generation

We select E2E, CommonGen, and WebNLG sourced from the GEM benchmark to evaluate DTVG's performance on natural language generation (NLG) tasks. We use T5-base as the backbone and reuse the task prompt vectors sourced from 6 natural language understanding (NLU) source tasks. We set the maximum 128 token length for both the input and output. We use the target as a simple reference to compute metrics for both PT and DTVG and report the best result on the valida-

tion set in Table 4.

C.7 Two Speed Learning Rate

For the full-dataset transfer setting, we search the learning rate within the set {3e-1, 4e-1, 5e-1} for the target task prompt and corresponding scaling term. For the scaling term of the source prompt task vectors, we search the learning rate within the set {4e-1, 6e-1, 8e-1, 1}. For few-shot adaptation and others, we set the learning rate of 0.3 for both the target task prompt and corresponding scaling term, and 0.4 for the scaling term of the source task prompt vectors.

C.8 Prompt Initialization

We initialized the soft prompt by randomly sampling the top 5000 vocabulary words for all tasks. In both full-dataset transfer and few-shot adaptation experiments, we utilize soft prompt tasks vectors from source tasks by full-dataset prompt tuning. In few-shot adaptation setting, we exclude the corresponding task prompt vectors when adapt to source tasks in GLUE.

D MRQA and 'Other' Benchmark

As shown in Table 7, DTVG realizes significant improvements over the vanilla prompt tuning with a 3.7%, 14.2% increase on MRQA and 'Other' in terms of relative average performance. Compared to other baselines, DTVG also achieves comparable or better performance on MRQA and 'other' benchmarks.

E Few-shot adaptation On GLUE and SuperGLUE benchmark

We compare our method with no transfer baseline PT, one for one baseline DePT, and all for one baseline MPT, and Table 8 shows the evaluation results on GLUE and SuperGLUE benchmarks. Our method can substantially improve the few-shot adaptation results in the most of settings. Specifically, compared to PT, our method on average improves the results across only (k= 4,16,32) shots. Meanwhile, our method also surpasses MPT and DePT, in terms of performance.

F Algorithm Details about DTVG

We give all the implementation details about DTVG for multi-task prompt tuning on Algorithm 1.

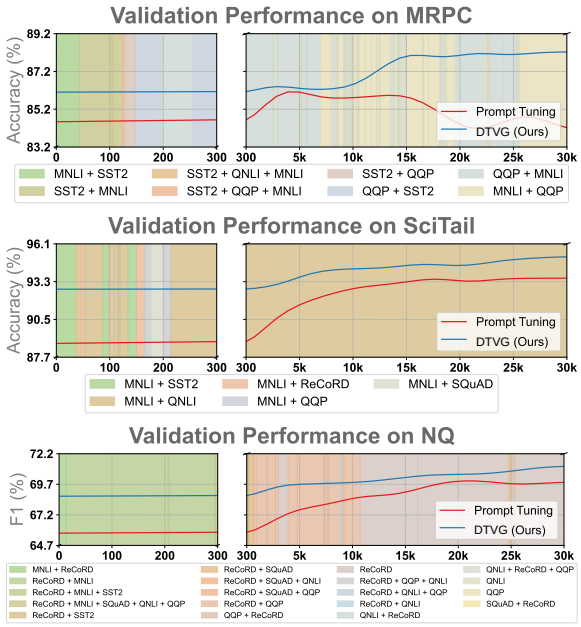

Figure 9: Validation Performance MRPC, SciTail, and NQ with source task grouping.

G Details about Source Task Grouping

Implementation details about Source Task Grouping for addressing optimization objective 5 are presented in Algorithm 2.

H Source Task Grouping

Figure 9 demonstrates that DTVG consistently outperforms vanilla prompt tuning in terms of performance across the MRQA, SciTail, and NQ datasets. Meanwhile, DTVG achieves dynamic grouping a appropriate task subset for the different target task. This indicates that DTVG is capable of effectively group a related source task combination tailored to different target tasks, thereby reduce negative transfer.

I Task similarity

TPV represents the change in parameters after fine-tuning from its initial parameters on a specific task and reflects the specific optimization direction of a task in the weight space. When we fix a unified initialization for all tasks, effectively constraining them to the same weight space, it means that when two TPVs are closer, their optimization directions are more aligned. As a result, when transferring between tasks, there will be fewer conflicts.

We conduct a case study using 5 source tasks and 6 target tasks with the same initialization to analyze the effectiveness of TPV in capturing the relation-

Method	param \ task	MRQA					Other				
		NQ (100k)	HP (72K)	SQA (117K)	News (74K)	Avg.	WG (40K)	Yelp (100k)	SciTail (27K)	PAWS (49K)	Avg.
Finetuning$_1$	220M	75.1	77.5	81.1	65.2	74.7	61.9	96.7	95.8	94.1	87.1
Adapters$_1$	1.9M	74.2	77.6	81.4	65.6	74.7	59.2	96.9	94.5	94.3	86.2
BitFit$_1$	280K	70.7	75.5	77.7	64.1	72.0	57.2	94.7	94.7	92.0	84.7
PT$_1$	76.8K	67.9	72.9	75.7	61.1	69.4	49.6	95.1	87.9	55.8	72.1
LoRA$_3$	3.8M	72.4	62.3	72.5	56.9	66.0	58.2	97.1	94.7	94.0	86.0
SPoT$_1$	76.8K	68.2	74.8	75.3	58.2	69.1	50.4	95.4	91.2	91.1	82.0
ATTEMPT$_1$	232K	70.4	75.2	77.3	62.8	71.4	57.6	96.7	93.1	92.1	84.9
MPT$_2$	77.6k	$72.0_{0.1}$	$75.8_{0.1}$	$77.2_{0.1}$	$63.7_{0.1}$	72.2	$56.5_{0.9}$	$96.4_{0.0}$	$95.5_{0.1}$	$93.5_{0.1}$	85.5
DePT$_3$	76.8k	$73.2_{0.1}$	$76.8_{0.3}$	$77.6_{0.2}$	$64.4_{0.1}$	73.0	$59.0_{0.2}$	$96.8_{0.1}$	$95.6_{0.2}$	$93.7_{0.1}$	86.3
DTVG (ours)	77.5k	$73.1_{0.1}$	$76.7_{0.0}$	$77.8_{0.3}$	$64.6_{0.1}$	73.1	$58.0_{0.0}$	$96.6_{0.1}$	$97.0_{0.1}$	$93.7_{0.0}$	86.3

Table 7: Performance on MRQA2019 and 'Other' benchmarks. "param \ task" denotes the number of learnable parameters for each task. $_1$ sourced from (Asai et al., 2022), $_2$ sourced from (Wang et al., 2023) and $_3$ sourced from (Shi and Lipani, 2024).

Method	k-shot	GLUE									SuperGLUE					
		MNLI	QQP	QNLI	SST2	STS-B	MRPC	RTE	CoLA	Avg.	Multi	Bool	WiC	WSC	CB	Avg.
PT	4	40.1	63.2	40.4	53.0	88.8	68.1	56.3	27.4	54.7	61.8	61.6	51.2	60.4	53.5	57.7
MPT		59.4	82.0	86.2	56.5	89.1	68.1	62.6	34.8	67.3	62.2	62.2	52.9	67.3	73.6	63.6
DePT		$44.0_{1.1}$	$77.4_{6.7}$	$85.8_{4.4}$	$59.3_{3.1}$	$84.1_{2.7}$	$73.5_{2.8}$	$63.5_{2.8}$	$29.3_{2.3}$	64.6	$62.3_{1.3}$	$62.7_{5.4}$	$57.5_{1.1}$	$67.9_{0.9}$	$75.0_{5.1}$	65.1
Our		$49.3_{1.7}$	$87.5_{0.7}$	$80.2_{0.3}$	$81.8_{1.9}$	$87.9_{0.5}$	$68.1_{0.0}$	$72.7_{0.7}$	$22.2_{4.5}$	68.7	$61.4_{0.2}$	$60.6_{1.5}$	$59.4_{1.7}$	$45.2_{1.0}$	$86.9_{1.7}$	62.7
PT	16	41.5	62.3	87.4	50.9	87.8	68.1	54.7	28.5	56.7	60.3	61.9	48.9	44.2	63.5	55.8
MPT		61.6	84.7	90.6	63.2	89.1	70.1	64.8	32.1	69.5	64.5	63.3	49.8	67.3	78.6	64.7
DePT		$61.8_{2.5}$	$80.3_{1.3}$	$91.2_{0.5}$	$77.6_{6.3}$	$87.1_{1.7}$	$78.1_{2.3}$	$71.9_{1.0}$	$27.1_{1.7}$	71.9	$60.6_{2.8}$	$66.9_{4.4}$	$59.6_{0.7}$	$57.7_{2.7}$	$78.6_{4.3}$	64.7
Our		$58.8_{0.6}$	$81.9_{1.2}$	$89.8_{1.1}$	$84.6_{1.1}$	$88.4_{0.4}$	$86.9_{0.4}$	$76.8_{1.0}$	$31.3_{2.3}$	74.8	$61.4_{3.1}$	$72.3_{1.4}$	$60.7_{0.4}$	$67.3_{0.0}$	$82.1_{2.9}$	68.8
PT	32	37.0	62.3	56.7	50.9	87.5	68.1	54.7	23.2	55.1	59.2	61.7	52.6	67.3	67.8	61.7
MPT		63.6	88.5	91.0	75.9	89.7	74.5	59.7	30.8	71.7	63.3	68.9	53.9	67.3	82.1	67.1
DePT		$63.3_{3.5}$	$80.1_{0.7}$	$91.3_{0.5}$	$80.4_{8.7}$	$89.2_{0.1}$	$81.4_{3.3}$	$72.7_{2.9}$	$28.6_{2.1}$	73.4	$60.1_{2.7}$	$67.2_{3.4}$	$58.0_{0.7}$	$63.1_{3.6}$	$82.1_{2.3}$	66.4
Our		$61.2_{0.1}$	$85.3_{0.8}$	$91.2_{0.1}$	$88.3_{1.4}$	$83.2_{4.7}$	$83.1_{4.7}$	$74.1_{2.7}$	$29.3_{1.5}$	74.5	$66.3_{6.1}$	$73.5_{1.1}$	$60.2_{1.0}$	$67.3_{0.0}$	$84.5_{1.7}$	70.4

Table 8: Few-shot adaptation on GLUE and SuperGLUE benchmark

Algorithm 1: DTVG

Input: source tasks set $\mathcal{S} = \{s^1, s^2, \ldots, s^n\}$, target task t, initialization soft prompt parameters P_{init}, maximum training steps $N_{\max}$

Output: Trained multi-task soft prompt parameters P^*_{mix}

Stage 1: Task prompt vector Learning ;

Initialize P_{init} for both sources and target task;

Boost the posterior probability and obtain their task prompt vectors;

Stage 2: Multi-task Prompt Transfer ;

for *each iterative* $k \leftarrow 1$ ***to*** $N_{\max}$ **do**

  Source Task Grouping: Group a subset of relevant source tasks $\mathcal{S}'$ from $\mathcal{S}$;

  Multi-Task Merging: Merge task prompt vectors from $\mathcal{S}' \cup \{t\}$ to get P_{mix} ;

  Boost the posterior probability on target task t with P_{mix}

return P^*_{mix}

Algorithm 2: Source Task Grouping

Input: source tasks set $\mathcal{S} = \{s^1, s^2, \ldots, s^n\}$, target task t
Output: selected task group $\mathcal{S}'$

Step 1: Rank Similarity to Target ;
Compute Similarity Ranking list Π;
Step 2: Maximize Knowledge Consistency ;
Initialize an empty source task group $\mathcal{S}' \leftarrow \emptyset$;
for *each index* π^i *from similarity rank list* Π **do**
    Let s^{π^i} be the task corresponding to index π^i;
    Calculate the contribution of s^{π^i} to $\mathcal{S}'$: $\Delta(\mathcal{S}', s^{\pi^i}) \leftarrow KC(\mathcal{S}' \cup \{s^{\pi^i}\}) - KC(\mathcal{S}')$;
    if $sim(t, s^{\pi^i}) \geq 0$ ***and*** $\Delta(\mathcal{S}', s^{\pi^i}) \geq 0$ **then**
        Add s^{π^i} to $\mathcal{S}'$: $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{s^{\pi^i}\}$;
return $\mathcal{S}'$

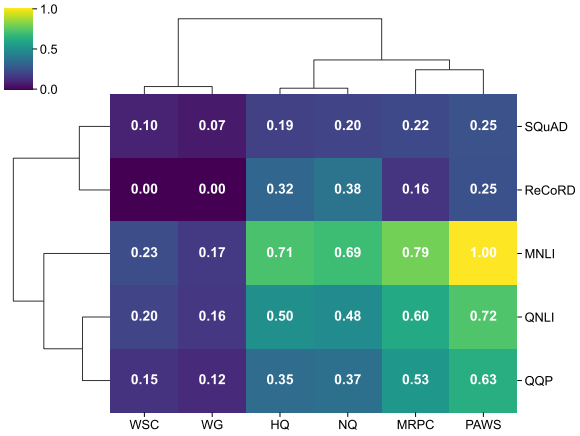

Figure 10: Task similarity of dot product result via TPV. We visualize the task similarity between 5 source tasks and 6 target tasks. We apply min-max normalization to reflect the relative relation among tasks.

ships between different tasks. Figure 10 shows the cluster map by computing pairwise task similarity score based on TPV (Eqn. 2). We observe that tasks perceived as similar are clustered together. Specifically, in the source tasks partition, SQuAD and ReCoRD are grouped in the QA cluster. QNLI and QQP belong to QA datasets. This clustering pattern is also observed in the target tasks partition. NQ and HQ are in the QA cluster, MRPC PAWS are Paraphrase Detection, and WSC and WG are Common Sense Reasoning. Furthermore, all target tasks show a consistently high relative task similarity with MNLI, a widely used intermediate task for fine-tuning PLMs (Phang et al., 2018). This highlights the TPV's ability to capture less obvious positive transfer. More details can be found in Figure 11.

J Computation and Time Costs

Dynamically calculating the task combinations during each parameter update does indeed introduce additional time and computation costs during training. However, this computation does not involve gradients, so it ultimately does not lead to a significant increase in time and computation burden. We visualize the result and training speed on RTE in Table 9. We observe that DTVG demonstrates a 16.5% improvement in performance while incurring only a 8.7% decrease in training speed compared to prompt tuning.

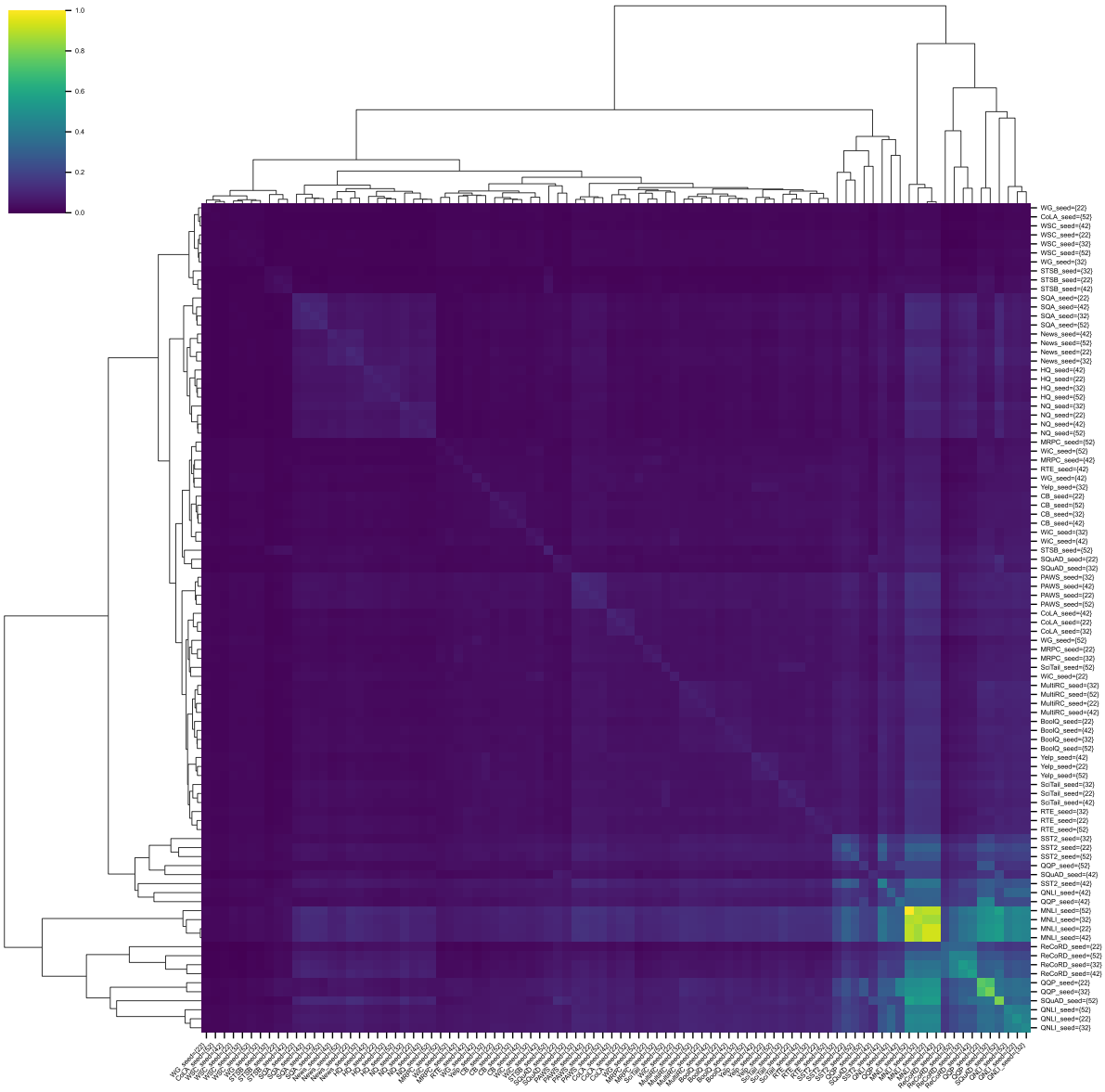

Figure 11: Task similarity visualizations of task prompt vectors. We conduct four experiments with different seeds in $\{22, 32, 42, 52\}$. We apply Min-Max normalization to ensure the relative relationships in the results are maintained.

Method	Test Acc on RTE	Traning samples per second
PT	74.1	64.2
DTVG	86.3	58.6

Table 9: Test result and training speed on RTE. We use T5-base as backbone