ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts

Akari Asai[♡] **Mohammadreza Salehi**[♡] **Matthew E. Peters**[◇] **Hannaneh Hajishirzi**[♡◇]
[♡] University of Washington [◇] Allen Institute for AI
{akari, mrsalehi, hannaneh}@cs.washington.edu
matthewp@allenai.org

Abstract

This work introduces a new multi-task, parameter-efficient language model (LM) tuning method that learns to transfer knowledge across different tasks via a mixture of soft *prompts*—small prefix embedding vectors pre-trained for different tasks. Our method, called ATTEMPT (ATTEntional Mixtures of Prompt Tuning), obtains *source prompts* as encodings of large-scale source tasks into a small number of parameters and trains an attention module to interpolate the source prompts and a newly initialized target prompt for every instance in the target task. During training, only the target task prompt and the attention weights, which are shared between tasks in multi-task training, are updated, while the original LM and source prompts are intact. ATTEMPT is highly parameter-efficient (e.g., updates 2,300 times fewer parameters than full fine-tuning), while achieving high task performance using knowledge from high-resource tasks. Moreover, it is modular using pre-trained soft prompts and can flexibly add or remove source prompts for effective knowledge transfer. Our experimental results across 21 diverse NLP datasets show that ATTEMPT significantly outperforms prompt tuning and outperforms or matches fully fine-tuned or other parameter-efficient tuning approaches that use over ten times more parameters. Finally, ATTEMPT outperforms previous work in few-shot learning settings.[1]

1 Introduction

Fine-tuning all the parameters of large language models (LMs) given target task training data is the most common practice for optimizing task performance (Devlin et al., 2019; Raffel et al., 2020). A recent line of research introduces parameter-efficient tuning methods (Houlsby et al., 2019; Li and Liang, 2021; Ben Zaken et al., 2022) that only update a small number of LM parameters; however, increasing efficiency often decreases the task performance (He et al., 2022). Moreover, these models are trained only using the task training data and do not benefit from large collection of other NLP tasks (Liu et al., 2019a). We posit that parameter-efficient tuning methods can leverage rich knowledge of high-resource tasks to improve both training efficiency and task performance.

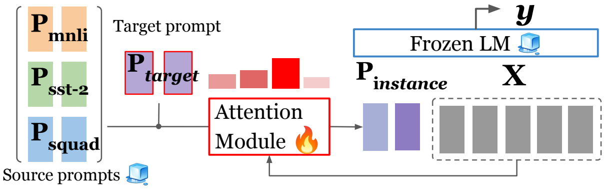


Figure 1: ATTEMPT combines multiple soft prompts trained on large-scale datasets (*source prompts*) to generate instance-wise prompts for a target task. At target task training, the LM and the source prompts are intact.

This work introduces a new parameter-efficient, modular multi-task tuning method called ATTEMPT (ATTEntional Mixtures of Prompt Tuning, previewed in Figure 1). ATTEMPT efficiently integrates knowledge from multiple tasks via a mixture of trainable soft prompts preprended to the input, keeping the original LM completely frozen. It first pre-trains transferable soft embeddings (Lester et al., 2021), called *source prompts*, on large-scale source tasks, which are likely to contain knowledge beneficial to other tasks. Then, ATTEMPT initializes a new *target prompt* for a given target task and learns an attention-weighted combination of source prompts and the target prompt. The attention module is a light-weight network that can be shared and trained simultaneously across tasks.

ATTEMPT offers three key advantages over previous multi-task fine-tuning or parameter-efficient tuning methods: first, it is **highly parameter-efficient** and achieves competitive performance despite updating only 0.4% of the parameters

[1] Our code is available at https://github.com/AkariAsai/ATTEMPT.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6655–6672
December 7-11, 2022 ©2022 Association for Computational Linguistics

in full fine-tuning. Second, it enables **modular multi-task learning** using pre-trained soft prompts, where knowledge from different tasks can be flexibly combined, reused, or removed, and new tasks can be added to the lists of source or target tasks. Unlike prior work that relies on precomputed priors on which tasks are related, ATTEMPT learns to focus on useful tasks from many source tasks. Moreover, at inference, a single LM with multiple pre-loaded soft prompts can perform multiple tasks without parameter reloading. Lastly, it **improves interpretability** on underlying task similarities in multi-task learning by generating attention distributions.

We conduct experiments on 21 datasets across diverse tasks, domains and output formats. ATTEMPT significantly outperforms previous prompt tuning-based approaches and matches state-of-the-art parameter-efficient transfer approaches or fully fine-tuned models that train orders of magnitude more parameters, especially on smaller datasets. ATTEMPT is also effective on few-shot domain adaptations (i.e., 4-32 shots).

Our analysis further shows that ATTEMPT is particularly parameter-efficient and competitive with larger backbone LMs, where other parameter-efficient transfer approaches shows rapid increases of the trainable parameters. Our ablation studies suggest that learned attentions, multi-task learning and modular transfer from multiple tasks largely contribute to the performance improvements. The attention distributions show the underlying similarities among seemingly different tasks (e.g., entailment and paraphrase detection), indicating signal for effective knowledge transfer across tasks.

2 Background and Problem Setup

We first enlist common paradigms in NLP for learning a target task, which differ in terms of available data and resources. We then describe our problem setup with respect to these paradigms.

Fine-tuning. The most common practice in learning a new target task T_{target} is to fine-tune all parameters of a pre-trained LM on the target task training data $\{(\boldsymbol{x}, \boldsymbol{y})\}$ (e.g., Devlin et al. 2019). Formally, given pre-trained LM parameters θ, fine-tuning results in a specialized model θ_{target} by optimizing: $\max_{\theta_{target}} p_{\theta_{target}}(\boldsymbol{y} \mid \boldsymbol{x})$.

Parameter-efficient tuning. To decrease training costs, parameter-efficient tuning updates a small number of parameters for the target task ϕ_{target}: $\max_{\phi_{target}} p_{\theta,\phi_{target}}(\boldsymbol{y} \mid \boldsymbol{x})$, where the number of ϕ_{target} is much smaller than θ_{target}. Adapter (Houlsby et al., 2019) and its variants (Mahabadi et al., 2021a; Rücklé et al., 2021) insert trainable layers in the LMs for each task, and BitFit (Ben Zaken et al., 2022) directly updates LM biases only. Highly efficient prefix-tuning (Li and Liang, 2021) and prompt tuning (Lester et al., 2021) keep the original LM frozen and only update *soft prompts* prepended to the input. In-context learning (Brown et al., 2020) uses massive-scale LMs to learn new tasks from demonstrations (*hard prompts*) without any parameter update of θ, but often perform worse than the aforementioned methods with parameter updates (Liu et al., 2022). Given the rapidly increasing size of pre-trained LMs (Chowdhery et al., 2022; Brown et al., 2020), efficiently tuning to a new target task is desirable, but it often incurs a performance cost compared to the fine-tuning methods or shows sensitivity toward initialization (Li and Liang, 2021; Lester et al., 2021). SPoT (Vu et al., 2022) demonstrates that transferring prompts to another task enhances the performance at the cost of massive search.

Multi-task transfer learning. Transfer learning methods attempt to learn a new target task given a collection of source tasks by updating the parameters of an LM, which has been proven effective in NLP (Khashabi et al., 2020; Raffel et al., 2020). Common approaches train on many different tasks (Liu et al., 2019a; Aribandi et al., 2022) or transfer a model fine-tuned on source tasks to another target task (Vu et al., 2020; Talmor and Berant, 2019). Several recent work introduce zero-shot or few-shot transfer of massive multi-task pretrained models (Sanh et al., 2022; Min et al., 2021; Wang et al., 2022a,b) via in-context learning, which does not require any parameter updates. However, those massive multi-task training approaches lack the flexibility of adding or removing source tasks even when some of the tasks cause negative interference between competing tasks (Zhang et al., 2020; Aghajanyan et al., 2021).

Our problem setup. We combine parameter-efficient tuning and multi-task learning. Given a collection of source tasks $T_1, \ldots T_t$, our goal is to learn a new task T_{target} by efficiently updating parameters ϕ_{target} given the target task data $\{(\boldsymbol{x}, \boldsymbol{y})\}$, transferring knowledge from the source

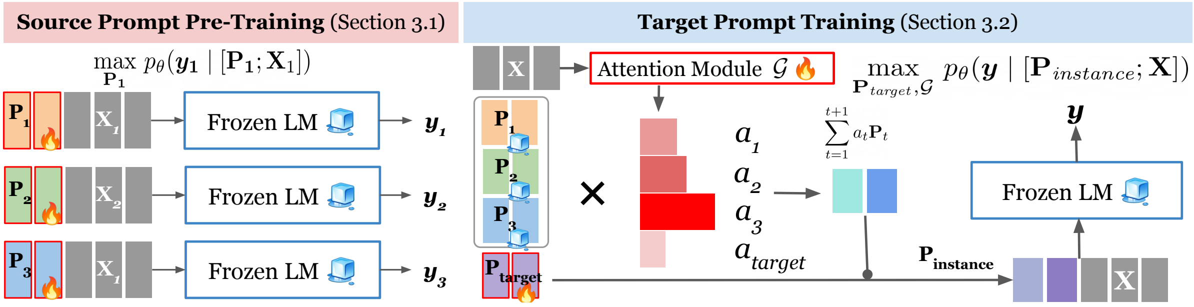

Source Prompt Pre-Training (Section 3.1)
Target Prompt Training (Section 3.2)
$\max_{\mathbf{P}_1} p_\theta(\boldsymbol{y}_1 \mid [\mathbf{P}_1; \mathbf{X}_1])$
$\mathbf{P}_1$
$\mathbf{X}_1$
Frozen LM
$\boldsymbol{y}_1$
$\mathbf{P}_2$
$\mathbf{X}_2$
Frozen LM
$\boldsymbol{y}_2$
$\mathbf{P}_3$
$\mathbf{X}_1$
Frozen LM
$\boldsymbol{y}_3$
X
Attention Module $\mathcal{G}$
$\max_{\mathbf{P}_{target}, \mathcal{G}} p_\theta(\boldsymbol{y} \mid [\mathbf{P}_{instance}; \mathbf{X}])$
$\mathbf{P}_1$
$\mathbf{P}_2$
$\mathbf{P}_3$
$\mathbf{P}_{target}$
×
a_1
a_2
a_3
a_{target}
$\sum_{t=1}^{t+1} a_t \mathbf{P}_t$
$\mathbf{P}_{instance}$
$\boldsymbol{y}$
Frozen LM
X

Figure 2: Overview of ATTEMPT. The parts framed in red are updated during training while other parts are intact.

tasks. Importantly, we do not know a priori which tasks provide useful inductive bias in the new target task (Ponti et al., 2022): seemingly different tasks can benefit from each other.

3 Method

ATTEMPT (depicted in Figure 2) leverages highly parameter-efficient prompt tuning (Lester et al., 2021) to obtain *source prompts* that encode knowledge from source tasks into a small number of parameters. It tunes instance-level prompts by integrating the *source prompts* and a *target prompt* newly initialized for a target task through an attention mechanism for every target task instance.

ATTEMPT pre-trains a set of source prompts $\mathbf{P}_1, \ldots, \mathbf{P}_t$ for source tasks (Section 3.1; left side of Figure 2) and initializes a target prompt $\mathbf{P}_{target}$ for the target task. It then computes attentions between embedded input $\mathbf{X}$ and the soft prompts for each instance $(\boldsymbol{x}, \boldsymbol{y})$ using an attention module $\mathcal{G}$ (Section 3.2.1). Subsequently, ATTEMPT produces instance-wise prompt $\mathbf{P}_{instance}$ by interpolating the source prompts and the target-task prompt given the computed attentions (Section 3.2.2). $\mathbf{P}_{instance}$ is then prepended to the input to form the final input to a frozen LM θ.

During training, ATTEMPT only updates the weights of $\mathbf{P}_{target}$ and $\mathcal{G}$ by maximizing the probability of generating $\boldsymbol{y}$ given $\mathbf{P}_{instance}$ and $\boldsymbol{x}$. Importantly, it uses the unique characteristic of prompt or prefix tuning, where task-specific parameters ϕ_{target} for different tasks can be trained in the same minibatch (Lester et al., 2021; Li and Liang, 2021). Hence, it can train a shared attention $\mathcal{G}$ and multiple target task prompts simultaneously for further parameter and inference efficiency (Section 3.3). Finally, we discuss parameter efficiency of ATTEMPT in Section 3.4.

3.1 Source Prompt Pre-training

We first obtain source prompts $[\mathbf{P}_1, \ldots, \mathbf{P}_t]$ for t high-resource datasets, such as MultiNLI (Williams et al., 2018), SQuAD (Rajpurkar et al., 2016) through prompt tuning (Lester et al., 2021). Each source prompt is only trained once for a source task and can be transferred to different target tasks. Formally, for an input sequence $\mathbf{X}$, a soft prompt is represented as $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_m] \in \mathbb{R}^{m \times d}$, where m is the prompt length, and d is the LM dimension. Input embeddings prepended by the prompt $[\mathbf{P}; \mathbf{X}]$ are fed into the frozen LM θ. During training, only prompt embeddings are updated by maximizing the likelihood of generating the target sequence $\boldsymbol{y}$, as follows:

$$\max_{\mathbf{P}} p_\theta(\boldsymbol{y} \mid [\mathbf{P}; \mathbf{X}]). \qquad (1)$$

3.2 Target Prompt Training

After initializing a soft prompt for a new target task $\mathbf{P}_{target}(= \mathbf{P}_{t+1})$, we learn instance-wise soft prompts $\mathbf{P}_{instance}$ for each instance in the target task by interpolating the source prompts and the target task prompt given attention scores generated by $\mathcal{G}$. Similar to Eq. 1, we concatenate the produced instance-wise prompt to the input and train ATTEMPT by maximizing the likelihood:

$$\max_{\mathbf{P}_{target}, \mathcal{G}} p_\theta(\boldsymbol{y} \mid [\mathbf{P}_{instance}; \mathbf{X}]). \qquad (2)$$

During training, the new task prompt $\mathbf{P}_{target}$ and $\mathcal{G}$ are updated via $\mathbf{P}_{instance}$, while source prompts and the original LM θ are untouched to preserve the knowledge learned from prior tasks or pretraining.

3.2.1 Input-prompt Attentions

ATTEMPT controls the influence of the set of source prompts on the instance-wise prompt by calculating input-prompt attentions. Specifically, an attention module $\mathcal{G}$ generates the attention weights

$a_1, \dots, a_{t+1}$ from input $\mathbf{X}$ to the prompts including both source prompts and the new target prompt.

Since the input $\mathbf{X} \in \mathbb{R}^{l\times d}$ and a soft prompt $\mathbf{P}_j \in \mathbb{R}^{m\times d}$ have different sequence lengths, we first perform the max-pool operation for each dimension on $\mathbf{X}$ and each source prompt embedding and obtain $\hat{\mathbf{X}} \in \mathbb{R}^d$ and $\hat{\mathbf{P}}_j \in \mathbb{R}^d$.[2] We then feed $\hat{\mathbf{X}}$ to a sub-network $\mathcal{G}$ to project it into the prompt spaces. For efficiency, $\mathcal{G}$ consists of down and up projection layers, as follows:

$$\begin{aligned}
\mathbf{H}_{down} &= \mathbf{W}_{down}^\top(\hat{\mathbf{X}})\\
\mathbf{H}_{up} &= \mathbf{W}_{up}^\top(\text{NonLinear}(\mathbf{H}_{down}))\\
\mathbf{H}_{out} &= \text{LayerNorm}(\mathbf{H}_{up}),
\end{aligned}$$

where $\mathbf{W}_{down} \in \mathbb{R}^{d\times r} (r < d)$ and $\mathbf{W}_{up} \in \mathbb{R}^{r\times d}$ are projection parameters to be updated during training. We use SiLU (Elfwing et al., 2017) for the non-linear layer and apply Layer Norm (Ba et al., 2016) on $\mathbf{H}_{up}$, observing that without layer norm, $\mathbf{H}_{up}$ often grows quickly and gradients explode.

Finally, we compute the attentions by calculating the product between $\hat{\mathbf{P}}_j$ and $\mathbf{H}_{out}$, and apply softmax over the prompts, as follows:

$$a_j = \frac{e^{\hat{\mathbf{P}}_j \mathbf{H}_{out}}/T}{\sum_{k=1}^{t+1} e^{\hat{\mathbf{P}}_k \mathbf{H}_{out}}/T}, \tag{3}$$

where T is a softmax temperature (Radford et al., 2021) and scale the logits in Eq. 3 to avoid making the attention module over-confident.

3.2.2 Prompt Interpolation

The final soft prompt for the instance $\mathbf{X}$ is calculated as the weighted sum of the prompts given the attention generated by Eq. 3:

$$\mathbf{P}_{instance}(\mathbf{X}) = \mathbf{P}_{target} + \sum_{j=1}^{t+1} a_j \mathbf{P}_j. \tag{4}$$

The second term on the right differs for different instances of the same task, while the $\mathbf{P}_{target}$ term is task-specific. The attentions act as a gate to control the influences from different prompts and enable a flexible composition of knowledge from multiple tasks. As shown in Eq. 4, the selection of $1 + a_{t+1}$ weights for the target-task-specific prompt $\mathbf{P}_{target}(= \mathbf{P}_{t+1})$ enables ATTEMPT to *down-play* the role of source prompts if the knowledge from none of the sources tasks is useful for the instance $\mathbf{X}$, while always keeping the influence of $\mathbf{P}_{target}$ so that it will be properly updated during training.

3.3 Multi-task Training and Inference

Training. ATTEMPT can jointly train the attention module $\mathcal{G}$ and multiple target task prompts. Here, we explain our approach on multi-task learning over a group of target tasks by sharing $\mathcal{G}$.

It first concatenates the training datasets, while keeping each task ID information. During training, we retrieve the target-task prompt corresponding to the instance given the task ID, calculate attentions over the set of the prompts and produce instance-wise prompt as described in Section 3.2. The loss for each target task prompt only backpropagates when the prompt is used, while the weights of the attention module is updated at each iteration.

This way, target tasks are loosely connected and together contribute to an improved and task-agnostic attention module, which is particularly effective when the target task training data is small. Moreover, this reduces the number of parameters to be updated per task and improves the efficiency of inference time.

Inference. At inference time, we load source prompts, all of the target task prompts and the shared $\mathcal{G}$ just once. For each instance, ATTEMPT retrieves the target task prompt and produces $\mathbf{P}_{instance}$ as in Eq. 4, and then concatenates $\mathbf{P}_{instance}$ to the input embedding. The inference process after producing instance prompt is exactly the same as in prompt tuning.

ATTEMPT enables loading multiple target task prompts and performing multiple target tasks simultaneously, significantly reducing the inference time model loading overhead. Existing approaches such as full fine-tuning or Adapter requires model loading for different target tasks, making its multi-task inference pipeline complicated.

3.4 Parameter Efficiency of ATTEMPT

For each task, we will introduce a new trainable soft prompt $m \times d$, where m is the length of the prompts and d is the LM's dimension. An attention module consists of two projection matrices and a layer norm, resulting in $d \times r + r \times d + 2d = 2rd + 2d$ parameters, where r is the projection dimension. As this can be shared across N target tasks, the additional parameters per task will be: $d \times m + \frac{2rd+2d}{N} = d(m + 2(r+1)/N)$. A unique characteristic of ATTEMPT or prompt tuning is their independence from the number of the LM layers; With Adapter or fine-tuning, the number of the parameters quickly increases as the backbone LMs

[2] This does not add any new parameters, and empirically performs slightly better than other pooling approaches.

get larger. ATTEMPT, in contrast, updates only the soft prompts and do not modify the LM higher layers, resulting in moderate parameter increases compared to other approaches. When we use T5-XL as a backbone LM, Adapter and BitFit updates about 6 million and 2 million parameters respectively, while ATTEMPT only updates and stores 172k parameters per task (Figure 7 in Appedix).

4 Experiments

4.1 Source and Target Tasks

We use 6 large-scale datasets as *source tasks*, and evaluate on 21 diverse *target tasks* including entailment, paraphrase detection, sentiment analysis, question answering (QA), commonsense reasoning. Datasets details are in Appendix Section B.

Source tasks. We use the following datasets with more than 100k annotations in total from GLUE, SuperGLUE and MRQA for source prompts: MNLI (Williams et al., 2018), QNLI (Demszky et al., 2018), QQP (Wang et al., 2019b), SST-2 (Socher et al., 2013), SQuAD (Rajpurkar et al., 2016), and ReCoRD (Zhang et al., 2018).

GLUE and SuperGLUE. We use 8 GLUE tasks (Wang et al., 2019b) and 5 SuperGLUE (Wang et al., 2019a) tasks as target datasets to test the model's natural language understanding abilities: BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), MultiRC (Khashabi et al., 2018), WiC (Pilehvar and Camacho-Collados, 2019), WSC (Levesque et al., 2012), RTE (Giampiccolo et al., 2007), CoLA (Warstadt et al., 2019), STS-B (Cer et al., 2017), MRPC (Dolan and Brockett, 2005), MNLI, QQP, QNLI and SST-2. Four of the GLUE datasets used as source tasks (MNLI, QQP, SST-2 and QNLI) are also included as target tasks to provide comprehensive comparisons with prior parameter-efficient tuning methods, whose evaluations often focus on GLUE (Lester et al., 2021; Ben Zaken et al., 2022).

Question answering. We use the MRQA 2019 shared task (Fisch et al., 2019) data to test on four large-scale QA datasets: 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).

Others. We experiments on four different datasets, whose tasks are related to the source tasks but domains differ. SciTail (Khot et al., 2018) is a scientific entailment dataset. Yelp-2 (Zhang et al., 2015) is a sentiment analysis dataset on Yelp reviews. WinoGrande (Sakaguchi et al., 2020) is commonsense reasoning task in multiple choice format. PAWS-Wiki (Zhang et al., 2019) is a Wikipedia-based paraphrase detection dataset.

4.2 Baselines and Implementation Details

Baselines. We compare ATTEMPT with: **fine-tuning (FT)**; **prompt tuning** (**PT**; Lester et al. 2021), where target prompt embeddings are initialized by randomly sampled top vocabularies; **SPoT** (Vu et al., 2022), where target prompts are initialized by source prompt embeddings trained on other tasks (details are in Appendix); **Adapter** (Houlsby et al., 2019), **AdapterDrop** (Rücklé et al., 2021) and **BitFit** (Ben Zaken et al., 2022). On GLUE, we also compare ATTEMPT with several state-of-the-art multi-task methods, which train a single model on different tasks: **FT-multi-task (FT-m)**, **Adapter-m**, **HyperFormer** (Mahabadi et al., 2021b), **HyperDecoder** (Ivison and Peters, 2022), and **AdapterFusion** (Pfeiffer et al., 2021).

Implementation details. Although our methods, ATTEMPT and ATTEMPT-m use the same six source task prompts, ATTEMPT-m trains a shared attention layer across multiple target tasks by conducting multi-task training, while ATTEMPT trains a task-specific attention layer separately. Unless specified, we use T5-base as our base LMs for ATTEMPT and all of the baselines.[3] If a dataset does not have public test split with annotations, we use a development set as our test set or split the development set into our development and test sets, following Mahabadi et al. (2021a). We train for 20 epochs on small datasets with less than 10k examples, 10 epochs on medium-size data with more than 10k examples, and 5 epochs on MRQA datasets and limit the maximum training data number of Yelp-2 to be 100k samples. To make $\mathcal{G}$ learn a good prompt composition for efficient knowledge transfer, we introduce different learning rates for $\mathcal{G}$ (Ponti et al., 2022) and also pre-train and trans-

[3] Although the original prompt tuning paper uses T5 v1.1 LM-adapt as the backbone LMs, despite our extensive hyperparameter searches across five different learning rates and five different batch sizes, we could not reproduce the original results. We found that T5-LM adapt v1.1 was especially sensitive and hard to tune when we use it as a backbone LM for parameter-efficient approaches. Therefore, in this work we used T5 as backbone models. Prior work in this line also uses T5 as backbone models (Mahabadi et al., 2021a).

		GLUE									Super GLUE					
data (# of train)	param / task	MNLI (393k)	QQP (364k)	QNLI (105k)	SST-2 (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.
Fine-tuning	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**
AdapterDrop	1.1M	86.3	90.2	93.2	93.6	**91.4**	86.3	71.2	62.7	84.4	72.9	82.3	68.3	67.3	85.7	75.3
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	77k	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	77k	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
Fine-tuning-m†	28M	85.7	91.1	92.0	92.5	88.8	**90.2**	75.4	54.9	83.8	–	–	–	–	–	–
Adapter-m†	1.8M	86.3	90.5	93.2	93.0	89.9	**90.2**	70.3	61.5	84.4	–	–	–	–	–	–
HyperFormer†	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	–	–	–	–	–	–
AdapterFusion*	–	84.2	90.7	–	92.2	–	90.3	76.8	–	–	–	76.3	–	–	**92.1**	–
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
ATTEMPT -m	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

Table 1: Results on GLUE. All of the results are based on T5-base models. For GLUE experiments, we exclude SQuAD and ReCoRD from source prompts inventories for comparison with prior work. We use Pearson Correlation for STS-B, F1 for MultiRC (Multi), and accuracy for other tasks as metrics. "param/task" denotes the number of the parameters trained for each task in GLUE. † from Mahabadi et al. (2021b); ‡ from Ivison and Peters (2022); * from Pfeiffer et al. (2021) and their base LM is RoBERTa-base (Liu et al., 2019b).

data (# of train)	params / task	NQ (100k)	HP (72k)	SQA (117k)	News (74k)	Avg.	WG (40k)	Yelp (100k)	SciTail (27k)	PAWS (49k)	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
BitFit	280k	70.7	75.5	77.7	64.1	72.0	57.2	94.7	94.7	92.0	84.7
Prompt tuning	77k	67.9	72.9	75.7	61.1	69.4	49.6	95.1	87.9	55.8	72.1
SPoT-t	77k	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
ATTEMPT-m	134k	72.5	76.7	78.0	63.9	72.8	58.6	96.2	94.6	92.8	85.6

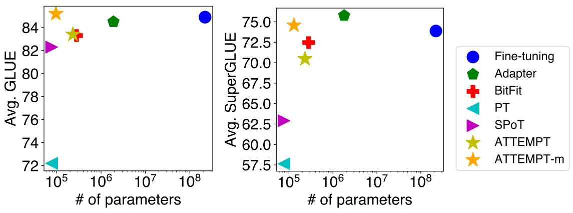
Table 2: Results on MRQA 2019 QA datasets, WinoGrande (WG), Yelp, Scitail and PAWS. We use F1 for MRQA and accuracy for others. "param/task" denotes parameter trained per task in MRQA and others.

fer the weights of $\mathcal{G}$ from the source tasks. More experimental details are in Appendix.

Prompt initialization. Each source prompt is initialized by randomly sampling tokens from the top vocabularies as in Lester et al. (2021). For target task prompt initialization, we use the MNLI source prompt for non-QA tasks and the SQuAD source prompt for QA, instead of initializing it with randomly sampled vocabularies for training stability.

5 Results

We present main results in Section 5.1 and few-shot domain transfer experiments on sampled tasks in Section 5.2, demonstrating the effectiveness of ATTEMPT especially when the data is scarce. Section 5.3 further provides set of analyses.



(a) GLUE (b) SuperGLUE

Figure 3: Parameter-efficiency and average scores. We use T5-base for all of the models.

5.1 Main Results

Tables 1 and 2 present the per-task performance of the GLUE and SuperGLUE datasets, and the other datasets, respectively.

Performance vs. efficiency. Figures 3a and 3b compare the performance of different models versus their number of updated parameters on GLUE and SuperGLUE. ATTEMPT-m significantly outperforms PT, SPoT and BitFit by a large margin, and matches Adapter or Fine-tuning despite updating much fewer parameter per each task and keeping the LM completely frozen. Table 1 shows ATTEMPT outperforms all of the multi-task baselines including recent HyperFormer or HyperDecoder. In addition to competitive performance on GLUE/SuperGLUE, Table 2 shows that ATTEMPT-m achieves 72.8 MRQA average F1, outperforming BitFit using twice as many parameters. Moreover, ATTEMPT-m yields 85.6% average accuracy on WinoGrande, Yelp, SciTail and PAWS, outperforming BitFiT (84.7%) and matching Adapter (86.2%) that updates ten times more parameters.

ATTEMPT largely improves prompt tuning. As pointed out by prior work (Mahabadi et al., 2021a; Lester et al., 2021; Sung et al., 2022), prompt tuning is sensitive to hyperparameters or initialization, and it has significantly lower performance on several datasets such as CoLA (10.2%), BoolQ (61.7%) or WiC (48.9%). SPoT (Vu et al., 2022) improves the target task prompt initialization with a prompt trained on other related tasks, but it still under-performs other approaches, and requires searching the source tasks beforehand. ATTEMPT largely outperforms those approaches on smaller datasets (e.g., CB, RTE), as well as large-scale MRQA datasets as shown in Table 2.

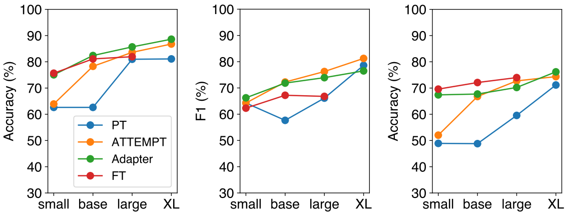
5.2 Few-shot Domain Adaptations

As shown in Table 2 ATTEMPT is particularly competitive on smaller dataset (e.g., RTE, WSC). Following Mahabadi et al. (2021b), we conduct few-shot experiments on BoolQ, CB and SciTail, to further verify the effectiveness of ATTEMPT under resource-constrained setup. Here, all of the models (Fine-tuning, Adapter, HyperFormer, SPoT and ATTEMPT) are first trained on the GLUE tasks and then transferred to new tasks using only k ($k = 4, 16, 32$) randomly sampled training data. More details of few-shot domain adaptation experiments are available at Appendix.

Table 3 shows that ATTEMPT significantly outperforms other methods in most of the setting. This indicate the effectiveness of transferring knowledge from multiple source tasks in a non-destructive manner in few-shot domain adaptation.

k-shot		FT	AD	SPoT	HF	ATP
BoolQ	4	50.5	53.5	50.5	48.0	**61.8**
	16	56.5	51.4	50.6	50.2	**60.0**
	32	58.4	54.5	61.2	58.3	**65.3**
CB	4	57.8	51.1	71.4	51.1	**82.1**
	16	77.0	74.8	64.3	74.8	**78.5**
	32	81.8	85.1	64.3	81.5	**85.7**
SciTail	4	79.6	79.5	69.6	**82.0**	80.2
	16	80.0	83.3	71.9	**86.6**	79.5
	32	82.0	85.1	71.9	**85.9**	80.2

Table 3: Few-shot results ($k = \{4, 16, 32\}$). FT, AD, HF and ATP denote Fine-tuning, Adapter, HyperFormer (Mahabadi et al., 2021b) and ATTEMPT.



(a) BoolQ (b) MultiRC (c) WiC

Figure 4: Performance with different backbone LMs.

5.3 Analyses

Power of scale. We empirically analyze how increasing the backbone LM size affects ATTEMPT performance. Figure 4 summarizes the performance of Adapter, ATTEMPT, prompt tuning (PT), and fully fine-tuned (FT) models vs. LM sizes on three SuperGLUE datasets.[4] ATTEMPT largely benefits from backbone LM size increase. This is aligned with the finding of Lester et al. (2021) that show prompt tuning is particularly effective when the backbone LM is larger. Moreover, ATTEMPT matches fully fine-tuned models even with T5-base or T5-large. This is in contrast to prompt tuning methods that suffers when the backbone LM is smaller. Furthermore, ATTEMPT performs on par with or outperforming Adapter with T5-3B, while updating 37 times less parameters.

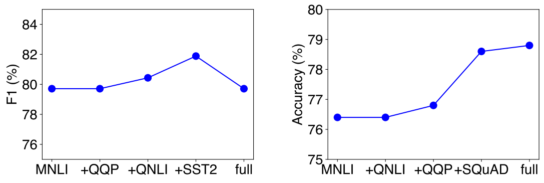
Ablation studies. We compare different variants of ATTEMPT to see the effect of each of the design choices.[5] We ablate ATTEMPT with (a) *no target*, which neither initializes nor adds target task prompts in Eq. 4, to assess the feasibility of adapting to a new task by only interpolating pre-

[4] We could not finetune T5-XL under the same computational constraints as the other baselines, so we do not report the fine-tuning with T5-XL performance.

[5] The ablations are mainly conducted over BoolQ, NewsQA and WinoGrande.

	BoolQ	NewsQA	WG
ATTEMPT-m	78.29	61.58	58.57
ATTEMPT	77.06	61.84	57.61
no target	50.89	55.26	47.89
no attention	73.57	52.55	56.03
single prompt	76.25	60.92	55.56

Table 4: Results of ablation studies. “WG” denotes WinoGrande. For NewsQA ablation, we used randomly sampled 10k data for training for quick ablation.



(a) RTE

(b) BoolQ

Figure 5: Performance on RTE and BoolQ dev sets when source prompts are added one by one, starting from the MNLI source prompt only.

trained source prompts; (b) *no attention*, which gives constant score $a_j = 1/t$ to all source prompts in Eq. 3, discarding attentions; (c) *single prompt*, which uses only a single source prompt to assess the effect of transferring knowledge from multiple tasks. Single prompt ablation is similar to SPoT except that instead of using source prompts for initialization and updating its during training, we keep the source prompt frozen while updating the target task prompt and the attention layers.

Table 4 indicates that all components contribute to performance improvements. Adding a trainable target-task-specific prompt (no target) is crucial to achieve good performance on all of the datasets, especially on BoolQ and WinoGrande. Constant attention causes large performance drop, especially on BoolQ and NewsQA, indicating that it is important to have learned attentions rather than simply averaging the multiple source prompts. Although the single prompt ablation baseline outperforms SPoT, possibly due to the non-destructive soft prompt transfer of ATTEMPT, there is notable performance decline relative to ATTEMPT. This demonstrates the effectiveness of leveraging multiple soft prompts to transfer knowledge from multiple diverse tasks.

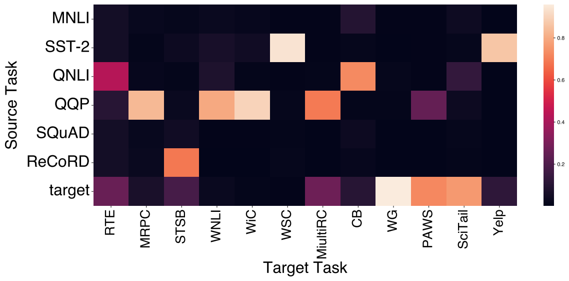


Figure 6: Attention visualizations of ATTEMPT.

Modularity: effects of variable source prompts. We study the modular nature of ATTEMPT that enables flexibly adding or removing source tasks. Figure 5 shows how including source tasks affects the final performance of ATTEMPT on two benchmarks, BoolQ and RTE. On both of the datasets, adding more source task prompts gives performance improvements, with an exception of adding SQuAD and ReCoRD on RTE (“full” in Figure 5a). This potentially happens because of the negative transfer due to the different natures of QA and RTE, while adding the two QA source prompts helps in BoolQ.

Interpretability: analysis on attentions. Figure 6 shows the attention weight matrix between source and target tasks by ATTEMPT. Note that for the target task prompt, we present the a_{t+1} weight before adding 1. Attention patterns differ for different tasks. Generally, $\mathcal{G}$ gives higher attentions to related source tasks: Yelp → SST-2, or PAWS-Wiki → QQP, which are the same tasks but are different in domains. QQP is often highly attended by some tasks that are seemingly different from paraphrasing (e.g., MultiRC, WNLI), which may indicate underlying task similarities between those tasks. Unlike the underlying task similarities, MNLI is not highly attended by some highly-related target tasks such as RTE. We hypoothesize that this is because the target task prompts for those tasks are initialized with the MNLI source prompt, and thus ATTEMPT may try to attend to other tasks. On WinoGrande or SciTail, $\mathcal{G}$ gives large attentions to the target task embeddings (“target”); this maybe because those two tasks have significantly different task format or input domain, and $\mathcal{G}$ ignores source prompts more.

6 Related Work

Parameter-efficient tuning. Here, we enlist additional parameter-efficient tuning methods that are close to our work. AdapterFusion (Pfeiffer et al., 2021) compose multiple different adapters by learning task-specific compositions on each task, and Friedman et al. (2021) take an average of multiple adapter layers after training adapters individually on different QA datasets. HyperFormer (Mahabadi et al., 2021b) and HyperDecoder (Ivison and Peters, 2022) train a shared hyper network to generate parameters of adapter layers. Qin and Eisner (2021) introduce mixture of soft prompts, where predictions given different prompts are ensembled for the same knowledge base relationship types. IDPG (Wu et al., 2022) and Instance-Dependent Prompt Tuning (Levine et al., 2022) learn to generate instance-wise prompts given the input encoded by LMs. Compared to the previous work, our main focus is transferring knowledge from multiple tasks to produce soft prompts rather than learning to generate them from scratch, and is much more efficient in terms of parameters and inference time.

Concurrent to our work, Liu et al. (2022) introduce $(\mathrm{IA})^3$ that multiplies intermediate activation by learned vectors for few-shot learning. Wang et al. (2022b) shows that combining a set of prompts retrieved from the prompt pool by a key-value mechanism yields competitive performance in computer vision continual learning. For generation tasks, Li et al. (2022) transfer multiple source prompts using multi-key memory network for prompt clustering and multi-head attention taking another LM output. In contrast, we present an efficient multi-task tuning that is effective in diverse NLP tasks. More importantly, prior work often relies on priors such as pre-computed clusters or another LM's predictions of which tasks should be used as source tasks. ATTEMPT removes the necessity of such priors by training an attention layer that learn to focus on relevant source tasks.

Several recent lines of research attempt to adapt a massive multi-task LM trained with instructions or demonstrations to a new task without any parameter updates (Sanh et al., 2022; Min et al., 2021; Wang et al., 2022a; Wei et al., 2022). The main focus of this paper is how to efficiently transfer rich multi-task knowledge from source tasks to target tasks with training data during target task training, while those work often emphasize on zero or few-shot transfer without any parameter updates.

Modular multi-task training. There is a large literature on composing multiple separate networks to handle different sub-tasks (Jacobs et al., 1991b,a; Andreas et al., 2016; McCann et al., 2018). As the LM size expands, several recent work tries to sparsely activate or employ light-weight modules for efficient multi-task learning (Gupta et al., 2022; Ponti et al., 2022; Fedus et al., 2022). In particular, we share the same intuition as the concurrent work (Ponti et al., 2022), which combines several skills encapsulated in parameter-efficient modules; however, our main focus is on how to transfer and share knowledge from resource-rich tasks in a super parameter-efficient way, while they focus on improving few-shot generalization ability. Moreover, ATTEMPT keeps LMs intact and updates fewer parameters.

7 Conclusion

We present a new parameter-effluent tuning method ATTEMPT, which learns to produce instance-wise prompts by interpolating multiple reusable soft prompts trained on source tasks and a new task-specific prompt, while keeping the original LM frozen. Our large-scale experiments demonstrate that ATTEMPT achieves a great trade-off between task performance and efficiency, introducing an interpretable and modular task transfer.

Limitations

Despite its parameter-efficiency and strong empirical results, ATTEMPT has several limitations: First, as prompt tuning increases the input token length by m prompt tokens, it increases the memory footprint and computational costs (Mahabadi et al., 2021a), although Lester et al. (2021) found that prompt length can be shortened when larger LMs are used as backbone models. We investigate this issue in Appendix Section C.10. Secondly, as the first step toward multi-task knowledge transfer via soft prompts, our evaluation focuses on classification and QA tasks, and our target tasks do not include the tasks that require long sequence generations (e.g., summarization). Future work can explore applications of ATTEMPT to more diverse sets of tasks. In addition, we use representative six NLP tasks as source tasks, but do not explore a large-scale experiments on many source task combinations. We will release pretrained source prompts and easily extendable code to facilitate future work on multi-task transfer via soft prompt transfer. Lastly, we do not test ATTEMPT on non-English tasks, and we will investigate the effectiveness of ATTEMPT in non-English languages or apply ATTEMPT for cross-lingual transfer.

Ethics Statement

ATTEMPT is trying to improve parameter-efficiency and transferrability of models so that groups with limited computational resources can still get benefit from state-of-the-art large-scale models. All of the experiments are based on widely-used general purpose datasets, which are unlikely to include harmful content. However, several datasets such as Yelp Review are created from existing review sites, and may have more risks of privacy issues or harmful content than some other datasets based on news or encyclopedic websites.

Acknowledgement

This research was supported by NNSF IIS-2044660, ONR N00014-18-1-2826, a Sloan fellowship and gifts from AI2, and the Nakajima Foundation Fellowship. We thank UW NLP and Allen NLP group members for their insightful discussion and Sandy Kaplan, Sewon Min, Ofir Press, and Yizhong Wang for their helpful feedback on this paper.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. In *EMNLP*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *CVPR*.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. 2022. Ext5: Towards extreme multi-task scaling for transfer learning. In *ICLR*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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

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

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)*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

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

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

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

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP)*.

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*.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2017. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*.

Dan Friedman, Ben Dodge, and Danqi Chen. 2021. Single-dataset experts for multi-dataset question answering. In *EMNLP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the*

ACL-PASCAL Workshop on Textual Entailment and Paraphrasing.

Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed H Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *ICLR*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

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

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

Robert A Jacobs, Michael I Jordan, and Andrew G Barto. 1991a. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive science*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991b. Adaptive mixtures of local experts. *Neural computation*.

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 *NAACL*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *EMNLP*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *TACL*.

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

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*.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. 2022. Standing on the shoulders of giant frozen language models. *arXiv preprint arXiv:2204.10019*.

Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Learning to transfer prompts for text generation. In *NAACL*.

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

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

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *ACL*.

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

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *ACL*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. MetaICL: Learning to learn in context. In *NAACL*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. In *ICLR*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch:

An imperative style, high-performance deep learning library. In *NeurIPS*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *EACL*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL*.

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

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *NAACL*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

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. *JMLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *EMNLP*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *AAAI*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

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

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *arXiv preprint arXiv:2206.06522*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *ACL*.

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*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *ACL*.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *EMNLP*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

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

Yu-An Wang and Yun-Nung Chen. 2020. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *EMNLP*.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *CVPR*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *TACL*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vydiswaran, and Hao Ma. 2022. IDPG: An instance-dependent prompt generation method. *arXiv preprint arXiv:2204.04497*.

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 *EMNLP*.

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*.

Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. 2020. A survey on negative transfer. *arXiv preprint arXiv:2009.00909*.

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

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *NAACL*.

Appendix

A More Method Details

A.1 Improving Multi-task Training

Learning effective interpolations of prompts is challenging, as input embeddings themselves do not necessarily correspond to meaningful prompt tokens, and we do not have any supervisions for the ground truth task mapping. We explore several approaches to improve the training with good inductive bias so that $\mathcal{G}$ learns a good prompt composition for efficient knowledge transfer.

Learning attention prior. We pre-train the attention module on source tasks and then use the learned projection layers and the layer norm to initialize the attention module on the target task(s). This learned prior can be also directly used for tasks that lack training data.

Two-speed learning rate. Ponti et al. (2022) shows that setting different learning rates for the composition module and the task-specific model parameters helps to provide useful inductive bias to encourage the model to learn the best skill composition. We also introduce this two-speed learning rate approach for ATTEMPT.

A.2 Pre-training $\mathcal{G}$ on Source Tasks

To learn the attention prior for $\mathcal{G}$, we run the same training process as in the target task training on the source tasks. In particular, we initialize another task-specific prompt for each source task, and trains both those task-specific prompts as well as the shared attention weights of $\mathcal{G}$ on the combinations of the source tasks as in Section 3.2.

A.3 Overview of Training

Algorithm Table 5 presents the overview of the training algorithm.

A.4 Parameter Efficiency of ATTEMPT

Figure 7 shows the number of the parameters to be updated for Prompt tuning, ATTEMPT, Adapter, and BitFit when we increase the size of the backbone LMs. As we can see, other parameter-efficient transfer approaches observe quick increases of the trainable parameters, while ATTEMPT shows small increases. In addition, ATTEMPT keeps the original LM frozen and does not modify the LM structures unlike those approaches.

A.5 Alternative Attention Design

ATTEMPT computes the same attention scores over m prompt tokens. Alternatively, we compute attention scores for *each* prompt token further flexibility and expressiveness. Here, instead of computing similarities between the summary representation of $\mathbf{H}_{out}$ and prompt $\hat{\mathbf{P}}_j$, we compute similarities between $\mathbf{H}_{out}$ and each lth prompt token as follows:

$$a_{lj} = \frac{e^{\mathbf{p}_{lj}\mathbf{H}_{out}}}{\sum_{k=1}^{t+1} e^{\mathbf{p}_{lk}\mathbf{H}_{out}}}. \quad (5)$$

For prompt token-level attention, in the second term on the right, each lth prompt token in the summary representation is calculated as the weighted summary of the lth prompt tokens.

Empirically the token-level attentions gives similar performance to the original attention in Eq. 3, while in some tasks it gives notable performance improvements. Due to the additional computational overhead, we use the max-pooling based unified attention (Eq. 3) as our default attention mechanism. Interestingly, we find that the attention distributions are significantly different among prompt tokens in different locations (e.g., giving significantly higher attentions to the target task prompt in the later tokens), potentially because of the position biases of pretrained models (Wang and Chen, 2020). This is beyond the scope of this work, but is certainly of interest for future work.

B Task and Dataset Details

We show the list of the datasets, tasks and domains for source tasks in Table 6 and for target tasks in Table 7, respectively. In summary, both source and target datasets cover diverse tasks, domains and output formats (i.e., span extraction, multiple-choice, classification).

C Experimental Details

C.1 Implementation Details

We use PyTorch[6] (Paszke et al., 2019) and huggingface `transformers`[7] (Wolf et al., 2020) to implement our models. For Adapter, BitFit, prompt tuning and BitFit baselines, we use the implementations by Mahabadi et al. (2021a).[8] We use huggingface `datasets`[9] library to use the data for the

[6] https://pytorch.org/

[7] https://github.com/huggingface/transformers

[8] https://github.com/rabeehk/compacter

[9] https://github.com/huggingface/datasets

Source Prompt Training
For jth source tasks in t source tasks, train a source prompt $\mathbf{P}_j$ by maximizing $p(\boldsymbol{y} \mid [\mathbf{P}_j, \mathbf{X}]$ individually **(Section 3.1) [Eq. 2]**

Target Prompt Training
Initialization: initialize a new prompt $\mathbf{P}_{target}$ and attention module $\mathcal{G}$
For each instance $(\boldsymbol{x}, \boldsymbol{y})$, after passing $\boldsymbol{x}$ to the embedding layer to get input embeddings $\mathbf{X}$,
Step 1: Compute instance-wise prompt $\mathbf{P}_{instance}$ for $\mathbf{X}$ **(Section 3.2)**

1. calculate attentions between $\mathbf{X}$ and a set of prompts $[\mathbf{P}_1, \ldots, \mathbf{P}_t, \mathbf{P}_{target}]$ using $\mathcal{G}$ **[Eq. 3]**
2. interpolate $\mathbf{P}_1, \ldots \mathbf{P}_t$ and $\mathbf{P}_{target}$ using attention scores **[Eq. 4]**

Step 2: Prepend $\mathbf{P}_{instance}$ to $\mathbf{X}$ and feed the final input to frozen LM θ
Step 3: Maximize $p(\boldsymbol{y} \mid [\mathbf{P}_{instance}, \mathbf{X}])$ and backpropagate to $\mathbf{P}_{target}$ and $\mathcal{G}$ via $\mathbf{P}_{instance}$ **[Eq. 2]**

Table 5: Training process of ATTEMPT.

experiments except for MRQA 2019 shared task. For MRQA 2019 shared task, we download the original training and development data from the official repository.[10]

C.2 Source Prompt Training Details

We fine-tune the source prompts on six large-scale datasets for 5 epochs. We use the checkpoints with the best development score as our source prompts. Each source prompt is initialized by randomly sampled tokens as in Lester et al. (2021). We found that although this random vocabulary based initialization is often unstable even in large-scale datasets, on the six source tasks, this approach gives reasonable performance, even with T5-small.

C.3 Attention Module Pretraining Details

As the six source tasks have significantly different length of input context (e.g., the input context of MNLI, SST-2, QQP or QNLI is on average less than 200 tokens while SQuAD or ReCoRD have the context longer than 512 tokens), we split the source tasks into the two groups: (1) MNLI, SST-2, QQP and QNLI; (2) SQuAD and ReCoRD. We use the resulting pretrained weights from group (2) for MRQA 2019, while for other experiments, we use the weights from (1).

C.4 General hyperparameters

We set the maximum token length to be 512 for MRQA datasets, 348 for MultiRC and 256 for all of other datasets. All of the experiments are conducted with a single GPU with 24 GB memory. On all of the datasets, training were completed within 24 hours. Per GPU batch size is 32, and for MRQA, we set the per GPU batchsize to be 16 and set the gradient accumulation step to 2 due to the out of memory error.

C.5 Hyperparameters for ATTEMPT

We use $T = d \times \exp(1)$, where d is the LM dimension size, to control the soft max temperature in Section 3.2. The prompt length m is 100 and the prompt tuning learning rate is 0.3 and optimize the objective function using Adam (Kingma and Ba, 2015). We set weight decay to be 1×10^{-5}. For the projection layers, we use $r = 100$. For the attention module $\mathcal{G}$, we found that the best learning rate varies across datasets and tune it on the development sets. In particular, we use the learning rate of 0.1 for SuperGLUE, and Yelp, WinoGrande, SciTail and PAWS multi-task experiments, and 0.3 for the other experiments.

C.6 Hyperparameters for Baselines

For all of the baselines, we set the warmup steps to be 500, use Adam for optimization with a linear learning rate scheduler.

Prompt Tuning. As in ATTEMPT, we use the prompt length of $m = 100$ and use the learning rate of 0.3 for prompt tuning and set weight decay to be 1×10^{-5}.

[10] https://github.com/mrqa/MRQA-Shared-Task-2019

Dataset Name	Category	Task	Domain	Metric
1. MNLI	GLUE	natural language inference (NLI)	various	accuracy
2. SST-2	GLUE	sentiment analysis	Movie Reviews	accuracy
3. QQP	GLUE	paraphrase detection	social QA questions (Quora)	accuracy & F1
4. QNLI	GLUE QA	NLI	Wikipedia	accuracy
5. SQuAD	MRQA 2019	extractive QA	Wikipedia	F1 & EM
6. ReCoRD	SuperGLUE	cloze-style QA	news (CNN, Daily Mail)	F1 & EM

Table 6: The details of the 6 source tasks. MNLI, SST-2, QQP and QNLI are also used as target tasks in GLUE experiments.

Dataset Name	Category	Task	Domain	Metric
1. CoLA	GLUE	acceptability	various	Matthews corr.
2. STS-B	GLUE	sentence similarity	various	Pearson&Spearman corr.
3. MRPC	GLUE	paraphrase detection	news	accuracy & F1
4. RTE	GLUE	NLI	News, Wikipedia	accuracy
5. MultiRC	SuperGLUE	QA	various	F1 & EM
6. BoolQ	SuperGLUE	boolean QA	Wikipedia	accuracy
7. WiC	SuperGLUE	word sense disambiguation	lexical databases	accuracy
8. WSC	SuperGLUE	coreference / commonsense	fiction books	accuracy
9. CB	SuperGLUE	NLI	various	accuracy
10. NQ	MRQA 2019	extractive QA	Wikipedia	F1 & EM
11. HotpotQA	MRQA 2019	extractive QA	Wikipedia	F1 & EM
12. SearchQA	MRQA 2019	extractive QA	Search snippets	F1 & EM
13. NewsQA	MRQA 2019	extractive QA	News article	F1 & EM
14. WinoGrande	Others	coreference / commonsense	WikiHow	accuracy
15. Yelp	Others	sentiment analysis	Yelp reviews	accuracy
16. SciTail	Others	NLI	science exams	accuracy
17. PAWS-Wiki	Others	paraphrase detection	Wikipedia	accuracy

Table 7: The details of the 17 target tasks except for 4 GLUE datasets, which are also used for evaluation. "NQ" denotes Natural Questions and lexical databases for WiC include WordNet, VerbNet, Wiktionary. For the datasets where two metrics are originally used, we use the underlined metric as our primary metric.

SPoT. We explore two approaches to initalize the target task prompt as in Vu et al. (2022): **SPoT-*generic* (SPoT-g)** and **SPoT-*targeted* (SPoT-t)**. SPoT-g first pre-trains source prompts on eight GLUE tasks and then uses the source prompts to initialize target task prompts, while SPoT-t uses prompt similarities to find top-k similar tasks and then initializes target task prompts using the top k prompts. As we only use 6 source tasks in this work, we use top 1 similar prompt as the transfer source of SPoT-t. We use the same hyperparameters as in prompt tuning. To select the source task for SPoT-t, we run prompt tuning on all of the source and target tasks for 5 epochs for medium and large-scale datasets and 20 epochs for smaller scale datasets and then compute the cosine similarity between a target prompt and the set of the source prompts. Regarding the SPoT-g training, we train a single source prompt on the combination of the GLUE source tasks following Vu et al. (2022). We found that SPoT-g baseline is not strong on MRQA or Others (i.e., Yelp, Scitail, WinoGrande and PAWS-Wiki), while it gives small performance improvements on GLUE from SPoT-t in some tasks. Therefore, we use SPoT-t in our main experiments.

Adapter. We use the default hyperparameters by Mahabadi et al. (2021a) for the Adapter baseline. We use GELU (Hendrycks and Gimpel, 2016) for non-linear layers, set the reduction factor to be 32 and the learning rate to be 3×10^{-4}.

BitFit. We use the learning rate of 3×10^{-4}.

Fine-tuning. We use the learning rate of 3×10^{-4}. Other hyperparameters are the same as the huggingface transformers T5 models.

C.7 Multi-task Training Details

The 17 datasets have significantly different length of input context, and training on the combinations of all of the datasets can make training inefficient. We conduct multi-tasking of 4 datasets (SuperGLUE, MRQA 2019, and others), while on GLUE, we train ATTEMPT-m on 8 GLUE tasks. We keep MultiRC training separated from other SuperGLUE tasks, as MultiRC has significantly longer context

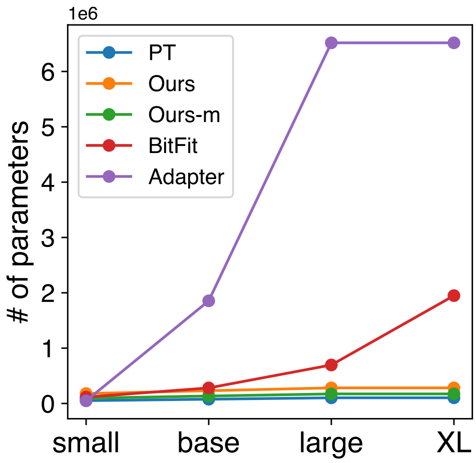


Figure 7: The number of the parameters to be updated with Adapter, BitFit, Fine-tuning, Prompt Tuning and ours using different backbone LMs. Ours and Ours-m denote ATTEMPT and ATTEMPT-m, respectively.

	Adapter			Fine-tuning		
datasets	Bool	MRC	WiC	Bool	MRC	WiC
T5-small	100	100	100	100	100	100
T5-base	64	64	100	32	32	100
T5-large	32	20	32	32	32	32
T5-3B	4	4	8	–	–	–

Table 8: The number of the batch sizes for fine-tuned models and adapter for the scalability experiments.

than other SuperGLUE datasets. We set the maximum length of the input to be 256, 256, 512, 256 for GLUE, SuperGLUE, MRQA 2019, and others task set, respectively. We set the maximum length of input to be 348 for MultiRC.

C.8 Few-shot Adaptation Experiments Details

Following (Mahabadi et al., 2021b), we run few-shot adaptation experiments for three times and takes the mean of the performance. We cite the performance of the fine-tuning, Adapter and HyperFormer from Mahabadi et al. (2021b), and train a single prompt tuning model on 8 GLUE tasks and then transfer it to few-shot tasks. For ATTEMPT, we load the attention weights trained on 8 GLUE tasks.

C.9 Scaling Experiments Details

During this experiment, we use only a single GPU with 24 GB GPU memory, as in our main experiments, to simulate a common resource environment. We found that under this computational constraint, we could not fine-tune the T5-3B model due to the out of memory error, even with a batch size of 1. Adapter, prompt tuning and ATTEMPT can be trained on a single GPU even with the T5-3B model. We provide the experimental details for the LM scaling experiments in Section 5.3. For ATTEMPT and prompt tuning, we use the same single GPU with 24 GB GPU memory as the main experiments. For Adapter and fine-tuning, we use a single GPU with 48 GB GPU memory but restrict GPU memory usage at 24 GB for a fair comparison. For the scalability experiments, we set the maximum token length to 216 across all datasets.

Per-device batch size for ATTEMPT and prompt tuning. For T5 small and base, we set per-GPU batch size to be 100 and 32, while for T5-large and T5-XL (3B), we use the batch size of 16 and 2, respectively.

Per-device batch size for Adapter. For Adapter experiments, we flexibly adjust the per-device batch size for each dataset to avoid out of the memory issues. The number of the per-device batch size is shown in Table 8.

Per-device batch size for fine-tuning. Similarly in Adapter, we adjust the per-device batch size for the fine-tuned models. The number of the per-device batch size is shown in Table 8. For fine-tuned models, we found that we cannot avoid the out of memory issue even with the batch size of 1, so we report the results with T5 small, base and large.

Performance Instability of fine-tuning with T5-large. We found that fine-tuning with T5-large is occasionally unstable and fails to learn a target task, and is sensitive to the batch size or learning rate. For instance, using different batch size results in 65% BoolQ accuracy. For those cases, we explored several learning rates and batch sizes and report the best performance. Several prior work report the instability of fine-tuning large-scale LMs (Mosbach et al., 2021; Dodge et al., 2020).

C.10 Memory Footprints

Despite its parameter-efficiency, prompt tuning based approaches increase the sequence length by prepending continuous emeddings in front of the original input sequence (Lester et al., 2021; Mahabadi et al., 2021a). We evaluate the memory footprint of full fine-tuning, Adapter, BitFit, prompt tuning and ATTEMPT. We use T5-base as a default base LM and set the per-gpu batch size to

	memory footprint	
	(base)	(XL)
Fine-tuning	9.0 GB	–
Adapter	5.9 GB	14.5 GB
BitFit	5.6 GB	14.2 GB
Prompt Tuning	8.5 GB	15.9 GB
ATTEMPT (single)	13.7 GB	16.1 GB

Table 9: The maximum memory footprint during training of fine-tuning, Adapter, BitFit, prompt tuning and ATTEMPT(single task).

32. We also compare the memory footprint using T5-3B with batch size of 2. We set the length of the prompt to 100.

As shown in Table 9, ATTEMPT increase the memory footprint from other methods, due to the increase of the input length, multiple pre-loaded source prompts and attention calculations. On the other hand, ATTEMPT shows moderate memory footprint increase when the backbone LM size gets larger (13.7 GB to 16.1 GB) while Adapter and BitFit show about three times more memory footprints than T5-base. This demonstrates that ATTEMPT is more parameter-efficient and can be more memory-efficient when the backbone LMs get even larger (e.g., 11 billions). Moreover, Lester et al. (2021) show that the input prompt length can be significantly reduced when the backbone LMs get larger, which further improve the memory efficiency of prompt tuning-based methods.