# MULTITASK PROMPT TUNING ENABLES PARAMETER-EFFICIENT TRANSFER LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Prompt tuning, in which a base pretrained model is adapted to each task via conditioning on learned prompt vectors, has emerged as a promising approach for the efficient adaptation of large language models to multiple downstream tasks. However, existing methods typically learn soft prompt vectors from scratch, and it has not been clear how to exploit the rich cross-task knowledge in task-specific prompt vectors to improve performance on target downstream tasks. In this paper, we propose multitask prompt tuning (MPT), which first learns a single transferable prompt by decomposing and distilling knowledge from multiple task-specific source prompts. We then learn multiplicative low rank updates to this shared prompt to efficiently adapt it to each downstream target task. Extensive experiments on 21 NLP datasets demonstrate that our proposed approach outperforms the state-of-the-art methods, including the full finetuning baseline in some cases, despite only tuning $0.035\%$ as many task-specific parameters.

## 1 INTRODUCTION

Finetuning Pretrained Language Models (PLMs) has led to significant improvements across various downstream NLP tasks (Devlin et al., 2019; Howard & Ruder, 2018; Raffel et al., 2020). However, the conventional paradigm of full task-specific finetuning is difficult to scale to multiple tasks given that contemporary PLMs can have hundreds of millions (or even billions) of parameters. There has thus been a growing interest in developing *parameter-efficient* methods for model tuning (Houlsby et al., 2019; Lester et al., 2021; Ding et al., 2022), where the goal is to learn only a small number of additional parameters per task while achieving performance comparable to full model finetuning.

Prompt tuning (PT), which prepends continuous prompt vectors to the input, has emerged as a promising approach for parameter-efficient transfer learning with PLMs (Liu et al., 2021a; Li & Liang, 2021; Lester et al., 2021; Liu et al., 2022b; 2021b). PT freezes the PLM parameters and only learns a small set of task-specific prompt vectors. Despite their impressive performance, there is still a large gap between prompt tuning and full finetuning for many models and tasks (Lester et al., 2021). Prompt vectors trained using task-specific training data only are more sensitive to initialization and require significantly more training time than finetuning (Su et al., 2022; Zhong et al., 2022).

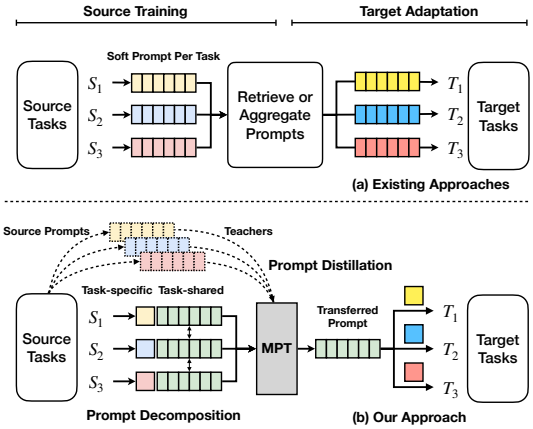


**Figure 1: A conceptual overview of our approach.** Instead of retrieving or aggregating source prompts, MPT learns a single transferable prompt exploiting rich cross-task shared knowledge. The transferable prompt is learned via prompt decomposition and distillation to enable parameter-efficient transfer learning with PLMs.

Recent work has proposed to address these issues through *transferring* prompt vectors from different tasks (Su et al., 2022; Zhong et al., 2022). These methods first train soft prompts on multiple source tasks and then use these pretrained prompts to initialize the prompt for further finetuning on a target task based on a (potentially learned) similarity measure (Vu et al., 2022; Asai et al., 2022) (see Figure 1, top). In this paper, we

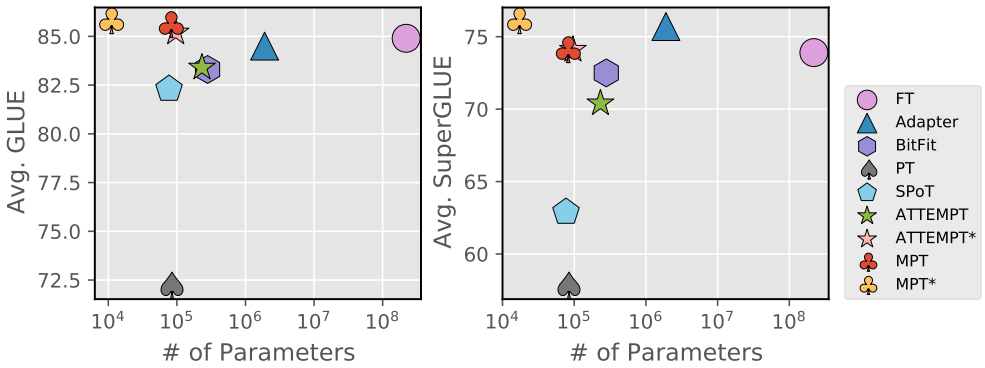


**Figure 2: Parameter efficiency on GLUE and SuperGLUE.** All results are based on T5-Base model (Raffel et al., 2020). Adapter (Houlsby et al., 2019), BitFit (Zaken et al., 2022), PT (Lester et al., 2021), SPoT (Vu et al., 2022), ATTEMPT (Asai et al., 2022). * indicates multitask training on target tasks. Our MPT approach—which transfers a single shared prompt learned from multiple source tasks using prompt decomposition and distillation—outperforms all the existing prompt tuning methods and full model finetuning (FT), despite updating much fewer task-specific parameters. Best viewed in color.

extend this line of work and introduce *multitask prompt tuning* (MPT), which uses multitask data to learn a *single* prompt that can be efficiently transferred to target tasks. While conceptually simple, learning a shared prompt space can be practically challenging as it requires learning commonalities across source tasks while minimizing interference. We decompose the soft prompt of each source task (which can be represented as a prompt matrix) as a multiplication of a shared matrix and a low-rank task-specific matrix, and find that this decomposition is more effective than simply sharing the prompt matrix across all tasks. This decomposition is learned through knowledge distillation from soft prompts obtained from regular prompt tuning. To transfer to new tasks, we perform low-rank multiplicative updates to the shared prompt matrix. Figure 1 (bottom) illustrates our approach.

Extensive experiments on 21 NLP datasets across diverse tasks demonstrate the effectiveness of our proposed approach over state-of-the-art prompt transfer methods. On the SuperGLUE benchmark (Wang et al., 2019), MPT with T5-Base (Raffel et al., 2020) yields a 16.3 improvement over the vanilla prompt tuning baseline (PT, Lester et al., 2021), and also outperforms the most competitive multitask prompt transfer baseline (ATTEMPT, Asai et al., 2022) despite tuning much fewer task-specific prompt parameters (77.6K vs 232K). On some benchmarks, MPT exceeds the performance of full finetuning while only requiring $0.035\%$ tunable parameters per task (see Figure 2). We also discover that MPT is very effective for few-shot learning with 4-32 labels. Finally, ablation studies further show that MPT matches the performance of full finetuning at different model scales ranging from 60M to 770M parameters. Our code and models will be made publicly available.

## 2 Related Work

**Parameter-Efficient Transfer Learning.** Parameter-efficient transfer learning for pretrained language models is an active research area (Ding et al., 2022). Adapters (Houlsby et al., 2019; Mahabadi et al., 2021) and variants (Hu et al., 2021; Karimi Mahabadi et al., 2021) insert trainable layers, while BitFit (Zaken et al., 2022) only updates the bias parameters without changing any other model parameters. Diff pruning (Guo et al., 2021) and FISH (Sung et al., 2021) learn sparse updates to the original PLM. Another popular choice is prompt tuning (Lester et al., 2021) which only updates soft prompts prepended to the input. Prefix-tuning for optimizing continuous prompts for natural language generation tasks is presented in Li & Liang (2021). UNIPELT learns to combine different tuning methods via gating mechanism (Mao et al., 2022). HyperPrompt (He et al., 2022) introduces task-conditioned hyperprompts that conditions the model on task-specific information for constructing prompts. Discrete (i.e., hard) prompts have also been shown to be effective in many cases (Schick & Schütze, 2021a;b; Gao et al., 2021; Malkin et al., 2022). However, our approach is most related to the transferability of prompts (Wang et al., 2021; Vu et al., 2022; Su et al., 2022), which focuses on boosting the performance of prompt tuning across many tasks. SPoT (Vu et al., 2022) selects one prompt using a similarity measure and ATTEMPT (Asai et al., 2022) adopts an attention mechanism over the source prompts to initialize the prompt for a target task. Unlike existing

works, our approach learns a single shared prompt by decomposing and distilling knowledge from source prompts in a structured way for efficient adaptation to a diverse set of target tasks.

**Multitask Learning.** Multitask learning, which focuses on simultaneously solving multiple related tasks, has been studied from multiple perspectives (Zhang & Yang, 2021; Ruder, 2017). Transferring a model fine-tuned on multiple source tasks to another target task is a common approach to multitask learning (Vu et al., 2020; Raffel et al., 2020; Aghajanyan et al., 2021a; Zhong et al., 2021; Clark et al., 2019b). Few recent works show zero-shot and few-shot transfer capabilities of language models through massive multitask learning over a large number of tasks (Sanh et al., 2022; Wang et al., 2022; Liu et al., 2022a; Wei et al., 2021). Designing specific parameter-sharing strategies is also another recent trend in multitask learning (Ruder et al., 2019; Sun et al., 2020; Misra et al., 2016). While our proposed approach is inspired by these methods, in this paper we focus on multi-task prompt transfer for parameter-efficient adaptation of language models, which still remains as a challenging and largely under-addressed problem.

**Knowledge Distillation.** Knowledge distillation has been used to improve performance and efficiency across many tasks (Gou et al., 2021), including model compression (Hinton et al., 2015; Jiao et al., 2020; Sanh et al., 2019), transfer learning (Furlanello et al., 2018; Xu et al., 2020), machine translation (Zhou et al., 2019), question answering (Hu et al., 2018), and document retrieval (Shakeri et al., 2019). Concurrent to our work, PANDA (Zhong et al., 2022) uses knowledge distillation with a new metric to better predict the prompt transferability across different combinations of source-target tasks. This differs from MPT, which uses a prompt decomposition strategy to leverage commonalities across the source tasks while minimizing interference between them. In addition, PANDA focuses on transferring from one source task to another target task using a similarity measure (similar to SPoT (Vu et al., 2022)), while our MPT approach leverages multitask learning to better exploit the rich cross-task knowledge in prompt transfer.

## 3 METHODOLOGY

Given a set of source tasks $\boldsymbol{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_\kappa\}$ and target tasks $\boldsymbol{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_\tau\}$, our goal is to learn a single soft prompt over $\boldsymbol{S}$ that can be efficiently updated to enable better performance on $\boldsymbol{T}$. Simply training a single soft prompt on $\boldsymbol{S}$ is sub-optimal as it can fail to leverage commonalities across source tasks while minimizing interference. To this end, our multitask prompt tuning (MPT), aims to efficiently compress task-shared knowledge in $\boldsymbol{S}$ into a single prompt $\phi_s$ to improve performance on $\boldsymbol{T}$ while filtering out task-specific information that is less useful for transfer learning.

**Prompt Tuning.** Given a pre-trained language model with parameters $\Theta$ and one target task $\mathcal{T}$ with training data $(\boldsymbol{X}, \boldsymbol{Y}) = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$, directly finetuning all the parameters by maximizing the conditional probability $P(\boldsymbol{Y}|\boldsymbol{X};\Theta)$ is expensive and often tends to overfit on small datasets. An alternative to finetuning that is more parameter-efficient is prompt tuning (PT), which randomly initializes a small number of learnable prompt vectors (i.e., soft prompts) to be prepended to the input embeddings of the PLM while freezing model parameters $\Theta$ (Lester et al., 2021; Liu et al., 2022b). Formally, for a sequence of input tokens with token embeddings as $\boldsymbol{x} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_n\} \in \mathbb{R}^{n \times d}$, PT prepends the soft prompt $\boldsymbol{P} \in \mathbb{R}^{l \times d}$ with the same dimension as the token embedding $d$ and vector length as $l$. Then PT optimizes the following loss function:

$$\mathcal{L}_{\text{PLM}} = -\sum_{i} \log P(\boldsymbol{y}_i \,|\, [\boldsymbol{P}; \boldsymbol{x}_i]\,; \Theta\,), \tag{1}$$

with respect to $\boldsymbol{P}$. While this approach has been successful on some tasks and models, researchers have observed that vanilla PT can sometimes lead to lower performance (especially on smaller PLMs), be slow to converge, and have high sensitivity to the initialization (Lester et al., 2021; Su et al., 2022; Zhong et al., 2022). Recent works address these issues by first training prompts on multiple source tasks, and then initializing the prompts for a target task via some similarity measure or learned attention (Asai et al., 2022; Vu et al., 2022). We extend this line of work and propose a novel framework for transferring multitask knowledge into a single soft prompt to enable more performant and parameter-efficient transfer learning to downstream target tasks $\boldsymbol{T}$.

### 3.1 MULTITASK PROMPT TUNING

Our proposed framework mainly consists of two stages, *source training* and *target adaptation*. The proposed MPT framework first focuses on the source training to generate a single soft prompt to

be reused in the second stage for target task adaptation. Specifically, task prompts for source tasks are decomposed into a task-shared component and a low-rank task-specific component (*prompt decomposition*), where the former is shared across all tasks, and the latter is task-specific. We also use prompt distillation to better transfer multitask knowledge to the shared component by distilling knowledge from multiple tasks-specific source prompts. Once learned, the shared prompt matrix is adapted to a downstream target task via low-rank multiplicative updates.

**Prompt Decomposition.** The goal of prompt decomposition is to enable efficient knowledge sharing across $\mathcal{S}$ while still allowing each task to maintain its own parameters to encode task-specific knowledge. Specifically, we decompose the soft prompt $\boldsymbol{P}_k$ for $k$-th task into two parts, as shown in Figure 3. Let $\boldsymbol{P}^* \in \mathbb{R}^{l \times d}$ denote the shared prompt across all tasks, and further let $\boldsymbol{u}_k \in \mathbb{R}^l, \boldsymbol{v}_k \in \mathbb{R}^d$ be the task-specific vectors for each task $k$. The task-specific vectors form a rank-one matrix $\boldsymbol{W}_k = \boldsymbol{u}_k \cdot \boldsymbol{v}_k^T$, which has the same dimensions as the shared prompt $\boldsymbol{P}^*$. The final task prompt $\widehat{\boldsymbol{P}}$ for $k$-th source task is then parameterized as:

$$\widehat{\boldsymbol{P}}_k = \boldsymbol{P}^* \circ \boldsymbol{W}_k = \boldsymbol{P}^* \circ (\boldsymbol{u}_k \cdot \boldsymbol{v}_k^T) \tag{2}$$

where $\circ$ denotes the Hadamard product between two matrices. Our parameterization of prompt decomposition is inspired by prior low-rank methods (Li et al., 2018; Aghajanyan et al., 2021b; Wen et al., 2020), such that general information of $\mathcal{S}$ can be captured by "slow" weights $\boldsymbol{P}^*$ shared across tasks and "fast" weights $\boldsymbol{W}_k$ could encode task-specific knowledge in a low-rank subspace.

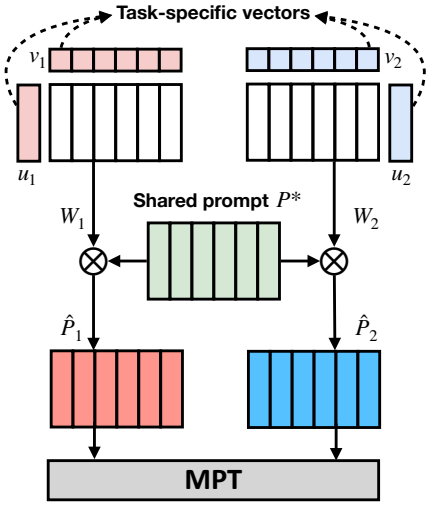


**Figure 3:** An illustration on prompt decomposition for two tasks.

**Prompt Distillation.** Simply using the mixed golden labels to guide the prompt decomposition directly can bias the shared component to overfit to larger tasks. We found knowledge distillation from separately-trained source prompts to be an effective strategy for learning good decomposable prompts. Specifically, we first obtain a teacher prompt $\boldsymbol{P}_k^t$ for the $k$-th source task by conventional prompt tuning. We then randomly initialize a corresponding student prompt as $\widehat{\boldsymbol{P}_k^s} = \boldsymbol{P}^* \circ (\boldsymbol{u}_k \cdot \boldsymbol{v}_k^T)$, where all student prompts share $\boldsymbol{P}^*$ and have their own task-specific vectors as described above. Following Sanh et al. (2019), we design distillation losses to transfer cross-task knowledge into the shared prompt matrix. The first loss is to match the output probability distributions of students and teachers through minimizing their KL-Divergence,

$$\mathcal{L}_{\text{Logits}} = \sum_k \sum_{i \in \mathcal{S}_k} \text{KL}(P(\boldsymbol{y}_i | [\boldsymbol{P}_k^t; \boldsymbol{x}_i]), P(\boldsymbol{y}_i | [\widehat{\boldsymbol{P}_k^s}; \boldsymbol{x}_i])). \tag{3}$$

We further use a temperature $T$ to control the smoothness of the output distribution for both teacher and student models as $p_j = \frac{1}{Z}\exp(z_j/T)$, where $z_i$ is the logit score for class $j$ and $Z$ is the normalization factor. We also have an additional mean squared loss on teacher model hidden states,

$$\mathcal{L}_{\text{Hidden}} = \sum_k \sum_{i \in \mathcal{S}_k} (\boldsymbol{H}_{ki}^s - \boldsymbol{H}_{ki}^t)^2, \tag{4}$$

where $\boldsymbol{H}_{ki}^t, \boldsymbol{H}_{ki}^s$ denotes the hidden states of teacher and student networks, respectively, consisting of a sequence of hidden vectors for $i$-th input. Such additional distillation loss from intermediate states has been shown to improve results in distilling PLMs (Jiao et al., 2020; Shleifer & Rush, 2020). Finally, our total loss function to train student source prompts for obtaining a single shared prompt to be transferred to the target side is formulated as follows:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{PLM}} + \lambda(\mathcal{L}_{\text{Logits}} + \mathcal{L}_{\text{Hidden}}), \tag{5}$$

where $\mathcal{L}_{\text{PLM}} = \sum_k \mathcal{L}_{\text{PLM}}^k$ represents the aggregated task losses for all source tasks, and $\lambda$ is a weight to balance the impact of distillation loss terms.

### 3.2 SOURCE TRAINING AND TARGET ADAPTATION

Training the single source prompt to be transferred to target tasks requires two steps. First, the teacher prompts for all source tasks are pretrained individually through vanilla prompt tuning. Then, we conduct multitask training on $\mathcal{S}$ to jointly learn the single shared prompt via the knowledge distillation loss function in Equation 5. We also adopt a simple stochastic task sampling strategy, which dynamically changes the number of tasks per batch. In particular, for each batch of multitask samples, we randomly select a number $k$ from $[2, \kappa]$ first, then randomly choose $k$ tasks from $\mathcal{S}$ and their corresponding samples to constitute mini-batches. Such dynamic task sampling strategies are common in the PLM multitask learning literature (Raffel et al., 2020).

For target adaptation, initialize the target prompt to be the Hadamard product of the shared prompt and the low-rank target prompt matrix and optimize with the regular task loss in Equation 1. We remark that our proposed method can also be used for multitask learning on target tasks to enable more parameter-efficient adaption of pretrained language models.

**Parameter-Efficiency.** Our method is parameter-efficient during both source training and target adaptation. Each task contains the shared prompt $l \times d$ that has the same dimensions as a vanilla soft prompt and a smaller number of task-specific vectors $(l + d)$. Thus, the total number of tunable parameters for a single target task is $(l \times d) + (l + d)$. For a *group* of target tasks, the total number of tunable parameters is $(l \times d) + (l + d)\tau$, where $\tau$ is the number of target tasks. We list and compare different methods in terms of the number of trainable parameters in Table 1.

## 4 EXPERIMENTS

We conduct extensive experiments on 21 diverse NLP datasets to show that MPT outperforms strong baselines in both full-dataset (Tables 1, 2) and few-shot adaptation settings (Table 3), while achieving more parameter-efficiency compared to existing methods (Figure 2). We also perform comprehensive ablation studies and analysis to better understand the effect of model sizes (Figure 4), prompt length (Figure 5) and different components in our approach (Table 4).

### 4.1 EXPERIMENTAL SETUP

**Datasets and Tasks.** As in Asai et al. (2022) we evaluate performance of MPT using 6 datasets with more than 100k annotations as *source* tasks (MNLI (Williams et al., 2017), QNLI (Demszky et al., 2018), QQP (Wang et al., 2018), SST-2 (Socher et al., 2013), SQuAD (Rajpurkar et al., 2016), and ReCoRD (Zhang et al., 2018)) and 21 datasets from four benchmarks as *target* tasks, namely MultiRC (Khashabi et al., 2018), BoolQ (Clark et al., 2019a), WiC (Pilehvar & Camacho-Collados, 2018), WSC (Levesque et al., 2012), and CB (De Marneffe et al., 2019) from SuperGLUE (Wang et al., 2019); RTE (Giampiccolo et al., 2007), CoLA (Warstadt et al., 2019), STS-B (Cer et al., 2017), MRPC (Dolan & Brockett, 2005), MNLI, QQP, QNLI and SST-2 from GLUE (Wang et al., 2018); Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), NewsQA (Trischler et al., 2017) and SearchQA (Dunn et al., 2017) from MRQA (Fisch et al., 2019); WinoGrande (Sakaguchi et al., 2021), Yelp-2 (Zhang et al., 2015), SciTail (Khot et al., 2018) and PAWS-Wiki (Zhang et al., 2019) from the Others benchmark in (Asai et al., 2022).

**Models.** Following the standard prompt tuning (Lester et al., 2021; Asai et al., 2022), we mainly experiment using the publicly available pretrained T5-Base model with 220M parameters (Raffel et al., 2020). In our ablation we also consider T5-Small, and T5-Large with 60M, and 770M parameters respectively, to empirically analyze the effect of model size on MPT performance in Figure4.

**Baselines.** We compare our approach with the following baselines. (1) Full finetuning (FT), where all the model parameters are tuned during adaptation on each downstream task, (2) vanilla prompt tuning (PT) (Lester et al., 2021), where target prompt vectors are initialized by randomly sampled top vocabularies, (3) existing prompt transfer methods, including SPoT (Vu et al., 2022) and ATTEMPT (Asai et al., 2022), that initialize target prompts by retrieving or aggregating source prompts, (4) popular parameter-efficient methods including Adapter (Houlsby et al., 2019) and BitFit (Zaken et al., 2022). On GLUE, we also compare with several state-of-the-art methods that adapt a pretrained model to all the target tasks using multitask learning, such as HyperFomer (Mahabadi et al., 2021), HyperDecoder (Ivison & Peters, 2022), multitask variants of FT, and Adapter. We directly quote numbers reported in published papers when possible or use publicly available source

**Table 1: Results on GLUE and SuperGLUE.** We adopt Pearson Correlation for STS-B, F1 for MultiRC (Multi), and accuracy for other tasks as evaluation metrics. "param/task" represents number of trainable parameters for each task in GLUE. The top part of the table denotes model adaptation to each target task (so param/task for MPT is just $(l \times d) + (l + d)$). The bottom part (marked by $^*$) denotes model adaptation to a *group* of tasks, where the param/task for MPT * is $(l \times d)/\tau + (l + d)$. See Section 3.2 for more details.

| Method | param/ task | GLUE | | | | | | | | | SuperGLUE | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MNLI | QQP | QNLI | SST-2 | STS-B | MRPC | RTE | CoLA | Avg. | Multi | BoolQ | WiC | WSC | CB | Avg. |
| Finetuning | 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 |
| Adapter | 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 |
| BitFit | 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 |
| PT | 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 |
| SPoT | 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 |
| ATTEMPT | 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 | 77.6K | $85.9_{0.07}$ | $90.3_{0.00}$ | $93.1_{0.07}$ | $93.8_{0.09}$ | $90.4_{0.05}$ | $89.1_{0.23}$ | $79.4_{1.22}$ | $62.4_{0.94}$ | $\mathbf{85.6}_{0.33}$ | $74.8_{0.07}$ | $79.6_{0.43}$ | $69.0_{0.25}$ | $67.3_{0.00}$ | $79.8_{2.91}$ | $\mathbf{74.1}_{0.73}$ |
| Finetuning* | 28M | 85.7 | 91.1 | 92.0 | 92.5 | 88.8 | 90.2 | 75.4 | 54.9 | 83.8 | - | - | - | - | - | - |
| Adapter* | 1.8M | 86.3 | 90.5 | 93.2 | 93.0 | 89.9 | 90.2 | 70.3 | 61.5 | 84.4 | - | - | - | - | - | - |
| HyperFomer* | 638K | 85.7 | 90.0 | 93.0 | 94.0 | 89.7 | 87.2 | 75.4 | 63.7 | 84.8 | - | - | - | - | - | - |
| HyperDecoder* | 1.8M | 86.0 | 90.5 | 93.4 | 94.0 | 90.5 | 87.7 | 71.7 | 55.9 | 83.7 | - | - | - | - | - | - |
| ATTEMPT* | 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* | 10.5K | $84.3_{0.57}$ | $90.0_{0.13}$ | $93.0_{0.24}$ | $93.3_{0.26}$ | $90.4_{0.07}$ | $89.2_{0.98}$ | $82.7_{0.41}$ | $63.5_{0.05}$ | $\mathbf{85.8}_{0.14}$ | $74.8_{0.07}$ | $79.2_{0.67}$ | $70.2_{0.82}$ | $67.3_{0.00}$ | $89.3_{0.00}$ | $\mathbf{76.1}_{0.31}$ |

codes (Karimi Mahabadi et al., 2021; Mahabadi et al., 2021; Asai et al., 2022) under the same backbone and experimental settings for a fair comparison.

**Implementation Details.** Following (Karimi Mahabadi et al., 2021), for all datasets, we use the development set as the testing set if the original testing set is not publicly available. We split the original development set into the development and testing set if the training set is small; otherwise, we separate a development set from the training set and use the original development set for testing. We limit the number of training data for Yelp to 100k. For source training, we train MPT on the mixture of source tasks for 5 epochs with the examples-proportional mixing strategy (Raffel et al., 2020) and stochastic task sampling described in Section 3.2. For prompt distillation, we calculate the hidden state loss for hidden states from both the encoder and decoder of T5.

For target adaptation, we reuse the shared prompt from MPT and take averaged source task-specific vectors to initialize the target task-specific vector. We train 20 epochs on small datasets, 10 epochs on large (more than 10k examples) datasets, and 5 epochs on the MRQA dataset. We run all the experiments three times with different random seeds and report the mean and variances. During source training, we set the default learning rate as $0.3$ for both task-shared and task-specific components. However, during target adaptation, we use a strategy of two-speed learning rates for those two components, as in Ponti et al. (2022). We set the learning rate to $0.3$ and $0.4$ for task-shared and task-specific components during adaptation for each target task. Following Lester et al. (2021), we set the default number of tunable tokens per each prompt as 100 and initialize the teacher and student prompts by randomly sampling tokens from T5's vocabulary (Raffel et al., 2020). We set the default batch size for T5-Base as 32 and for model scaling experiments, the batch sizes for T5-Small and T5-Large are 100, and 12 respectively. The default input length for most tasks are set to $256$, except MultiRC and MRQA benchmark have the input length as $348$ and $512$. We set the distillation loss coefficient $\lambda$ in Equation 5 to $0.9$ and keep it fixed for all our experiments. In few-shot experiments, for each number of shots $k$, we randomly sample $10$ times from the training set with different random seeds and report the mean performances. Following (Asai et al., 2022), we report our performance BoolQ, CB, and SciTail with same validation and testing sets in full-dataset setting. Note that for few-shot setting, the source prompt learning still uses the full set of the source tasks.

### 4.2 Results and Analysis

**Full-Dataset Adaptation.** Tables 1- 2 show the per-task performance of different methods on all four benchmarks. As seen from Table 1 (top part), MPT establishes new state-of-the-art results for parameter-efficient finetuning on both GLUE and SuperGLUE. When compared to vanilla PT (Lester et al., 2021), MPT obtains more than $+10\%$ points improvement on average performance ($+13\%$ on GLUE, $+16\%$ on SuperGLUE) with the same number of task-specific parameters, which demonstrates that multitask prompt tuning provides an effective means of improving the performance of PT, especially on small datasets. MPT consistently outperforms SPoT (Vu et al., 2022) to obtain an average of $85.6\%$ on GLUE and $74.1\%$ on SuperGLUE, which is $+3.3\%$ and $+11.2\%$ point accuracy improvements respectively. Furthermore, MPT achieves $2.1\%$ and $3.6\%$ average accuracy improvements over ATTEMPT (Asai et al., 2022), despite updating $3\times$ fewer parameters on GLUE and SuperGLUE respectively. Similarly, when comparing against BitFit (Zaken et al., 2022) which only tunes the bias vectors, MPT outperforms it by $+2.3\%$ on GLUE and $+3.6\%$ on SuperGLUE, while tuning $2\times$ fewer parameters for each target task. Among compared methods,

**Table 2: Results on MRQA and Others.** We use F1 for MRQA tasks and accuracy for others as the evaluation metrics. MPT outperforms ATTEMPT on both benchmarks, while tuning $67\%$ less parameters.

| Method | param/task | MRQA | | | | | Others | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NQ | HP | SQA | News | Avg. | WG | Yelp | SciTail | PAWS | Avg. |
| Finetuning | 220M | 75.1 | 77.5 | 81.1 | 65.2 | **74.7** | 61.9 | 96.7 | 95.8 | 94.1 | **87.1** |
| Adapter | 1.9M | 74.2 | 77.6 | 81.4 | 65.6 | 74.7 | 59.2 | 96.9 | 94.5 | 94.3 | 86.2 |
| BitFit | 280K | 70.7 | 75.5 | 77.7 | 64.1 | 72.0 | 57.2 | 94.7 | 94.7 | 92.0 | 84.7 |
| PT | 76.8K | 67.9 | 72.9 | 75.7 | 61.1 | 69.4 | 49.6 | 95.1 | 87.9 | 55.8 | 72.1 |
| SPoT | 76.8K | 68.2 | 74.8 | 75.3 | 58.2 | 69.1 | 50.4 | 95.4 | 91.2 | 91.1 | 82.0 |
| ATTEMPT | 232K | 70.4 | 75.2 | 77.3 | 62.8 | 71.4 | 57.6 | 96.7 | 93.1 | 92.1 | 84.9 |
| MPT | 77.6K | $72.0_{0.11}$ | $75.8_{0.14}$ | $77.2_{0.05}$ | $63.7_{0.06}$ | $72.2_{0.09}$ | $56.5_{0.87}$ | $96.4_{0.01}$ | $95.5_{0.26}$ | $93.5_{0.13}$ | $85.5_{0.32}$ |

**Table 3: Few-Shot Results with $k$ = {4, 16, 32}.** FT: Finetuning, AD: Adapter, PT: Prompt tuning, ST: SPoT, HF: HyperFormer, ATP: ATTEMPT. Numbers in bracket denote the number of parameters tuned for each task. Our proposed MPT consistently outperforms PT by a very large margin and competitive or even better than existing methods on majority of the cases, while tuning much fewer task-specific parameters.

| $k$-shot | | FT (220M) | AD (1.9M) | PT (76.8K) | ST (76.8K) | HF (638K) | ATP (232K) | MPT (77.6K) |
|---|---|---|---|---|---|---|---|---|
| BoolQ | 4 | 50.5 | 53.4 | 61.6 | 50.5 | 48.0 | 61.8 | **62.2** |
| | 16 | 56.5 | 51.4 | 61.9 | 50.6 | 50.2 | 60.0 | **63.3** |
| | 32 | 58.4 | 54.5 | 61.7 | 61.2 | 58.3 | 65.3 | **68.9** |
| CB | 4 | 57.7 | 51.1 | 53.5 | 71.4 | 60.7 | **82.1** | 73.6 |
| | 16 | 77.0 | 74.8 | 63.5 | 64.3 | 76.3 | 78.5 | **78.6** |
| | 32 | 80.0 | 74.8 | 67.8 | 64.3 | 81.4 | **85.7** | 82.1 |
| SciTail | 4 | 79.6 | 79.5 | 57.7 | 69.6 | 82.0 | 80.2 | **80.2** |
| | 16 | 80.0 | 83.2 | 60.8 | 71.9 | 86.5 | 79.5 | **87.3** |
| | 32 | 81.9 | 85.0 | 60.2 | 71.9 | 85.8 | 80.2 | **86.3** |

Adapters is the most competitive in terms of average accuracy on both benchmarks. MPT also outperforms Adapters (Houlsby et al., 2019) on GLUE with $4\times$ fewer task-specific parameters. More surprisingly, MPT outperforms the full model finetuning baseline on both benchmarks, despite tuning $0.035\%$ as many task-specific parameters (see Figures 2 for a comparison between different methods versus their number of updated parameters on GLUE and SuperGLUE benchmarks).

When compared with state-of-the-art multitask baselines which train a single model on different target tasks, including recent HyperFormer (Mahabadi et al., 2021) and HyperDecoder (Ivison & Peters, 2022), Table 1 (bottom part) shows that our MPT$^*$ (w/ prompt decomposition on target tasks) performs well and also further improves upon the single target task baseline. This reveals the potential of our method to further leverage multitask knowledge on the *target* side to enable even more parameter-efficient adaptation of pretrained language models.

Table 2 shows the performance of different methods on the MRQA and Others benchmark. Our approach significantly improves the average performance of PT by $+2.8\%$ on MRQA and $+13.5\%$ on the Others benchmark, while adding only $0.01\%$ more task-specific parameters. Similarly, MPT obtains $85.5\%$ average accuracy on WinoGrande, Yelp, SciTail, and PAWS, outperforming BitFit ($84.7\%$), which updates $10\times$ more task-specific parameters. We further investigate the performance of MPT by increasing prompt length from 100 to 300 and find an average improvement of $0.8\%$ on MRQA and $0.6\%$ on Others, closing the gap between MPT and Adapters (e.g., only $0.1\%$ difference in Others benchmark: see Appendix C for more details). While our improvements being highly parameter-efficient are encouraging, the accuracy gap between MPT and the full finetuning is still significant in MRQA, which indicate opportunities for future work in multitask prompt tuning.

**Few-Shot Adaptation.** Following prior works (Mahabadi et al., 2021; Asai et al., 2022), in addition to the full dataset adaptation on four benchmarks, we conduct few-shot experiments on BoolQ, CB, and SciTail tasks to measure how pretrained MPT prompts can be generalized to new tasks with only a few training examples available ($k = 4, 16, 32$). Table 3 shows the results of our approach and other baselines, including full-model finetuning, Adapter, and HyperFormer. As can be seen from Table 3, vanilla PT performs poorly in few-shot adaptation (esp., CB and SciTail), suggesting randomly initialized prompts are hard to generalize to new tasks with only a few shots. SPoT improves the performance of PT on CB and SciTail tasks, and MPT outperforms both PT and SPoT. We also observe that other methods in Table 3 (Finetuning, Adapter, HyperFormer, and ATTEMPT) have trouble in the few-shot setting. These results indicate that MPT can effectively use cross-task knowledge in source tasks to target tasks where there are only a few labeled examples. Appendix D shows full performance comparison between PT and MPT on all the GLUE and SuperGLUE tasks.

**Scaling.** We conduct scaling experiments to analyze how MPT performs with increasing pretrained model sizes on three SuperGLUE tasks as in (Asai et al., 2022). Figure 4 shows the performance of MPT as well as full model finetuning (FT), Adapter, prompt tuning (PT), and ATTEMPT with three different T5 models (T5-Small, T5-Base, T5-Large). Performance on WiC, MultiRC, and BoolQ shows that MPT can greatly benefit from scaling up the backbone model and outperforms PT and ATTEMPT consistently across all model sizes. These results show that our prompt decomposition strategy is not only able to achieve the best parameter efficiency but also effective across different model scales ranging from 60M to 770M parameters.

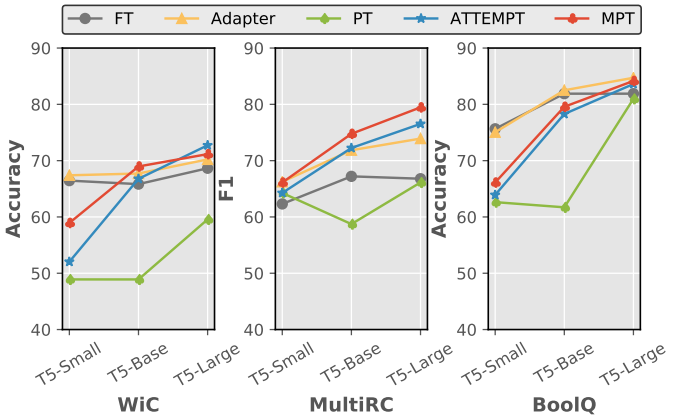


**Figure 4: Model Scaling.** With the increase of backbone PLM sizes (from T5-Small to T5-Large), the performance of our proposed MPT is improved consistently across tasks. Best viewed in color.

In addition to increasing model sizes, we also increase the length of the prompt $l$ to add more parameters and compare it with vanilla PT on SuperGLUE. Figure 5 compares PT and MPT over various prompt lengths $l = \{100, 200, 300\}$. From Figure 5, we observe that increasing prompt length for PT only produces marginal improvement, which is consistent with recent findings in (Lester et al., 2021; Li & Liang, 2021). On the other hand, MPT can get consistent improvements till 300 prompt length to reach the average performance of 76.8% on SuperGLUE. However, a further increase of prompt length from 300 to 400 lead to 1.8% drop in accuracy, indicating the optimal prompt length to be of 300 for our current experiments. We leave the in-depth exploration of the relationship between prompt capacity and knowledge transfer to future work.

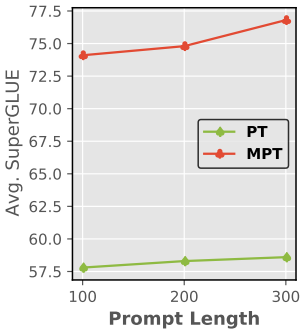


**Figure 5: Prompt Scaling.** Increasing prompt length can effectively boost MPT performance.

**Analyzing Learned Prompts.** We conduct qualitative analysis on prompts learned using MPT to investigate whether cross-task knowledge is indeed encoded in the task-shared prompt which makes it easier for target tasks to effectively adapt and encode their own knowledge effectively.

Following (Vu et al., 2022), we leverage task embedding to compute cosine similarities between all target task pairs after adaptation, where each task is represented by the composition of task-shared and task-specific prompts (averaged to obtain a single vector). Figure 6 shows the visualization of cosine similarity matrices for SPoT and MPT on SuperGLUE tasks. We find that task embeddings can effectively cluster similar tasks together, such as MultiRC is similar to BoolQ, and WiC is also close to both of them. Moreover, we also observe that MPT has clearer clusters than SPoT, which verifies our hypothesis that MPT helps target tasks to encode task-specific knowledge more effectively.

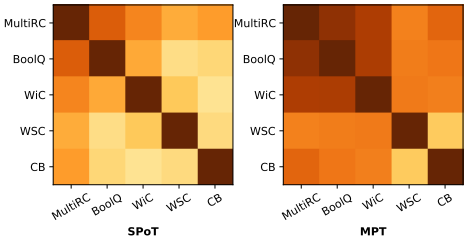


**Figure 6: Analyzing task correlation using prompt similarities on SuperGLUE.**

### 4.3 Ablation Studies

We conduct extensive ablation studies to measure the importance of various components of MPT and justify several important modeling strategies in our framework in the following subsections.

**Effectiveness of Prompt Decomposition.** Table 4 presents our ablation studeis on SuperGLUE for testing the effect of prompt decomposition and prompt distillation. We fix all the hyper-parameters

across all settings and rerun MPT source training to get various ablated versions of the transferred prompt. First, we ablate both prompt decomposition and distillation and initialize a vanilla prompt to be shared across all source tasks (no task-specific vectors). We train it with the simple mixing of all datasets, then transfer the resulting prompt to target tasks for adaptation. Table 4 shows that simply training a single soft prompt only produces an average accuracy of $69.5\%$ on SuperGLUE (top row), as it fails to leverage commonalities across source tasks while minimizing interference. To measure the effect of prompt decomposition, we replace the vanilla source prompt with our decomposable prompt with task-shared and task-specific components and train it without prompt distillation (third row in Table 4), which gives us $3.5\%$ average performance improvement on SuperGLUE. This ablation clearly indicates the importance of our proposed prompt decomposition strategy in MPT and demonstrates that the shared component can effectively capture the rich cross-task knowledge to be beneficial to target downstream tasks.

**Table 4: Ablation studies on SuperGLUE.**

| Decomposition | Distillation | Avg. Score |
|---|---|---|
| ✗ | ✗ | 69.5 |
| ✗ | ✓ | 70.6 |
| ✓ | ✗ | 73.0 |
| ✓ | ✓ | 74.1 |

**Effectiveness of Prompt Distillation.** To test the effect of prompt distillation, we ablate prompt decomposition and train a vanilla prompt shared by all the source tasks with the same training loss of MPT in Equation 5. The teacher models are kept the same for this ablation and MPT. Compared with the simple baseline (first row in Table 4), adding prompt distillation (second row) produces $1.1\%$ average performance improvement, which verifies the effectiveness of prompt distillation. With distillation, the vanilla shared prompt can be trained by more fine-grained learning signals from each task, but without prompt decomposition, knowledge of all the source tasks is entangled together, which may hurt transfer performance on target tasks. Finally, we observe that prompt distillation combined with prompt decomposition yields the best average performance of $74.1\%$ on SuperGLUE benchmark. This confirms that distilling knowledge from separately-trained source prompts is an effective strategy for learning good decomposable prompts.

We further investigate the individual components of prompt distillation to see their influences on final performance. We remove the loss of hidden states from Equation 5 and find that it produces an average performance of $73.68\%$ on SuperGLUE, verifying the effectiveness of regularizing hidden states in conjunction with logits to reach its full performance, which is consistent with findings in (Sanh et al., 2019). Lastly, we consider a variant of distillation loss to match the teacher and student prompts directly by adding an MSE loss to minimize the distance between those two prompts. Replacing our proposed distillation losses with this prompt distance loss and jointly training it with prompt decomposition yields an average SuperGLUE performance of $73.6\%$, which performs worse than the distillation losses based on logits and hidden states.

**Ablation on Target Adaptation Strategies.** When transferring the shared prompt from source training to target tasks, we ablate MPT with the choices of how to tune the task-shared and task-specific components for target tasks. We find that only updating the task-shared component (i.e., removing target task-specific vectors) or only updating task-specific vectors (i.e., freezing task-shared component) produce unsatisfied results ($62.5\%$ and $71.3\%$ SuperGLUE, respectively). This indicates the significance of keeping both components for prompt decomposition on target adaptation.

**Effectiveness of Stochastic Task Sampling.** Inspired by stochastic depth for training deep neural networks (Huang et al., 2016), we propose a multitask training strategy in Section 3.2, which is to stochastically sample a various number of tasks within each mini-batch to help the shared component of MPT be robust to task variances. Ablating this training strategy produces an average performance on SuperGLUE as $73.66\%$, which verifies the importance of this simple multitask training strategy.

## 5 Conclusion

We introduced and studied multitask prompt tuning (MPT), which learns a single transferable prompt by decomposing and distilling knowledge from multiple source tasks as well as their task-specific source prompts. MPT decomposes the task prompt by the Hadamard product of a shared prompt matrix and a rank-one task-specific matrix. The shared component is then transferred and adapted to target tasks to be further tuned. Empirically we found this approach enables parameter-efficient transfer learning to target downstream tasks across diverse NLP benchmarks, even outperforming full finetuning baseline in some cases despite tuning much fewer task-specific parameters.

## REFERENCES


Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5799–5811, 2021a.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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)*, pp. 7319–7328, 2021b.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attentional mixtures of soft prompt tuning for parameter-efficient multi-task knowledge sharing. *arXiv preprint arXiv:2205.11961*, 2022.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

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

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc Le. Bam! born-again multi-task networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5931–5937, 2019b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pp. 107–124, 2019.

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

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

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

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

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

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*, 2019.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pp. 1607–1616, 2018.

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

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, 2017.

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

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of ACL*, 2021.

Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, pp. 8678–8690, 2022.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

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

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. Attention-guided answer distillation for machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2077–2086, 2018.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

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

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

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

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

Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

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

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

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

Xiang Lisa Li and Percy Liang. 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)*, pp. 4582–4597, 2021.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*, 2022a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021a.

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

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

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 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)*, pp. 565–576, 2021.

Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. Coherence boosting: When your pretrained language model is not paying enough attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8214–8236, 2022.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6253–6264, 2022.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3994–4003, 2016.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4885–4901, 2020.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

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

Edoardo M Ponti, Alessandro Sordoni, and Siva Reddy. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914*, 2022.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4822–4829, 2019.

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

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, 2021a.

Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2339–2352, 2021b.

Siamak Shakeri, Abhinav Sethy, and Cheng Cheng. Knowledge distillation in document retrieval. *arXiv preprint arXiv:1911.11065*, 2019.

Sam Shleifer and Alexander M Rush. Pre-trained summarization distillation. *arXiv preprint arXiv:2010.13002*, 2020.

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

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, et al. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3949–3969, 2022.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33: 8728–8740, 2020.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24193–24205. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/cb2653f548f8709598e8b5156738cc51-Paper.pdf.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. 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)*, pp. 4149–4158, 2019.

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

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. Exploring and predicting transferability across nlp tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7882–7926, 2020.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 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)*, pp. 5039–5059, 2022.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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*, pp. 353–355, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Chengyu Wang, Jianing Wang, Minghui Qiu, Jun Huang, and Ming Gao. Transprompt: Towards an automatic transferable prompting framework for few-shot text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2792–2802, 2021.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2022.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *European Conference on Computer Vision*, pp. 588–604, 2020.

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

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *arXiv preprint arXiv:2104.08835*, 2021.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 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)*, pp. 1–9, 2022.

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

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

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Yuan Zhang, Jason Baldridge, and Luheng He. 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)*, pp. 1298–1308, 2019.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *arXiv preprint arXiv:2208.10160*, 2022.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2856–2878, 2021.

Chunting Zhou, Jiatao Gu, and Graham Neubig. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*, 2019.

## A MPT FOR NLG TASKS

**Table 5:** Applying MPT-T5-Large prompts to NLG tasks. MPT consistently outperforms PT on both tasks.

| | E2E | | | | | WebNLG | | |
|---|---|---|---|---|---|---|---|---|
| | BLEU | NIST | METEOR | Rouge-L | CIDEr | BLEU | METEOR | TER (↓) |
| PT | 29.11 | 5.00 | 0.343 | 51.50 | 1.72 | 46.02 | 0.37 | 46.89 |
| MPT | 32.14 | 5.35 | 0.363 | 52.88 | 1.86 | 52.27 | 0.40 | 41.36 |

We follow the standard evaluation protocol of prior works (Asai et al., 2022; Vu et al., 2022) to conduct our experiments on GLUE, SuperGLUE, MRQA and Other benchmarks. In this section, we conduct additional experiments on NLG tasks by applying T5-MPT source prompt on target NLG tasks. In particular, we transfer the T5-Large prompt trained using 6 diverse source tasks used in our current experiments for adaptation to two target tasks, namely E2E (Novikova et al., 2017) and WebNLG (Gardent et al., 2017). Table 5 shows that our proposed MPT significantly outperforms standard PT (Lester et al., 2021) on both NLG tasks across all the metrics. Our BLEU improvements over PT are $3.03\%$ and $6.25\%$ on E2E and WebNLG tasks respectively, showing the effectiveness of our approach on both NLU (e.g., classification, NLI, QA tasks) and NLG tasks. This is particularly an impressive result since the source tasks were all NLU tasks, i.e., MPT can transfer knowledge from NLU tasks to NLG tasks.

## B MPT WITH MORE NUMBER OF SOURCE TASKS

**Table 6:** MPT performance on MRQA and Others with more number of source tasks.

| | MRQA | | | | | Others | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | NQ | HP | SQA | News | Avg. | WG | Yelp | SciTail | PAWS | Avg. |
| MPT (w/ 6 Source Tasks) | 72.0 | 75.8 | 77.2 | 63.7 | 72.2 | 56.5 | 96.4 | 95.5 | 93.5 | 85.5 |
| MPT (w/ 12 Source Tasks) | 72.1 | 76.4 | 77.9 | 64.0 | 72.6 | 56.6 | 96.8 | 95.9 | 92.9 | 85.6 |

Following (Asai et al., 2022), we select 6 representative NLP tasks as source datsks, including 2 NLI, 1 paraphrase, 1 sentiment analysis and 2 large-scale QA tasks, which are general/diverse enough and can enable knowledge transfer to other tasks. In this section, we investigate the effect of adding more remotely relevant source tasks to our proposed multitask prompt tuning. Specifically, we add 6 additional diverse source tasks, including one topic classification (AGNews (Zhang et al., 2015)), three multi-choice QA (CommmonsenseQA (Talmor et al., 2019), OpenBookQA (Mihaylov et al., 2018), ARC (Clark et al., 2018)), one adversarial NLI (ANLI (Nie et al., 2020)) and one commonsense (winogrande (Sakaguchi et al., 2021)) datasets. Table 6 shows the results on MRQA and Others benchmarks. As can be seen, MPT with 12 tasks is still very effective for target adaptation on both benchmarks, slightly outperforming MPT trained using 6 tasks.

Furthermore, it would be compelling to use benchmarks like CrossFit (Ye et al., 2021) consisting of 160 NLP tasks as source tasks for analyzing the performance of MPT on parameter-efficient transfer learning. While we currently do not possess the compute resources for this extreme large-scale study, we hope to cover this an interesting future work. Last but not least, we will release pretrained source task prompts and easily extendable code so that it can motivate further studies on task scaling and understanding transferabilty across a more diverse set of source and target tasks.

## C PROMPT SCALING FOR MRQA AND OTHERS

Our proposed MPT outperforms both Finetuning and Adapter baslines on GLUE and SuperGLUE benchmarks (Table 1). However, they're still better than MPT on MRQA and Others benchmarks (Table 2) with 2832 and 24 times more parameters than MPT, respectively. While adding the same number of prompt parameters as the Adapter to close the performance gap on MRQA and Others benchmarks is an interesting suggestion, we note that it requires a prompt length of 2400 tokens on T5-Base, which can be computationally inefficient due to transformer's quadratic complexity with the input length. Table 7 shows that increasing prompt length from 100 to 300 yields an average improvement of 0.8% on MRQA and 0.6% on Others, further closing the gap between MPT and

**Table 7:** Performance on MRQA and Others benchmark by scaling prompt length. All results are based on T5-Base. MPT-300 is very competitive to Adapter on Others benchmaek while being highly parameter-efficient.

| | | **MRQA** | | | | | **Others** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | param/task | NQ | HP | SQA | News | Avg. | WG | Yelp | SciTail | PAWS | Avg. |
| Fine-tuning | 220M | 75.1 | 77.5 | 81.1 | 65.2 | 74.7 | 61.9 | 96.7 | 95.8 | 94.1 | 87.1 |
| Adapter | 1.9M | 74.2 | 77.6 | 81.4 | 65.6 | 74.7 | 59.2 | 96.9 | 94.5 | 94.3 | 86.2 |
| MPT-100 | 77.6K | 72.0 | 75.8 | 77.2 | 63.7 | 72.2 | 56.5 | 96.4 | 95.5 | 93.5 | 85.5 |
| MPT-300 | 231.5K | 72.6 | 76.4 | 78.4 | 64.3 | 73.0 | 57.0 | 97.0 | 96.8 | 93.8 | 86.1 |

Adapters (e.g., only 0.1% difference in Others benchmark). We also test with a prompt length of 400 tokens but did not notice any significant improvements. We believe this is because the optimal prompt length in our current experiments is around 300 tokens as discussed in our prompt scaling analysis. Applying MPT for every layer of the pretrained model, instead of only input layer (like P-Tuning v2 (Liu et al., 2021b)) could be a promising direction to further improve the performance: we leave this as an interesting future work.

## D Few-Shot Results on GLUE and SuperGLUE

**Table 8:** Few-Shot results on GLUE and SuperGLUE with $k = \{4, 16, 32\}$. MPT consistently outperforms PT, demonstrating generalizability of MPT prompts to new tasks with only a few training examples.

| $k$-shot | | **MNLI** | **QQP** | **QNLI** | **SST-2** | **STS-B** | **MRPC** | **RTE** | **CoLA** | **Avg.** | **Multi** | **BoolQ** | **WiC** | **WSC** | **CB** | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | PT | 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 |
| 16 | PT | 41.5 | 62.3 | 59.9 | 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 |
| 32 | PT | 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 |

Following (Asai et al., 2022), we conduct few-shot experiments on BoolQ, CB, and SciTail tasks for a fair and direct comparison with other parameter-efficient methods, namely SpoT (Vu et al., 2022), ATTEMPT (Asai et al., 2022) and Hyperformer (Mahabadi et al., 2021). In this section, we conduct more comprehensive few-shot experiments on all the GLUE and SuperGLUE tasks by comparing vanilla PT (Lester et al., 2021) and MPT. As shown in Table 8, we can observe that not only MPT outperforms the vanilla PT by a large margin in most of the datasets, but also MPT can perform very well on many datasets to reach their full-dataset performance with 16 or 32 shots, such as QQP, QNLI, STS-B, and WSC. These results clearly indicate that MPT can effectively use cross-task knowledge in source tasks to target tasks where there are only a few labeled examples.

## E Generalization of MPT to GPT Models

Our current MPT is built on top of prompt tuning (Lester et al., 2021), which is mostly applied to T5 (Raffel et al., 2020). So, we follow prior works, such as SPoT (Vu et al., 2022) and ATTEMPT (Asai et al., 2022), to conduct experiments on T5-variants. However, our proposed approach is quite generic and can be applied to both T5 and GPT models. This is primarily because MPT only prepends a prompt matrix (i.e., virtual tokens) to the input embedding layer and hence can be adopted to any transformer models (encoder-only, encoder-decoder, or decoder-only), not limited to T5. Specifically, MPT focuses on decomposing the prompt matrix into task-specific and task-shared components, which introduces minimal intrusion to the backbone model. Similarly, the distillation part of MPT is also model-agnostic and can be generalized to GPT models. We leave the extension of MPT to GPT models as an interesting topic for future work.