Attentional Mixtures of Soft Prompt Tuning for Parameter-efficient Multi-task Knowledge Sharing

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 **ATTEMPT** (**ATTE**ntional **M**ixture of **P**rompt **T**uning), a new modular, multi-task, and parameter-efficient language model (LM) tuning approach that combines knowledge transferred across different tasks via a mixture of soft prompts while keeping original LM unchanged. ATTEMPT interpolates a set of prompts trained on large-scale source tasks and a newly initialized target task prompt using instance-wise attention computed by a light-weight sub-network trained on multiple target tasks. ATTEMPT is parameter-efficient (e.g., updates 1,600 times fewer parameters than fine-tuning) and enables multi-task learning and flexible extensions; importantly, it is also more interpretable because it demonstrates which source tasks affect the final model decision on target tasks. Experimental results across 17 diverse datasets show that ATTEMPT improves prompt tuning by up to a 22% absolute performance gain and outperforms or matches fully fine-tuned or other parameter-efficient tuning approaches that use over ten times more parameters.[1]

1 Introduction

Fine-tuning of large-scale language models (LMs) to get specialized models for specific tasks is known to be the best practice for optimizing task performance (Devlin et al., 2019; Aribandi et al., 2022) but is achieved at the significant cost of training and serving specialized models for many tasks. This motivates recent research on *parameter-efficient tuning* (Houlsby et al., 2019; Li and Liang, 2021; Ben Zaken et al., 2022), which focuses on tuning specialized models by updating a small number of their parameters. Yet, those specialized models fail to benefit from knowledge transfer across many tasks and leverage rich cross-task data (Liu et al., 2019). We posit that multi-task knowledge transfer across tasks and parameter-efficient transfer can benefit from an integrated approach.

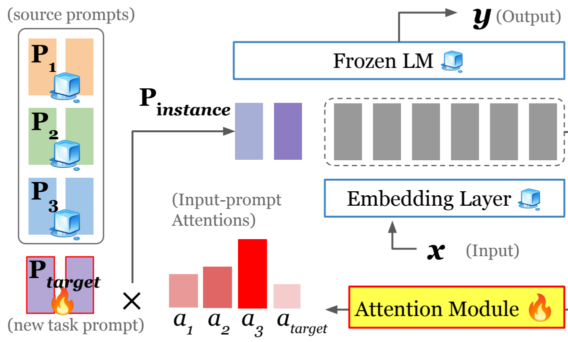


Figure 1: ATTEMPT combines multiple soft prompts trained on large-scale dataset (source prompts) to generate instance-wise target prompts. At training for a target task, the large language model (LM) and the source prompts are kept frozen.

In this work, we introduce a new parameter-efficient, multi-task tuning method called **ATTEMPT** (**ATTE**ntional **M**ixture of **P**rompt **T**uning, previewed in Figure 1). ATTEMPT efficiently combines knowledge from multiple tasks via soft prompts, keeping the original LMs intact. It first trains transferable soft prompts (Lester et al., 2021), called *source prompts*, on large-scale source tasks, which are likely to contain knowledge that can be beneficial to other tasks. Then, for a target task, ATTEMPT initializes a new task prompt and learns an attention-weighted combination of source prompts and the new task-specific prompt. The attention module is light-weight sub-network that is shared across different tasks and trained in a multi-task manner.

ATTEMPT offers three key advantages over previous multi-task fine-tuning or parameter-efficient tuning: first, it is **highly parameter-efficient** and achieves competitive performance despite updating about 0.6% of the parameters updated for full fine-tuning per task. Second, it enables **modular multi-task learning**, where one can reuse and com-

[1] Our code will be available at `https://github.com/AkariAsai/ATTEMPT`.

bine knowledge from multiple different tasks and flexibly add a new task without destroying prior task knowledge. At inference, a single LM with multiple pre-loaded soft prompts specialized for each task can perform many different tasks without parameter reloading. Lastly, it **improves interpretability** on underlying task similarities in multi-task learning by generating attention distributions from input to prompts for each instance.

We conduct experiments on 17 datasets across diverse tasks, domains and output formats. ATTEMPT significantly outperforms previous prompt tuning-based approaches (up to 22%) and matches or exceeds state-of-the-art parameter-efficient transfer approaches or fully fine-tuned models that train orders of magnitude more parameters, especially on smaller datasets. We show that training a single model on multiple tasks often suffers from negative interference or under-fitting to less-represented tasks, especially when the tasks are distant and the number of trainable parameters is limited. However, ATTEMPT enables efficient knowledge transfer and achieves competitive performance for both dominant and less-represented tasks. Our analysis further shows that learned attention weights, multi-task learning and knowledge transfer from multiple tasks largely contribute to the performance improvements. ATTEMPT is particularly parameter-efficient and competitive when the backbone frozen LM is larger, where other parameter-efficient transfer approaches shows rapid increases of the trainable parameters. The attention distributions show the underlying similarities among tasks.

2 Background

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 for the task θ_{task} by gradient updates so that given $\boldsymbol{x}$, the model gives higher probabilities for the ground truth output $\boldsymbol{y}$: $\max_{\theta_{task}} p_{\theta_{task}}(\boldsymbol{y} \mid \boldsymbol{x})$. Fine-tuning can also distort pre-trained features (Kumar et al., 2022; Levine et al., 2022) and is prone to overfitting to small training data (Jiang et al., 2020).

Parameter-efficient tuning. To overcome the inefficiencies of fine-tuning, parameter-efficient tuning updates a small number of parameters for the target task ϕ_{task}: $\max_{\phi_{task}} p_{\theta,\phi_{task}}(\boldsymbol{y} \mid \boldsymbol{x})$, where the number of ϕ_{task} is much smaller than the number of θ_{task}. Recent work also found that it can preserve the rich pre-trained knowledge since it freezes most of θ and avoids the aforementioned forgetting issue (He et al., 2021; Lester et al., 2021). Several approaches introduce additional modules to LMs or directly update a small number of LM parameters; Adapter (Houlsby et al., 2019) and its variants (Mahabadi et al., 2021; Rücklé et al., 2021) insert trainable layers to the LMs for each task, and BitFit (Ben Zaken et al., 2022) updates LM biases. In contrast, prefix-tuning (Li and Liang, 2021) and prompt tuning (Lester et al., 2021) freeze the original LMs and update only trainable *soft prompts* prepended to input. Given the rapidly increasing size of pre-trained LMs (Chowdhery et al., 2022; Brown et al., 2020), efficient fine-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).

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) since the knowledge learned from one task can be useful to another task. Common approaches include multi-task learning on many different tasks (Liu et al., 2019; Aribandi et al., 2022) or transferring a model fine-tuned on a single source task to another target task (Vu et al., 2020; Talmor and Berant, 2019). This often results in negative interference between competing tasks (Zhang et al., 2020; Aghajanyan et al., 2021) and requires massive computational resources to train a joint model on large cross-task data or to search for the best transfer source tasks (Albalak et al., 2022). Moreover, it is difficult to add a new task to multi-task models since updating them on a new task can result in catastrophic forgetting (Kirkpatrick et al., 2017); therefore, models must be retrained on a set of multiple tasks (Biesialska et al., 2020).

3 Method

Problem setup and motivation. Given a collection of source tasks $T_1, \dots T_t$, our goal is to learn a new task T_{target} by efficiently transferring knowledge from the source tasks by updating parameters

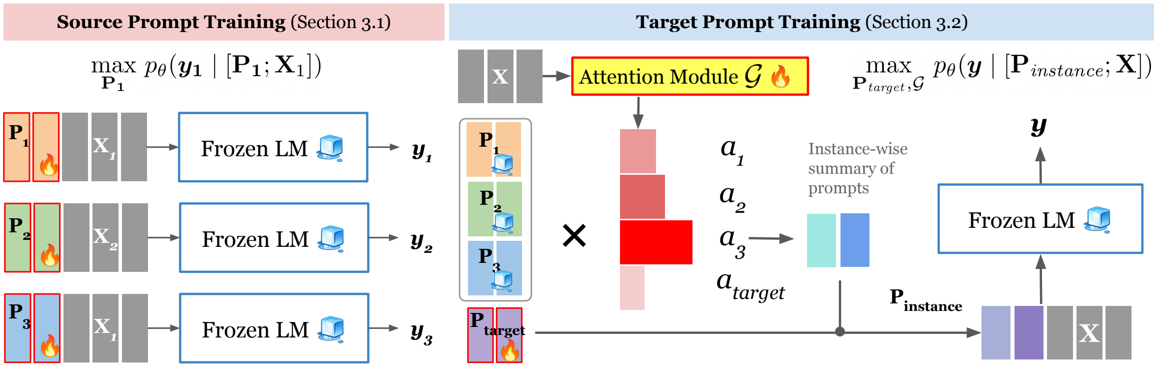


Figure 2: Overview of ATTEMPT. The parts framed in red are updated during training while other parts are instact.

given the target task labeled data $\{(\boldsymbol{x}, \boldsymbol{y})\}$. 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 (e.g., paraphrasing v.s. boolean question answering). We posit that (1) task-specific knowledge can be stored in a small number of parameters ϕ_{task} instead of requiring updating of all LM parameters θ to θ_{task} (*parameter-efficiency*), and (2) multiple task knowledge can be reused, flexibly added and combined to solve a new task, i.e., a (*modular multi-tasking*).

Overview. Our method, called **ATTEMPT** (**ATTE**ntional **M**ixture of **P**rompt **T**uning), is shown in Figure 2 and Algorithm Box 1. It leverages highly parameter-efficient prompt tuning (Lester et al., 2021) to encode knowledge from source tasks into a small number of parameters (*source prompts*). ATTEMPT first pre-trains a set of source prompts $\mathbf{P}_1, \ldots, \mathbf{P}_t$ for source tasks $T_1, \ldots, T_t$ (Section 3.1; left side of Figure 2). For a target task, it first initializes a target-task-specific prompt $\mathbf{P}_{target}$ and an attention module $\mathcal{G}$. Given an instance $(\boldsymbol{x}, \boldsymbol{y})$, it computes attentions between embedded input $\mathbf{X}$ and the prompts (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 $\boldsymbol{x}$ as in the original prompt tuning.

Training ATTEMPT on multiple target tasks. Unlike other parameter-efficient tuning approaches, prompt or prefix tuning can train task-specific parameters θ_{task} for different tasks in the same mini-batch (Li and Liang, 2021; Lester et al., 2021). Leveraging this advantage, we can train a shared attention module $\mathcal{G}$ and multiple target task prompts simultaneously on a collection of different tasks, which allows ATTEMPT to further transfer knowledge across different *target* tasks and improve parameter and inference efficiency (Section 3.3).

3.1 Source Prompt Training

We first obtain a soft prompt, a small trainable embedding in front of the input embeddings following Lester et al. (2021), which is described in detail below. We run prompt tuning on several high-resource datasets, such as Multi-NLI (Williams et al., 2018) or SQuAD (Rajpurkar et al., 2016) and use the resulting soft prompts as *source prompts* to be transferred to many different target tasks. Unlike prior work, such as Vu et al. (2022), that uses other tasks' prompts to initialize prompts for new tasks, we keep the source prompts intact and transfer them in a non-destructive manner.

Formally, given t source tasks, we train prompts on each task and acquire t source prompts, $[\mathbf{P}_1, \ldots, \mathbf{P}_t]$, where $\mathbf{P}_j$ corresponds to the j-th source task. The training details of source prompts are as follows: Let input $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$, where l is the length of the input sequence, $\mathbf{x}_i \in \mathbb{R}^d$ corresponds to the i-th token embedding of the input sequence $\boldsymbol{x}$, and d is the LM dimension. A soft prompt is represented as $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_m] \in \mathbb{R}^{m \times d}$, where m corresponds to the length of the prompt embeddings (e.g., $m = 100$), and is often specific to each task. Input embeddings prepended by the prompt $[\mathbf{P}; \mathbf{X}]$ are fed into the frozen LM θ. During training, only prompt embeddings will be updated by maximizing the likelihood of generat-

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 1: Training process of ATTEMPT.

ing the target sequence $\boldsymbol{y}$, as follows:

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

3.2 Target Prompt Training

For a new target task T_{target}, we first initialize a target-task-specific prompt $\mathbf{P}_{target}$ and attention module $\mathcal{G}$. For each instance in the target task $(\boldsymbol{y} \mid \boldsymbol{x})$, ATTEMPT produces instance-wise soft prompt $\mathbf{P}_{instance}$ by interpolating the source prompts and newly initialized target-task-specific prompt (Section 3.2.2) given attention scores generated by $\mathcal{G}$ (Section 3.2.1).

$$\mathbf{P}_{instance} = \mathbf{P}_{target} + \mathcal{G}(\mathbf{X}, [\mathbf{P}_1, \ldots, \mathbf{P}_t, \mathbf{P}_{target}]).$$

Similar to the original prompt tuning in Eq. 1, we then concatenate the produced instance-wise prompt to the input and train ATTEMPT by maximizing the likelihood, as follows:

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

During training, this new task prompt $\mathbf{P}_{target}$ as well as the attention module $\mathcal{G}$ are updated via $\mathbf{P}_{instance}$, while source prompts (Section 3.1) and the original LM are untouched.

3.2.1 Input-prompt Attentions

ATTEMPT controls the influence of the set of source prompts on the final instance-wise prompt by calculating input-prompt attentions. Since the input embeddings $\mathbf{X} \in \mathbb{R}^{l \times d}$ 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 $\hat{\mathbf{P}}_j \in \mathbb{R}^d$.

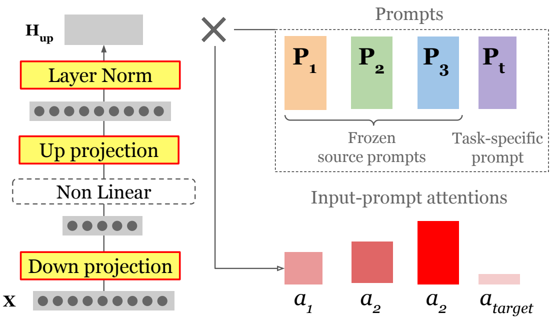


Figure 3: The overview of our attention module for calculating input-prompt attentions.

We then feed $\hat{\mathbf{X}}$ to a sub-network (shown on the left side of Figure 3) to project it into the prompt spaces. For efficiency, ATTEMPT first feeds input embeddings $\hat{\mathbf{X}}$ to a sub- network consisting of down and up projection, as follows:

$$\mathbf{H}_{down} = \mathbf{W}_{down}^{\top}(\hat{\mathbf{X}})$$
$$\mathbf{H}_{up} = \mathbf{W}_{up}^{\top}(\mathrm{NonLinear}(\mathbf{H}_{down})),$$

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. We 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 $\mathbf{H}_{up}$, and apply softmax over the prompts, as follows:

$$a_j = \frac{e^{\hat{\mathbf{P}}_j \mathbf{H}_{up}}}{\sum_{k=1}^{t+1} e^{\hat{\mathbf{P}}_k \mathbf{H}_{up}}}. \tag{3}$$

We introduce a softmax temperature (Radford et al., 2021), and scale the logits in Eq. 3 by $1/d \times \exp(K)$ to avoid making the attention module over-confident on specific source tasks, where K is a temperature hyperparameter.

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 in the same task, while the $\mathbf{P}_{target}$ term is task-specific. The attentions act as gating to control the influences from different prompts and let us flexibly compose knowledge from multiple tasks. As seen in Eq. 4, it gives a $1 + a_{t+1}$ weight to the target-task-specific prompt $\mathbf{P}_{target} (= \mathbf{P}_{t+1})$. This enables ATTEMPT to *ignore* the source prompts if the knowledge from none of the sources tasks is useful to solve the instance $\mathbf{X}$, while always keeping the influence of $\mathbf{P}_{target}$ so that it will be properly updated during training.

3.3 Mixed-task Mini-batch Training

As discussed above, ATTEMPT can jointly train the attention module and multiple target task prompts. For mixed-task mini-batch multi-task training, we first concatenate 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. By sharing the attention module, different target tasks are loosely connected and together contribute to an improved and task-agnostic attention module, particularly effective when the target task training data is small. Moreover, sharing the attention layer weights enables us to reduce the number of parameters to be updated per task and improves the efficiency of inference time since ATTEMPT loads the shared attention weights and prompts one time for all target tasks and perform different tasks.

We also 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.

4 Experiments

We run experiments on 17 datasets across diverse tasks, domains and output formats.

4.1 Source and Target Tasks

We use 6 large-scale datasets as *source tasks*, and evaluate on 17 diverse *target tasks* across entailment, paraphrase detection, sentiment analysis, question answering, commonsense reasoning, scientific reasoning. There is no overlap between source and target tasks. The details of the datasets used in this work are in Appendix Section B.3.

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., 2018), SST-2 (Socher et al., 2013), SQuAD (Rajpurkar et al., 2016), and ReCoRD (Zhang et al., 2018).

GLUE and SuperGLUE. We use the following GLUE (Wang et al., 2019b) and 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), COPA (Roemmele et al., 2011), MultiRC (Khashabi et al., 2018), RTE (Giampiccolo et al., 2007), WiC (Pilehvar and Camacho-Collados, 2019), WSC (Levesque et al., 2012), CoLA (Warstadt et al., 2019), STS-B (Cer et al., 2017) and WNLI (Wang et al., 2019b).

Question answering. We use the MRQA 2019 shared task (Fisch et al., 2019) data and experiments on four large-scale QA datasets: Natural Questions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), NewsQA (Trischler et al., 2017) and SearchQA (Dunn et al., 2017). Although those datasets often have long input, we cut off it at the maximum token length of 512.

Others. In addition to the datasets from GLUE, SuperGLUE and MRQA, we test our models on four different datasets, whose tasks are related to the source tasks but the domains differ. SciTail (Khot et al., 2018) is an entailment dataset in the scientific domain. Yelp-2 polarity (Zhang et al., 2015) is a sentiment analysis dataset that uses Yelp data. WinoGrande (Sakaguchi et al., 2020) is a commonsense reasoning dataset in a multiple choice format. PAWS-Wiki (Zhang et al., 2019) is a paraphrase detection dataset that uses Wikipedia.

4.2 Baselines

Single-task baselines. Single-task baselines train specialized models or parameters for each target task. We compare our model with: **fine-tuned (FT)** models; **prompt tuning** (**PT**; Lester et al. 2021), where source prompt embeddings are initialized by randomly sampled top vocabularies; and **SPoT-*generic*** **(SPoT-g)** and **SPoT-*targeted*** (**SPoT-t**; Vu et al. 2022). SPoT-*generic* first pre-trains source prompts on eight GLUE tasks and then uses the source prompts to initialize target task prompts, while SPoT-*targeted* uses prompt similarities to find top-k similar tasks and then initializes target task prompts using the weighted average of the top k prompts. We also compare our models with several widely used parameter-efficient tuning approaches. **Adapter** (Houlsby et al., 2019) inserts adapter layers in the transformer block. **BitFit**, on the other hand, freezes the weights and updates only the bias terms of the LMs.

Multi-task baselines. Multi-task baselines share all or part of the task-specific parameters across multiple target tasks to enhance knowledge sharing across different tasks. **FT-MT**, **Adapter-MT** and **SPoT-{g,t}-MT** train a single model on a set of target tasks. **Ours-MT**, on the other hand, still keeps separate target task-specific embeddings but jointly trains the attention module on the same set of training data. Due to significant differences in input context length and data size, we divided 16 tasks except for MultiRC into four categories and conduct multi-task training on four different tasks. Section B.6 in Appendix contains more details.

4.3 Experiment Settings

Unless specified, we use T5-base as our base LMs.[2] Following Mahabadi et al. (2021), if a dataset does not have public test split with annotations, we use development set as our test set or split the development set into our development and test sets. We train all of the baselines 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. We limit the maximum training data number of Yelp-2 to be 100k samples. We run inferences on the test data using the model with the best development performance. We set prompt tuning learning rate to be 0.3 while the learning rate for the attention module is tuned on the development sets. We use $K = 1$ to control the soft max temperature in Section 3.2. We set the maximum token length to be 512 for MRQA datasets, 348 for MultiRC and 256 for all of other datasets. The prompt length m is set to be 100.

5 Results

ATTEMPT outperforms prompt tuning by a large margin (up to 22%) on most of datasets and often matches or outperforms other parameter-efficient or fully fine-tuning approaches that train orders of magnitude more parameters. Table 2 shows the results for the GLUE and SuperGLUE datasets, and Table 3 presents the results of the other datasets.

ATTEMPT significantly improves prompt tuning and SPoT. Experimental results on Table 2 clearly show that on most of the datasets, it significantly outperforms prompt tuning, SPoT-t and SPoT-g. As pointed out by prior work (Mahabadi et al., 2021; Lester et al., 2021; Li and Liang, 2021), prompt tuning is sensitive to hyperparameters or initialization, and it has significantly lower performance on several datasets such as BoolQ (61.71%)

[2] 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., 2021).

# of train data	params / task	WNLI (634)	STS-B (7k)	MRPC (3.7k)	RTE (2.5k)	MultiRC (5.1k)	BoolQ (9.4k)	WiC (6k)	WSC (554)	CB (250)	Avg.
Single-task training											
FT	220M	50.00	89.68	90.20	71.94	72.77	81.10	70.22	59.61	85.71	74.58
Adapter	1.8M	50.00	90.73	85.29	71.94	75.87	82.45	67.08	67.30	85.71	75.17
BitFit	280k	50.00	90.88	86.76	67.63	74.51	79.57	69.59	59.61	78.57	73.01
PT	77k	50.00	89.48	68.14	54.68	58.73	61.71	48.90	51.92	67.86	61.28
SPoT-g	77k	58.33	90.69	87.52	71.94	73.97	77.24	67.00	50.00	46.42	69.23
SPoT-t	77k	52.78	90.00	79.72	69.78	74.21	71.68	48.90	53.84	71.43	68.03
ATTEMPT-ST	232k	55.56	89.67	85.74	73.38	74.39	77.06	66.77	53.84	78.57	74.06
Multi-task training											
FT-MT	55M	27.78	89.25	86.10	75.53	72.77	79.32	69.59	63.46	71.42	70.58
Adapter-MT	450k	50.00	84.00	85.28	70.5	75.87	81.89	65.20	28.84	78.57	68.90
SPoT-g MT	19k	50.00	49.81	66.28	46.04	73.97	61.59	50.78	32.69	46.42	53.06
ATTEMPT-MT	134k	50.00	90.33	85.17	76.97	74.39	78.29	66.46	69.23	82.14	74.78

Table 2: Results on GLUE and SuperGLUE. "FT," "PT," "SPoT-g," and "SPoT-t" denote fine-tuning, prompt tuning (randomly initialized prompts), SPoT-general, and SPoT-target, respectively. ATTEMPT-{ST, MT} is our method.

# of train data	params / task	NQ (100k)	HP (72k)	SQA (117k)	News (74k)	Avg.	WG (40k)	Yelp (100k)	SciTail (27k)	PAWS (49k)	Avg.
Single-task training											
FT	220M	75.12	77.46	81.14	65.23	74.73	61.87	96.65	95.78	94.13	87.11
Adapter	1.8M	74.20	77.56	81.39	65.64	74.69	59.21	96.85	94.47	94.25	86.20
BitFit	280k	70.66	75.47	77.68	64.09	71.98	57.22	94.66	94.71	92.03	84.66
PT	77k	67.94	72.85	75.68	61.08	69.38	49.57	95.12	87.88	55.77	72.09
SPoT-t	77k	68.18	74.76	75.27	58.22	69.11	50.43	95.41	91.20	91.13	82.04
ATTEMPT-ST	232k	70.39	75.17	77.29	62.78	71.41	57.61	96.66	93.10	92.06	84.86
Multi-task training											
FT-MT	55M	74.10	76.99	78.71	61.84	72.91	60.22	97.06	92.02	93.01	85.58
Adapter-MT	450k	74.59	78.79	80.64	66.16	75.04	57.93	96.82	96.62	93.82	86.30
SPoT-t MT	19k	68.51	72.21	73.39	61.58	68.93	51.67	95.64	91.41	91.60	82.58
ATTEMPT-MT	134k	71.82	75.42	77.50	63.46	72.05	58.56	96.21	94.63	92.79	85.55

Table 3: Results on MRQA 2019 QA datasets, WinoGrande (WG), Yelp, Scitail and PAWS. "NQ", "HP," "SQA," and "News" denote Natural Questions, HotpotQA, SearchQA, News QA, respectively.

or WiC (48.90%) than other parameter-efficient transfer approaches. Even on PAWS with 49k training data, it shows a low accuracy (55%), though all other baselines achieve a higher than 90% accuracy due to the simplicity of the task. As shown by the results of SPoT-t or SPoT-g, initializing a target task prompt with a source prompt trained on other data often alleviates this issue, which is consistent with the finding of Vu et al. (2022). However, there remains a large performance gap between other parameter-efficient approaches and those prior prompt tuning-based approaches. Also the improvements given by better initialization is limited when the training data is large, as shown in Table 3. ATTEMPT shows significant performance improvements over those approaches on smaller datasets (e.g., CB, RTE), as well as large datasets such as MRQA 2019 QA dataset and SciTail.

ATTEMPT-MT improves performance particularly on smaller datasets. Despite using less per-task trainable parameters, ATTEMPT-MT outperforms ATTEMPT-ST across many of the target datasets. The gains are larger on smaller datasets (e.g., 3.57% and 9.39 % accuracy improvements on CB and WSC). We hypothesize that sharing attention module is particularly helpful on smaller datasets, since learning good input-prompt attentions from limited training sample is hard.

ATTEMPT matches methods that update more parameters. On GLUE and SuperGLUE (Table 2), ATTEMPT-MT yields average of 74.78, outperforming BitFit using two times more per-task trainable parameters and performing on par with Adapter and fine-tuning approaches, which train and store 14 times and 1,600 times more parameters than ATTEMPT, respectively. As shown in Table 3, ATTEMPT-MT achieves 72.05 average F1, which again outperforms BitFit using twice as many parameters and yield 85.55% average accuracy on WinoGrande, Yelp, SciTail and PAWS, significantly outperforming BitFiT (84.66%) and matching Adapter (86.20%). In addition to better trade-off between task performance and parameter-efficiency, ATTEMPT provides unique properties. ATTEMPT-MT keeps the original LMs completely untouched and loads multiple target prompts and the shared attention module simultaneously, allowing it to perform many different tasks without model reloading at inference time. Moreover, ATTEMPT also shows marginal increases in the number of trainable parameters when the backbone LM size increases, while other methods show drastic increase of trainable parameters. We provide detailed discussion on this in Section 6.2.

Multi-task learning in ATTEMPT v.s. other fine-tuning models. Although multi-task pre-training has shown its effectiveness, fine-tuning a single model or parameters on a combination of many different tasks often degrades performance (Aghajanyan et al., 2021), especially with parameter-efficient transfer approaches, where we have limited parameters to be shared (Karimi Mahabadi et al., 2021; Ivison and Peters, 2022). SPoT-t MT and FT-MT show performance drops from their single-task variants across different datasets. Even among the MRQA 2019 shared tasks where all four target tasks are more related than SuperGLUE or GLUE, the performance drop still occurs; SPoT-t MT shows a larger performance decline on HotpotQA and SearchQA (2.55 and 1.88 F1 drop, respectively), and FT-MT shows a 3.39 F1 decline on NewsQA. This indicates that even among similar tasks, negative interference remains. Although Adapter-MT shows better performance than its single-task version on MRQA 2019, it also shows declining performance on other datasets.

Table 2 shows larger performance deterioration occur on smaller datasets, probably because those datasets are less represented during multi-task training. SPoT-g MT, especially, suffers from negative interference between different tasks (e.g., SPoT-g WiC: 67.0% $\rightarrow$ SPoT-g MT: 50.78% on WiC), indicating that sharing a single prompt for multiple diverse tasks is challenging. In contrast, ATTEMPT-MT keeps the target prompts separate while sharing the attention module to efficiently run multi-tasking and shows performance improvements over our ATTEMPT-ST as discussed above.

	BoolQ	NewsQA	WG
ATTEMPT-ST	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
no prior transfer	77.03	60.95	57.22
ATTEMPT-MT	78.29	61.58	58.57
no prior transfer	78.29	61.44	56.98

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

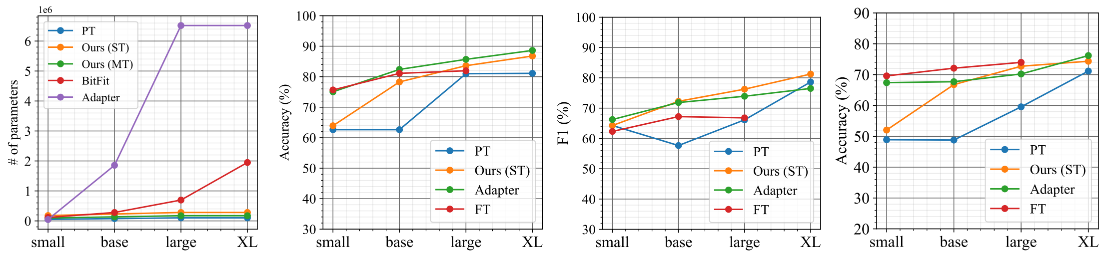
6 Analyses

We conduct additional experiments and analyses to shed light on the following topics: 1) the factors that contribute to the performance improvements of ATTEMPT (Section 6.1), 2) the degree of benefit by increasing the size of the backbone LM (Section 6.2), and the process by which the knowledge combination though attentions occurs (Section 6.3).

6.1 Ablation Studies

We compare different variants of ATTEMPT to see the effect of each of the design choices. 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-trained source prompts; (b) *no attention*, which removes the learned attention layers and gives the constant score $a_j = 1/t$ to all source prompts in Eq. 3; (c) *single prompt*, which uses only a single source prompt in Eq. 4 to assess the effect of transferring knowledge from multiple tasks; and (d) *no prior transfer*, which does not transfer attention generator weights from source tasks. The ablations are mainly conducted on the top of the single-task model (ATTEMPT-ST), and we conduct ablation (d) on ATTEMPT-MT, as well. We sample three tasks—BoolQ, NewsQA and WinoGrande—for the ablation studies.

Table 4 reveals that all components contribute to performance improvements. Regarding the fundamental modeling choice, adding a trainable target-task-specific prompt (no target) is crucial to achieve good performance on all of the datasets, especially



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

Figure 4: Scalability Experiments. (a) shows the number of the parameters with different backbone LMs. (b), (c) and (d) shows the performance of fine-tuning, prompt tuning, Adapter and ATTEMPTusing T5-{small, base, large, 3B (XL)} on BoolQ, MultiRC and WiC, respectively. Ours in figures denotes ATTEMPT.

on BoolQ and WinoGrande. This demonstrates that unless the source tasks are highly related to a target task, learning a new target task without task-specific prompt is challenging. No attention (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. Single prompt ablation is similar to SPoT-t except that instead of using source prompts for initialization, we keep the source prompt frozen while updating the target task prompt and the attention layers. Although this variant outperforms SPoT-t and demonstrates the effectiveness of keeping prior source prompts frozen, there is notable performance decline relative to ATTEMPT-ST, suggesting that combining multiple source prompts is effective. Pre-training $\mathcal{G}$ on the combinations of source tasks shows improvements on the NewsQA single-task and the WinoGrande multi-task.

6.2 Parameter Efficiency and Power of Scale

Analysis on parameter-efficiency of ATTEMPT. For each task, we will introduce a new trainable soft prompt embeddings $m \times d$, 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. As this can be shared across N target tasks, the additional parameters per task will be: $d \times m + \frac{2rd+2d}{N} = d(m + 2(r+1)/N)$.

Figure 4a shows the number of the parameters to be updated per task when we use different backbone LMs. With Adapter or fine-tuning, the number of the parameters quickly increases, but ATTEMPT 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 larger backbone LMs. Moreover, even if we use T5-large as the backbone model, the number of the parameters to be trained and stored is still smaller than the ones of adapter or BitFit using a smaller LM (i.e., T5-base). Given that recent LMs are growing larger and larger, the ability to induce the rich knowledge learned from pre-training using small number of parameters is important, and we argue that prompt tuning-based approaches, which keep the original LMs intact and is independent from the number of the LM layers, can be particularly effective.

Scaling backbone LMs. We also empirically analyze the effect of increasing the backbone LM sizes. We evaluate Adapter, ATTEMPT-ST, prompt tuning and fully fine-tuned model performance on three SuperGLUE datasets: BoolQ, MultiRC and WiC. Lester et al. (2021) show that prompt tuning is particularly effective when the backbone LM is larger while it struggles when used with a smaller LM, as also found in our experiments. 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.

Figure 4b, Figure 4c and Figure 4d show the performance on BoolQ, MultiRC, and WiC using different backbone LMs (T5-small, T5-base, T5-large, and T5-XL). As we can see, ATTEMPT largely benefits from backbone LM size increase. ATTEMPT outperforms prompt tuning across different tasks and backbone LMs, and matches or outperforms fully fine-tuned models with T5-large. Furthermore, our model also perform on par with

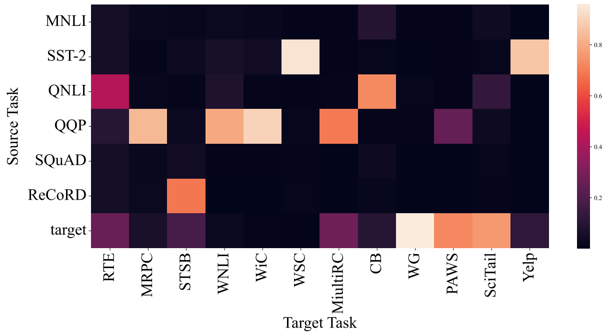


Figure 5: Attention visualizations of ATTEMPT-ST.

Adapter with T5-3B, while updating and storing much smaller numbers of parameters (Figure 4a).

6.3 Analysis on Attention

Figure 5 shows the attention weight matrix between source and target tasks, produced by ATTEMPT-ST. Attention patterns differ for different tasks. Generally, $\mathcal{G}$ gives higher attentions to related source tasks. For instance, $\mathcal{G}$ gives significantly higher attentions to SST-2 for Yelp, or QQP for PAWS-Wiki, 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. 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 distributions, and $\mathcal{G}$ ignores source prompts more often.

7 Additional Related Work

Parameter-efficient transfer learning. In addition to the approaches discussed in the previous sections (Houlsby et al., 2019; Ben Zaken et al., 2022; Li and Liang, 2021; Lester et al., 2021; Vu et al., 2022), many parameter-efficient transfer approaches have been introduced recently. AdapterFusion (Pfeiffer et al., 2021) attempts to compose multiple different adapters by learning task-specific compositions on each task after training adapters individually. Qin and Eisner (2021) also introduce data-dependent mixture of soft prompt modeling, where they ensemble predictions given different prompts. ATTEMPT directly produces instance-wise prompts by interpolating multiple prompts, removing the necessity of running multiple forward passes for a single input. IDPG (Wu et al., 2022) and Instance-Dependent Prompt Tuning (Levine et al., 2022) learn to generate instance-wise prompts given encoded input. Particularly Instance-Dependent Prompt Tuning tries to achieve parameter-efficient multi-task learning and trains the generator component in a multi-task manner. Our main focus is transferring knowledge from multiple tasks in a non-destructive manner rather than learning to generate them from scratch, and also much more parameter-efficient. Concurrent to our work, Liu et al. (2022) introduce $(IA)^3$ that multiplies intermediate activation by learned vectors for parameter-efficient few-shot learning. HyperFormer (Karimi Mahabadi et al., 2021) and HyperDecoder (Ivison and Peters, 2022) train a shared hyper network to generate adapters for parameter-efficient multi-task learning.

Modular approaches and sparse networks. 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 works try to sparsely activate or employ light-weight modules for efficient multi-task learning (Gupta et al., 2022; Ponti et al., 2022; Fedus et al., 2021). 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, our approaches keep LMs intact and update less parameters.

8 Conclusion

In this work, we introduce ATTEMPT, a new modular, multi-task and parameter-efficient tuning approach for efficient knowledge transfer. ATTEMPT produces instance-wise soft prompts by interpolating multiple reusable soft prompts trained on source tasks and a new task-specific prompt, while keeping the original LM frozen. Our experiments show that ATTEMPT largely improves previous prompt tuning-based approaches, and matches or outperforms prior parameter-efficient transfer approaches or fully fine-tuned models despite using much smaller number of parameters to be updated. Our analysis shows that transferring knowledge from multiple tasks and sharing attention modules largely contribute to performance improvement.

Acknowledgement

This research was supported by NSF IIS-2044660, ONR N00014-18-1-2826, the Allen Distinguished Investigator Award, the Sloan Fellowship, 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 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alon Albalak, Yi-Lin Tuan, Pegah Jandaghi, Connor Pryor, Luke Yoffe, Deepak Ramachandran, Lise Getoor, Jay Pujara, and William Yang Wang. 2022. FETA: A benchmark for few-sample task transfer in open-domain dialogue. *arXiv preprint arXiv:2205.06262*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

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 *International Conference on Learning Representations*.

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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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. *Advances in neural information processing systems*, 33:1877–1901.

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

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

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. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

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*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

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*, pages 1–9, Prague. Association for Computational Linguistics.

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*.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2208–2222, Online. Association for Computational Linguistics.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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*, 15(2):219–250.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991b. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pretrained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

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

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

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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

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. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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

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*, KR'12, page 552–561. AAAI Press.

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*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

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

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*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

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 *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

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

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 *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

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 *International Conference on Machine Learning*, pages 8748–8763. PMLR.

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

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

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

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 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

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 *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

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

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

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.

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. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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 *International Conference on Learning Representations*.

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

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

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

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

Appendix

A More Method Details

A.1 Pre-training $\mathcal{G}$ on source tasks.

To pre-train $\mathcal{G}$, 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.

B Experimental Details

B.1 Details of the Attention Module Pretraining

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).

B.2 Implementation Details

We use PyTorch[3] (Paszke et al., 2019) and huggingface `transformers`[4] library to implement our models. For Adapter, BitFit, prompt tuning and BitFit baselines, we use the implementations by Mahabadi et al. (2021).[5] We use huggingface `datasets`[6] library to use the data for the experiments except for MRQA 2019 shared task. For MRQA 2019 shared task, we download the original training and development data from the official repository.[7]

B.3 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.

[3] https://pytorch.org/

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

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

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

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

B.4 Hyperparameters for ATTEMPT

General hyperparameters. We optimize the objective function using Adam (Kingma and Ba, 2015) with learning rate 0.3 for prompt tuning following Lester et al. (2021). All of the experiments are conducted with a single GPU with 24 GB memory. 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. We set the number of the prompt to be 100 throughout our experiments. We set weight decay to be 1×10^{-5}. For the projection layers, we use $r = 100$.

Learning rates for attention modules. 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.

Target-task prompt initialization. To initialize the target task specific prompt, $\mathbf{P}_{target}$, we use the Multi NLI source prompt for classification task and the SQuAD source prompt for question answering, instead of initializing the target-task-specific prompt with randomly sampled vocabularies for training stability.

B.5 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}.

SPoT-g and 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

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 5: The details of the 6 source tasks.

Dataset Name	Category	Task	Domain	Metric
1. WNLI	GLUE	coreference / NLI	fiction books	accuracy
2. STS-B	GLUE	sentence similarity	misc.	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 6: The details of the 17 target tasks. "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.

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

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), and for those datasets, we use SPoT-t as a primary baseline.

Adapter. We use the default hyperparameters by Mahabadi et al. (2021) 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 $\times 10^{-4}$. Other hyperparameters are the same as the hugging-face transformers T5 models.

B.6 Multi-task Training Details

As discussed in Section 4.1, the 17 datasets have significantly different length of input context, and training on the combinations of all of the datasets can make training inefficient. In this work, we conduct multi-tasking of 4 datasets (GLUE, SuperGLUE, MRQA 2019, and others), while keeping SuperGLUE MultiRC training separated, as MultiRC has significantly longer context 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.

B.7 Task and Dataset Details

We show the list of the datasets, tasks and domains for source tasks in Table 5 and for target tasks in Table 6, respectively. In summary, both source and target datasets cover diverse tasks, domains and output formats (i.e., span extraction, multiple-choice, classification).

C Details for the Scaling Experiments

C.1 Experimental Details

We provide the experimental details for the LM scaling experiments in Section 6.2. 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.

Maximum token length and the number of the training epochs. 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 7.

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 7. 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.

C.2 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).