Prototype-based HyperAdapter for Sample-Efficient Multi-task Tuning

Hao Zhao[1]* **Jie Fu**[2]*† **Zhaofeng He**[1]†
[1]Beijing University of Posts and Telecommunications [2]Hong Kong University of Science and Technology
{haozhao,zhaofenghe}@bupt.edu.cn jiefu@ust.hk

Abstract

Parameter-efficient fine-tuning (PEFT) has shown its effectiveness in adapting the pre-trained language models to downstream tasks while only updating a small number of parameters. Despite the success, most existing methods independently adapt to each task without considering knowledge transfer between tasks and are limited to low-data regimes. To overcome this issue, we propose Prototype-based HyperAdapter (PHA), a novel framework built on the adapter-tuning and hypernetwork. It introduces an instance-dense retriever and a prototypical hypernetwork to generate the conditional modules in a sample-efficient manner. This leads to comparable performance improvements against existing PEFT methods on multi-task learning and few-shot transfer learning. More importantly, when the available data size gets smaller, our method outperforms other strong baselines by a large margin. Based on our extensive empirical experiments across various datasets, we demonstrate that PHA strikes a better trade-off between trainable parameters, accuracy on stream tasks, and sample efficiency. Our code is publicly available at https://github.com/Bumble666/PHA

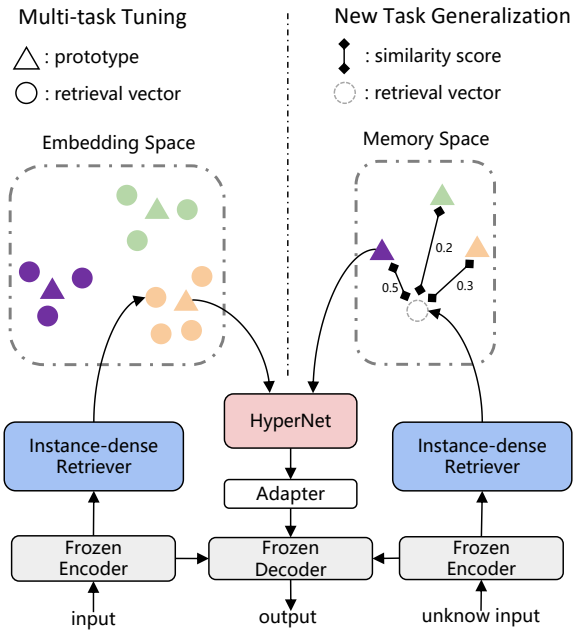

Figure 1: Prototype-based HyperAdapter generates task-specific adapter with a shared hypernetwork and prototypes calculated by task-specific instances. During generalization, the relevant task prototype is retrieved by the Instance-dense Retriever, and the new adapter layers are generated by hypernetwork.

1 Introduction

Fine-tuning a pre-trained language model (PLM) yields extraordinary potential for simultaneous adaptation to multiple downstream tasks in a multi-task setting. However, fine-tuning all the parameters of models induces substantial storage and deployment costs, especially as pre-trained model sizes are growing rapidly. To address this issue, several works (Houlsby et al., 2019; Lester et al., 2021; Karimi Mahabadi et al., 2021a; Hu et al., 2022; Ding et al., 2022; Gui and Xiao, 2023; Zeng et al., 2023; Liao et al., 2023; Xie and Lukasiewicz, 2023) have developed parameter-efficient fine-tuning which trains compact modules per task and adapts PLMs to downstream tasks. Nonetheless, these methods require learning different modules to adapt to diverse tasks, and the cost of the parameter increases proportionally with the number of tasks. On the other hand, training task-specific modules separately fails to reap benefits from other relative tasks.

Recent work (Karimi Mahabadi et al., 2021b; Ivison and Peters, 2022) has proposed training a hypernetwork to generate the parameters of these modules to achieve a better trade-off between parameter efficiency and adaption for downstream tasks. These methods encourage the multi-task learning model to capture the shared information by leveraging task-shared hypernetwork while eliminating negative task interference by generating con-

*Equal technical contribution, co-first authors.
†Corresponding authors.

ditioned modules individually. Despite these methods' success in multi-task learning, there are still some issues:(1) hypernetwork-based methods generally optimize the specific embedding and shared hypernetwork together by end-to-end training without any regularization. Task-specific information is inseparably intertwined, which suppresses the efficiency of the hypernetwork, especially in resource-limited settings. (2) these existing approaches generalize to new tasks that require task-specific prior knowledge or knowledge from frozen pre-trained models.

These works (Karimi Mahabadi et al., 2021b; Pfeiffer et al., 2023) indicate that the task-shared hypernetwork serves as a cross-task information captor, while a specific embedding should encapsulate task-level semantic features in order to extract pertinent information from the hypernetwork for generating corresponding module parameters. Empirically, task-level features are typically implicitly represented by related instance features. A natural idea to encourage embedding generation is to calculate the central points (prototypes) of task-specific instance features.

In this paper, we introduce the Prototype-based HyperAdapter(PHA), a novel framework built on adapter-tuning that achieves both multi-task learning and generalization to new tasks in a sample-efficient manner. As depicted in Figure 1, PHA consists of two main components, Instance-dense Retriever and Prototypical HyperNetworks. The first part aims to train a retriever to discriminate the instances from different tasks in embedding space. For the second part, we aim to estimate the task-specific prototypes with the instance-level features and keep the prototypes as embeddings to be trained with the hypernetwork.

Specifically, we project the encoded instance features into embedding space using the retriever. To avoid instances interference in embedding space, we train the retriever with the InfoNCE estimator (Oord et al., 2018). As a result, it clusters intra-task instances and increases the distances between inter-task instances. The projected features here can be deemed as instance-level semantic features that are used to estimate task-level embeddings. Inspired by PCL (Li et al., 2021), we estimate the task-specific embedding using the contrastive prototypical loss, which encourages the prototypes to become the center points of instance-level features. Compared with the existing method, where the specific embeddings are optimized directly during tuning, our method efficiently learns specific embedding with side information, which helps to optimize the embedding space in low-data regimes. During the adaptation of new tasks, since we maintain the previous task-level semantic features as prototypes that align with the instances in the embedding space, we match the corresponding prototype for the current new task by calculating the distance between the new instances and the previous prototypes.

We evaluate PHA on 13 NLP datasets across diverse tasks. Extensive experiments show the effectiveness of PHA, especially in low-data regimes. Meanwhile, PHA is able to achieve a few-shot domain adaption with 4-32 shots. For example, PHA outperforms the strong multitask adapter transfer baseline by 1.0% with lower trainable parameters on the GLUE benchmark. In low resource regimes where only 100 samples per task from the GLUE benchmark is available, PHA outperforms adapter-tuning by 8.0%. Our analysis shows that PHA efficiently captures specific information and shared information while reducing negative transfer. We also present a detailed analysis to demonstrate that the instances from different tasks can be identified by corresponding prototypes and used for new task adaption.

2 Background

HyperNetworks. A hypernetwork(Ha et al., 2017) can generate parameters to be used by networks or modules. Specifically, the hypernetwork, denoted as h_w, leverages an embedding $\boldsymbol{I}$ to generate the module parameters $\boldsymbol{\phi}$:

$$\boldsymbol{\phi} = h_w(\boldsymbol{I}) \tag{1}$$

Adapter. Adapter(Houlsby et al., 2019) is commonly used in parameter-efficient tuning that aims to apply PLM to downstream tasks. Specifically, adapter-based tuning inserts trainable task-specific modules into transformer layers, while keeping the PLM fixed. The adapter $A^l(x)$ for layer l is defined as:

$$A^l(\boldsymbol{x}) = \boldsymbol{D^l}(\text{ReLU}((\boldsymbol{U^l}(x))) + x, \tag{2}$$

where $\boldsymbol{D^l} \in \mathbb{R}^{d\times b}$ and $\boldsymbol{U^l} \in \mathbb{R}^{b\times d}$ are down/up-projection matrices. $x \in \mathbb{R}^d$ refers to the input. d is the hidden dimension of PLM, b is the bottleneck size satisfying $b \ll d$.

He et al. (2022a) propose using the parallel adapter, which is different from the traditional sequential insertion method. They demonstrate the more efficient parameter-efficient fine-tuning that the parallel adapter offers.

3 Method

Problem Setup. We are following a general multi-task learning problem. Given a pre-trained language model $\mathcal{M}_\theta$ with parameters θ and a set of target tasks $\{D\} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\tau\}$, where τ is the total number of tasks and $\{\mathcal{D}_i\} = \{x_i^n, y_i^n\}_{n=1}^{N_i}$ represents the training data of the i-th task with N_i samples. The main objective of our study is to fine-tune $\mathcal{M}_\theta$ for downstream tasks $\{D\}$ using a multi-task learning setup and to ensure that it is capable of generalizing to new tasks.

Method Overview. The key idea of our approach is to directly learn prototype embeddings for each task from training instances which is acquired by using a task-shared encoder, then generating task-specific adapter layers by retrieving prototype embedding and feeding it into a hypernetworks. As shown in Figure 1, the encoded instances are projected into retrieval vectors by the instance-dense retriever for prototype learning. These prototypes that represent the task-specific information enable hypernetwork to update efficiently. This potentially allows more sample-efficient fine-tuning and few-shot transfer learning.

3.1 Instance-dense Retriever

We define an instance-dense retriever for better generalization. Let $h_i \in \mathbb{R}^d$ denote the last layer's mean-pooled hidden state of the training sample, which belongs to the i-th task. To align the embedding feature with the training sample in the latent space, an instance-dense retriever $G(\cdot)$ is applied to construct the retrieval vector $z_i = G(h_i)$, where $G(\cdot)$ is an MLP consisting of two feed-forward layers and a ReLU non-linearity. Furthermore, we need the retriever to have the ability which explicitly encourage alignment between instances from the same task, as well as push away instances from different tasks.

To efficiently learn the discriminative retriever, we introduce the following loss function $\mathcal{L}_{IR}$ based on the InfoNCE:

$$\mathcal{L}^i = \sum_{z_i \in D} \frac{-1}{N_i - 1} \sum_{z_j \in \hat{D}_i} \log \frac{\exp f(z_i \cdot z_j)}{\sum_{z_m \in S(i)} \exp f(z_i \cdot z_m)}, \tag{3}$$

$$\mathcal{L}_{\text{IR}} = \frac{1}{\tau} \sum_{i=1}^{\tau} \mathcal{L}^i, \tag{4}$$

where $\mathcal{L}^i$ is the learning objective for task i, $f(\cdot)$ is the cosine similarity function. $\hat{D}_i$ is a set of positive samples of z_i and $S(i)$ denotes a set of negative samples for z_i.

The instance-dense retriever aggregates instance-level information from the same task and enables flexible reuse of knowledge used for few-shot transfer learning.

3.2 Prototypical HyperNetworks

Simply using the learned task embedding to encapsulate task-specific information biases the task-shared hypernetwork to overfit the training data distribution, which means that inadequate sample efficiency and mixed knowledge are more susceptible to changes in distribution during cross-task transferring.

To overcome this issue, we propose to implicitly exploit the instance-level information to instruct the task embedding instead of end-to-end training. To be more specific, we found the contrastive formulation is an efficient strategy for learning robust and sample-efficient Hypernetworks. We first initialize a set of embedding $\{k_i\}_{i=1}^{\tau}$, where $\{k_i\} \in \mathbb{R}^d$ is a trainable vector to learn the specific information of the i-th task. The learning objective for an embedding is

$$\mathcal{L}^i = \sum_{z_i \in D} \frac{-1}{N_i - 1} \log \frac{\exp f(z_i \cdot k_i)}{\sum_{k_m \in V(i)} \exp f(z_i \cdot k_m)}, \tag{5}$$

$$\mathcal{L}_{\text{Pro}} = \frac{1}{\tau} \sum_{i=1}^{\tau} \mathcal{L}^i, \tag{6}$$

where $V(i)$ is a set of negative embedding. The objective forces each embedding to make use of the relative relationships between samples across tasks and avoid sample-inefficient knowledge transfer.

To generate specific parameters for different transformer layers and reduce the number of trainable parameters, we introduce a learnable layer embedding denoted as e_m, following a similar recipe as in Hyperformer (Karimi Mahabadi et al., 2021b). m denotes the m-th layer of transformer model.

Let $H(\cdot)$ denote the HyperNetwork which generates the weight matrices D_i^m and U_i^m for task conditional adapter A_i^m:

$$(D_i^m, U_i^m) = H(C(k_i, e_m)), \tag{7}$$

where $C(\cdot)$ is a project network to concatenate the task embedding and layer embedding into a mixed embedding I_i^m.

Inspired by Pfeiffer et al. (2021) and He et al. (2022a), we only insert these conditional parameters (Adapter) into the Feed-Forward Networks (FFN) sub-layer in parallel:

$$\mathbf{y} = \mathrm{FFN}(LN(\mathbf{x})) + \mathrm{A}(LN(\mathbf{x})), \quad (8)$$

where $LN(\cdot)$ represents the LayerNorm layer. This enables efficient decoupling of knowledge from different tasks to task prototypes and adapts the changeable data distribution during transfer learning.

3.3 Multi-task Tuning and New Task Generalization

PHA achieves sample-efficient multi-task learning and few-shot adaption with different training methods.

Multi-task Tuning. We follow a general multi-task learning setup, where the task identity is included, and the different datasets are concatenated together. To achieve efficient fine-tuning, the encoder with task-shared adapters is used for encoding the training sample, and we estimate the embedding corresponding to the retrieval vector given the task identity via the loss function in Equation 5. The decoder is attached by the specific adapters conditioned on contextual information. The whole model is trained in a sequence-to-sequence setting with the following objective function:

$$\mathcal{L}_{\mathrm{Total}} = \mathcal{L}_{\mathrm{PLM}} + \lambda(\mathcal{L}_{\mathrm{IR}} + \mathcal{L}_{\mathrm{Pro}}), \quad (9)$$

where $\mathcal{L}_{\mathrm{PLM}} = \sum_{i=1}^{\tau} \mathcal{L}_i$ denotes the cross-entropy loss for all training tasks and λ is a scalar balancing factor.

PHA allows the specific embedding to efficiently capture contextual information which helps the hypernetwork to generate the parameters of adapter layers in sample-efficient adaption.

New Task Generalization. For few-shot adaption, we retrieve the adaption embedding k_a by calculating the similarity scores of retrieval vector z and learned embeddings $\{k\}$ after multi-task training:

$$k_a = \arg\max_i f(k_i \mid z). \quad (10)$$

During training, we feed the adaption embedding to hypernetworks that generate the weight matrices of adapters for new tasks and optimize with the cross-entropy loss.

Our method enables efficient generalization to new tasks with limited training examples, owing to the retrieved adaption embedding containing knowledge similar to that required for the new task.

4 Experiments

4.1 Datasets

Following prior works on multi-task learning for natural language understanding (NLU) tasks, we consider 8 datasets from GLUE (Wang et al., 2019b) benchmark and 4 datasets from SuperGLUE (Wang et al., 2019a) benchmark to evaluate the performance of our models. This benchmarks is a collection of text classification tasks including CoLA (Warstadt et al., 2019) for sentence acceptability, SST-2 (Socher et al., 2013) for sentiment analysis, MNLI (Williams et al., 2018), QNLI (Demszky et al., 2018), RTE (Giampiccolo et al., 2007), CB (De Marneffe et al., 2019) for natural language inference, STS-B (Cer et al., 2017) for sentence similarity, MRPC (Dolan and Brockett, 2005), QQP (Wang et al., 2019c) for paraphrasing similarity, WSC (Levesque et al., 2012) for coreference resolution, BoolQ (Clark et al., 2019) for question answering and WiC (Pilehvar and Camacho-Collados, 2019) for word sense disambiguation. In addition, we also introduced an additional dataset: SciTail (Khot et al., 2018) for few-shot adaption.

4.2 Baselines

To evaluate the effectiveness of our proposed method, we conduct an analysis against several established methods that serve as strong baselines for multi-task learning: **Adapter** (Houlsby et al., 2019) and **Shared-Adapter**, we train adapters on a single task or a group of tasks and place them into transformer layers. **Hyperformer** (Karimi Mahabadi et al., 2021b) and **Hyperdecoder** (Ivison and Peters, 2022) that use task-conditioned or sample-conditioned Hypernetworks to generate adapters and place them into transformer layers. In addition, We compare our method with **Fully fine-tuning(FT)**, and **Shared-FT** share the model across different tasks. We also compare the state-of-art prompt transfer methods: **Prompt tuning(PT)** (Lester et al., 2021), prompt tuning prepends tunable embeddings to the input layer, and the embeddings are initialized with each task respectively. **SPoT** (Vu et al., 2022) and **ATTEMPT** (Asai et al.,

		GLUE									SuperGLUE				
Method	**Tunable Params**	**CoLA**	**SST-2**	**STS-B**	**MRPC**	**QQP**	**MNLI**	**QNLI**	**RTE**	**Avg**	**BoolQ**	**WiC**	**CB**	**WSC**	**Avg**
FT†	220M	**61.8**	**94.6**	89.7	90.2	**91.6**	**86.8**	93.0	71.9	84.9	**81.1**	**70.2**	85.7	59.6	74.2
Shared-FT†	28M	54.9	92.5	88.8	90.2	91.1	85.7	92.0	75.4	83.8	78.5	69.5	85.2	66.7	75.0
Adapter†	1.9M	64.0	93.2	**90.7**	85.3	90.2	86.5	93.2	71.9	84.5	82.5	67.1	85.7	**67.3**	75.7
Shared-Adapter†	1.8M	61.5	93.0	89.9	90.2	90.5	86.3	93.2	70.3	84.4	78.4	67.3	85.2	64.7	73.9
PT*	76.8k	10.6	90.9	89.5	68.1	89.7	81.3	92.8	54.7	72.2	61.7	48.9	67.9	51.9	57.6
Hyperformer++†	638K	59.0	93.7	90.3	88.6	89.9	85.0	93.3	**77.5**	84.7	75.8	68.9	81.5	52.9	69.8
HyperDecoder‡	1.8M	55.9	94.0	90.5	87.7	90.5	86.0	93.4	71.7	83.7	77.8	66	92.6	66.7	75.8
PHA (ours)	616K	60.6	94.0	88.9	89.2	90.9	86.3	**93.4**	**80.4**	**85.5**	80.7	64.8	**96.3**	62.7	**76.1**

Table 1: Overall comparison on Multi-task Adaption. T5-base is used as the PLM backbone of all methods. We also report **Tunable Params**, which represents the number of parameters that need to be fine-tuned for each task. The best result on each block is in **bold**. For GLUE results, † denotes results reported from (Karimi Mahabadi et al., 2021b). ‡ denotes results reported from (Ivison and Peters, 2022). $*$ denotes results reported from (Asai et al., 2022).

2022), **MPT** (Wang et al., 2023) adapts to the target tasks with the shared prompts obtained by distilling knowledge from the source tasks.

4.3 Experiments Details

Following the setting of Karimi Mahabadi et al. (2021b), when an original testing set is unavailable, the validation set is utilized as the testing set. In situations where the dataset contains less than 100k records, the validation set is divided into two sets: validation and testing. Conversely, larger datasets utilize 1000 training set samples selected for validation, with the original validation set used for testing. For the multi-task adaptation experiment, we performed multi-task learning on 8 datasets from GLUE and 4 datasets from SuperGLUE. For the low-data adaption experiment, we separately sample each individual task in GLUE with different proportions and quantities $(100, 500, 1000, 2000, 4000, 1\%, 3\%, 5\%)$. As for the evaluation strategy, we use Pearson Correlation for STS-B and accuracy for other tasks as metrics. We save a checkpoint every 1000 steps for all models and report the average performance of all tasks on a single checkpoint. In the few-shot adaption experiment, we randomly sample $k = 4, 16, 32$ instances from the training set while the entire test set is used for testing. We mainly use T5-Base (220M) model (Raffel et al., 2020) as the pre-trained language model. In addition, we also use T5-Small (60M) and T5-Large (770M) to explore the effect of model size on PHA performance in Section 4.4.5. Unless specified, we train for 65k steps using the AdamW (Loshchilov and Hutter, 2019) optimizer and set the batch size as 128 for all experiments. During training, the initial learning rate is set to 3e-4 with linear decay and 500 warm-up steps. We set the balancing factor $\lambda = 0.1$ in Eq. 9 and keep it fixed for all our experiments. All experiments run for 5 times with different seeds and we report the average for each result. The detailed configurations per method on diverse datasets are shown in Appendix A.

4.4 Results and Analysis

4.4.1 Multi-task Adaptation

Table 1 shows the evaluation results on GLUE and SuperGLUE. The results indicate that PHA outperforms all comparative methods regarding performance improvement while maintaining parameter efficiency. Note that we do not compare with SPoT, ATTEMPT, and MPT since they require pre-training prompts to save the knowledge from source tasks and transfer them to target tasks. Extending these methods to the same setting where only pre-trained models are available is beyond our scope. Therefore, under the experimental setup of multi-task learning, our method cannot achieve a fair comparison with them. It is worth mentioning that MPT, ATTEMPT, and our method both use the same two-step training method in the few-shot transfer setting (Section 4.4.3). Specifically, our PHA approach achieves a performance increase of $+1.0\%$ over Adapter while only using $3\times$ fewer trainable parameters. When we compared with the state-of-art adapter-based multi-task methods including recent Hyperformer++ and Hyperdecoder that use a hypernetwork to generate conditioned parameters similar to our approach, PHA achieves 0.8% and 1.8% point accuracy improvements respectively on GLUE while utilizing the same or lower number of trainable parameters. This demon-

k-shot		FT†	Adapter†	PT†	SPoT†	HF†	ATP†	HD	MPT‡	PHA(our)
4	CB	57.7	51.1	53.5	71.4	60.7	**82.1**	69.1	73.6	76.5
	SciTail	79.6	79.5	57.7	69.6	82.0	80.2	75.4	80.2	**82.5**
	BoolQ	50.5	53.4	61.6	50.5	48.0	61.8	54.4	62.2	**68.2**
16	CB	77.0	74.8	63.5	64.3	76.3	78.5	75.3	78.6	**79.6**
	SciTail	80.0	83.2	60.8	71.9	86.5	79.5	85.4	87.3	**87.7**
	BoolQ	56.5	51.4	61.9	50.6	50.2	60.0	64.6	63.3	**71.3**
32	CB	80.0	74.8	67.8	64.3	81.4	**85.7**	79.6	82.1	82.7
	SciTail	81.9	85.0	60.2	71.9	85.8	80.2	85.1	86.3	**88.6**
	BoolQ	58.4	54.5	61.7	61.2	58.3	65.3	68.3	68.9	**72.0**

Table 2: Result for few-shot transfer ($k = 4, 16, 32$). We report the accuracy for all tasks over 10 random seeds. The best result on each block is in **bold**. All of the models are trained on GLUE tasks with T5-Base backbone. † denotes results reported from (Asai et al., 2022). ‡ denotes results reported from (Wang et al., 2023).

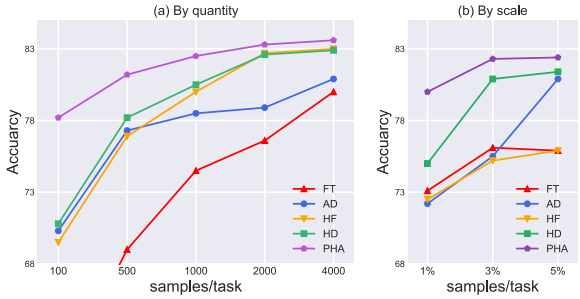

Figure 2: Results on GLUE for sample efficiency experiments, which sample the various number or proportions of training samples per task. AD, HF and HD denote Adapter, HyperFormer and HyperDecoder.

strates the potential of our method to reduce the negative interference between tasks better. In addition, we observe that FT performs the best in all experimental methods, while it requires 220M trainable parameters on a single task. We find that our PHA is as competitive as that of the FT(85.5 vs. 84.9) and reduce the trainable parameters from 100% to 0.28%. We also analyze the effectiveness of PHA in parameter efficiency, as detailed in Appendix B.

4.4.2 Low-data Adaptation

In our framework, we posit that the task prototypes, which are estimated by instance-level features, can better represent task information and improve the performance of hypernetworks in a sample-efficient manner during multi-task adaptation. To verify that our proposed method generalizes better when there are only limited available resources, we conduct low-data adaption experiments on the GLUE benchmark. Following Karimi Mahabadi et al. (2021b), we train all models(Fine-tuning, Adapter, Hyperformer++, and Hyperdecoder) for 15k steps. More details are in Section 4.3.

As shown in Figure 2, PHA outperforms other baselines consistently across different configurations. We obverse that Fine-tuning performs the worst in cases of limited available data. A potential reason is that tuning all parameters of the base model makes it susceptible to keep the model in the state of over-fitting, especially in the low-data regimes. As for other experimental methods, we observe that when the number of available samples is relatively smaller, the existing state-of-the-art multi-task methods with hypernetworks close to the performance of Adapter until the number of available samples increases. This observation indicates that hypernetworks struggle to generate optimal parameters when using randomly initialized task embeddings or instances as contextual information under low-data regimes. Moreover, to better simulate the training task distribution in the GLUE benchmark, we randomly sample each individual task in GLUE for different proportions. Figure 2(b) shows the comparison between PHA and adapter-based methods. Similar to the results in Figure 2(a), our proposed method outperforms others by a large margin. We also conduct experiments on 4 datasets (BoolQ, WiC, CB, WSC) from SuperGLUE, as detailed in Appendix C.

The superior performance of PHA over all competing methods indicates that our proposed task prototype efficiently captures contextual information to enable hypernetwork to generate the parameters of adapter layers.

4.4.3 Few-shot Adaptation

We explore how our proposed method performs when adapting to new tasks with sample-efficient. Specifically, following Wang et al. (2023), we conduct few-shot experiments on BoolQ, CB, SciTail and compare PHA with the strong baselines, including Fine-tuning, Adapter, Prompt tuning, SPoT, Hy-

Prototype	Retriever	GLUE Score
×	×	84.0
×	√	84.3
√	×	84.7
√	√	**85.5**

Table 3: Ablation study on GLUE.

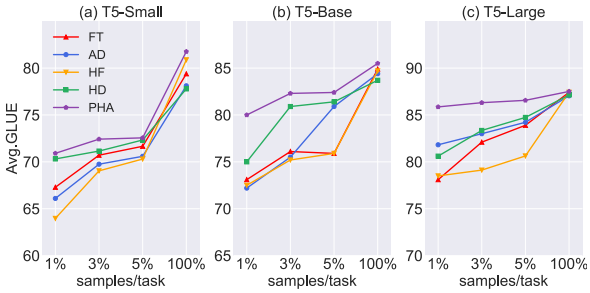

Figure 3: The comparison of PEFTs (Adapter, Hyperformer++, Hyperdecoder, and our proposed PHA) and FT on GLUE with three scale pre-trained models under low-data regimes and full-data regimes.

performer, ATTEMPT, HyperDecoder, and MPT. The results in Table 2 are obtained by training an 8-task adaptation for GLUE and fine-tuned with few-shot samples from BoolQ, CB, and SciTail. More details are in Section 4.3.

Table 2 summarizes the results on the few-shot adaption setting. Among Adapter-based transfer methods, PHA brings around $3\% \sim 20\%$ absolute improvements over the Adapter across different settings. While Hyperformer achieves better generation for new tasks, it requires us to have a precise understanding of target tasks. PHA significantly improves the performance of Hyperformer without requiring task-specific prior knowledge. In addition, our proposed method significantly outperforms other prompt-based transfer methods in most settings.

Our results demonstrate that our approach effectively generalizes to new tasks despite the limited availability of training samples.

4.4.4 Ablation Study

We perform an ablation study on the GLUE benchmark to evaluate the effectiveness of the proposed modules. The prototype design and the instance-dense retriever are removed independently for this purpose. As shown in Table 3 (row 2), when we remove the prototype design and use the retrieval instances to train, we observe the performance has a significant drop. This shows that the instance-level

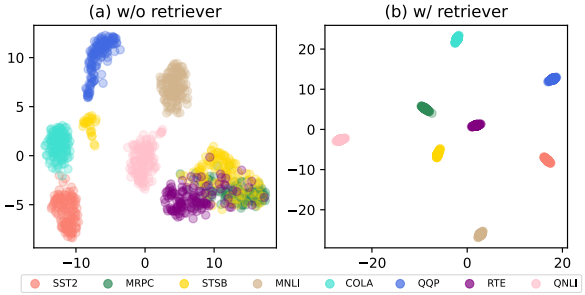

Figure 4: t-SNE visualizations for samples w/ (right) and w/o (left) retriever.

information hinders the positive transfer across tasks under the limitation of hypernetwork capacity, while our design of task prototypes allows hypernetwork to capture shared-task information well, which is vital for enabling positive transfer across tasks. Table 3 (row 3) removes the retriever. The task prototypes are estimated by the originally encoded instances. This results in intertwined task prototypes due to the relative dispersion of instance information in the embedding space. The decrease in performance suggests that adding the instance-dense retriever enables the prototype to encode task-specific knowledge better. Furthermore, we provide a visualization of the encoded instances from the GLUE benchmark to compare the effect of adding and removing the instance-dense retriever, as shown in Figure 4. While samples belonging to the same task tend to be located near each other in the latent space, samples from different classes (e.g., STS-B, MRPC, RTE) still interleave with each other. After the retriever is added, instances from the same task are tightly clustered, while different tasks are widely separated.

4.4.5 Impact of Model Scale

To verify that our method is applicable to different pre-trained model sizes, we also experiment T5 with sizes from Small (60M) to Large (770M) on GLUE datasets, while reporting the average result of PHA as well as fully Fine-tuning (FT), Adapter (AD), Hyperformer++ (HF) and Hyperdecoder(HD). As shown in Figure 3, under three scales of pre-trained model, we find that PHA achieves superior and competitive performances in low-data and full-data regimes, respectively. This indicates that our proposed prototypical strategy is still able to achieve the best sample efficiency when the size of large-scale transformer models increases.

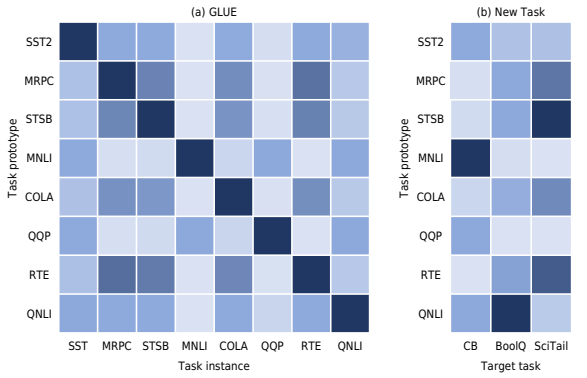

Figure 5: Visualization of the similarity scores, which are calculated by the dense retriever, between the instances and task prototype in GLUE (a) and between pre-trained prototypes and the task-agnostic instances in three datasets (b). Darker colors indicate higher scores.

4.4.6 Effect of Retriever for Generalization.

The retriever plays an important role in adapting to new tasks. To explore how the retriever works after training under a multi-task setting, we consider the similarity scores, described in Eq. 10, to measure retrieval results. Specifically, we randomly sample each individual task in GLUE and calculate the similarity scores through the trained task prototypes and retrieval vectors transferred by the trained retriever. Figure 5(a) shows a visualization of similarity scores. We find that the retriever precisely retrieved the task identity of the corresponding task instance. This suggests that task prototypes and instance vector has aligned in embedding space to enable more efficient capture of single-task common features.

We also demonstrate that the retriever has the ability to match the corresponding task prototype to target tasks that require generalization. Figure 5(b) illustrates that the similarity score is relatively high for related tasks such as CB and MNLI, SciTail, and STSB, all of which belong to the NLI task family. As for QNLI and BoolQ, since the task prototype trained on GLUE does not include Boolean Question Answering (QA) tasks, the retriever has matched QNLI prototype, which is in the same domain as BoolQ. Therefore, our proposed method can naturally generalize to new tasks when the related task prototype and the hypernetwork containing cross-task knowledge are both available.

5 Related Work

Multi-task Learning and Transfer. Multi-task learning(MTL) aims to take advantage of the shared information between the different tasks and train a unified model to simultaneously solve multiple tasks. In the context of NLP, this is typically achieved by sharing certain layers across all tasks while using task-specific layers for specific tasks (Liu et al., 2019). With the popularization of large language models (LLMs), Raffel et al. (2020) explores the training of LLMs on various tasks which are transformed into a unified format, and some works (Aghajanyan et al., 2021; Aribandi et al., 2022; Sanh et al., 2022; Wei et al., 2022) indicate that the LLMs can be better generalized to new tasks through large-scale multitasking training. More recent work (Pfeiffer et al., 2021; Vu et al., 2022; Asai et al., 2022; Wang et al., 2023) focuses on multi-task transfer with parameter-efficient fine-tuning as the increasing LM size. Though the effectiveness of multitask learning is improved, they need to finetuning the LLMs twice on source tasks, which are carefully selected, and multiple target tasks. This limits the applicability of the methods. Differently, our proposed method only requires a pre-trained model to achieve multi-task learning and transfer to a new task.

Several works (Jin et al., 2020; Karimi Mahabadi et al., 2021b; Ivison and Peters, 2022; He et al., 2022b) introduce hypernetworks (Ha et al., 2017) to share the cross-task information by generating the parameters of adapter layers (Houlsby et al., 2019) from specific embeddings during multi-task learning. Our work is motivated by Ivison and Peters (2022), but proposes to use task-level information represented by prototypes to optimize the embedding distribution of hypernetworks, which reduces negative transfer between different tasks and improves the performance of adaption across tasks, especially in low-resource regimes.

Prototype Learning. Prototype learning is widely used to improve the representation ability of networks for few-shot learning. Some works (Gao et al., 2019; Caron et al., 2020; Ding et al., 2021; Li et al., 2021) indicate that the prototype is forced to learn some common features of samples within the class by prototype estimation. Cui et al. (2022) propose to construct a verbalizer for prompt-based few-shot tuning by estimating prototypes with contrastive learning. This differs from our method, which uses a prototypical strategy to explore the specific information for corresponding tasks.

6 Conclusion

We introduce Prototype-based HyperAdapter (PHA), a novel framework built on adapter-tuning. PHA achieves both multi-task adaption and adaption to a new task in a sample-efficient manner. It generates parameters of adapter layers conditioned on task-specific prototypes which are calculated by the corresponding instance-level features. In addition, the specific prototype is retrieved and transferred to new tasks to be further tuned. The resulting method significantly outperforms previous SOTA on full/low-data multi-task adaption and few-shot adaption.

Limitations

Our work has demonstrated strong experimental results and sample efficiency in multitasking adaption. However, there are several limitations: Firstly, in few-shot adaption, the method we proposed, which tunes the base model on 8 NLP tasks, can generalize to new target tasks efficiently. But tuning on more large-scale tasks may result in better generalization improvements. Secondly, as shown in Figure 5, a new task may be related to multiple task prototypes, rather than a single one. In our method, we only select the most relevant prototypes, which may ignore the transfer of some weakly related knowledge. In addition, we use adapters in this work, but our method could possibly also benefit from other parameter-efficient approaches (Lester et al., 2021; Mahabadi et al., 2021; Li and Liang, 2021; Hu et al., 2022; Liu et al., 2022).

Acknowledgements

This work is supported by National Key R&D Program of China (Grant No. 2022YFF1202400), Major Science and Technology Innovation Program of Hangzhou (Grant No. 2022AIZD0154), Natural Science Foundation of China (Grant No. 62176025, 62301066), Beijing Nova Program (Grant No. 20220484161), the Fundamental Research Funds for the Central Universities (Grant No. 2023RC72) and Theme-based Research Scheme (T45-205/21-N), Research Grants Council of Hong Kong.

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.

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*.

Akari Asai, Mohammadreza Salehi, Matthew Peters, and Hannaneh Hajishirzi. 2022. ATTEMPT: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc.

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.

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*.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7014–7024, Dublin, Ireland. Association for Computational Linguistics.

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*.

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

Ning Ding, Xiaobin Wang, Yao Fu, Guangwei Xu, Rui Wang, Pengjun Xie, Ying Shen, Fei Huang, Hai-Tao Zheng, and Rui Zhang. 2021. Prototypical representation learning for relation extraction. In *International Conference on Learning Representations*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6407–6414.

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.

Anchun Gui and Han Xiao. 2023. HiFi: High-information attention heads hold for parameter-efficient model adaptation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8521–8537, Toronto, Canada. Association for Computational Linguistics.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *International Conference on Learning Representations*.

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

Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, Yaguang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022b. HyperPrompt: Prompt-based task-conditioning of transformers. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8678–8690. PMLR.

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.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Hamish Ivison and Matthew Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1715–1730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tian Jin, Zhun Liu, Shengjia Yan, Alexandre Eichenberger, and Louis-Philippe Morency. 2020. Language to network: Conditional parameter adaptation with natural language descriptions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6994–7007, Online. Association for Computational Linguistics.

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

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. 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.

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

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

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

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2021. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*.

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.

Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4260, Toronto, Canada. Association for Computational Linguistics.

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

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.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

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.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. 2023. Modular deep learning. *arXiv preprint arXiv:2302.11529*.

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.

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*.

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 *International Conference on Learning Representations*.

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.

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.

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

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*.

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

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning. In *The Eleventh 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.

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 *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.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhongbin Xie and Thomas Lukasiewicz. 2023. An empirical analysis of parameter-efficient methods for debiasing pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15730–15745, Toronto, Canada. Association for Computational Linguistics.

Guangtao Zeng, Peiyuan Zhang, and Wei Lu. 2023. One network, many masks: Towards more parameter-efficient transfer learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7580, Toronto, Canada. Association for Computational Linguistics.

A Implementation Details

We set the dimension of the retrieval vector (z_i), the task prototype (k_i), and the layer embedding z_m to 128. For the instance-dense retriever $G(\cdot)$ with bottleneck architecture, we select the bottleneck size to 128. For the mixed embedding I_i^m, the dimension of I_i^m is selected as 32 for multi-task experiments and 64 for other experiments. We use the default hyperparameters by Ivison and Peters (2022) for Adapter and Hyperdecoder. The hyperparameters of other baselines are set according to the original papers (Karimi Mahabadi et al., 2021b; Asai et al., 2022; Wang et al., 2023). Following Karimi Mahabadi et al. (2021b); Ivison and Peters (2022), the length of the sequence is 128 at the encoder and the decoder. We use its HuggingFace (Wolf et al., 2020) Pytorch (Paszke et al., 2019) implementation. All data is obtained from huggingface datasets and preprocessed into a "seq2seq" format following Raffel et al. (2020).

B Parameter Efficiency

Our proposed method not only balances sample efficiency and accuracy on stream tasks but also achieves parameter efficiency. We compare the trainable parameters of PHA with other baselines. We assume that the basic model is a transformer model with an L-layer encoder-decoder structure for τ-tasks and d is the model dimension. The tasks-share adapter is attached to the encoder layer and the condition-generate adapter is attached to the decoder layer in our work. The bottleneck dimension of Adapter is b. Adapter for the single layer costs $2db + b + d$. Given bottleneck size d', the instance-dense retriever $G(\cdot)$ with bottleneck architecture to construct the retrieval vector $z \in \mathbb{R}^{d'}$, which result in $dd' + d'^2$ parameters. We have the task prototype $k_i \in \mathbb{R}^{d'}$ for i-th task and layer embedding $e_m \in \mathbb{R}^{d'}$ for m-th layer. These have a total of $\tau d' + Ld'$ parameters. The project network $C(\cdot)$ is used for concatenating the task prototype $k_i \in \mathbb{R}^{d'}$ and layer embedding $e_m \in \mathbb{R}^{d'}$ into a mixed embedding $I_i^m \in \mathbb{R}^{d_h}$, which result in $2d'd_h$ parameters. The hypernetworks costs $d_h(2db + b + d)$ parameters. The total cost of PHA is $dd' + d'^2 + (\tau d' + Ld') + 2d'd_h + d_h(2db + b + d) + L(2db + b + d)$. There are a large number of tasks and transformer layers in our work settings. For the adapter, its parameter count increases linearly with the task and number of layers. For Hyperformer, its parameter count is more than our method due to using two hypernetworks. We used a bottleneck structured retriever to reduce trainable parameters compared to Hyperdecoder.

samples/ task	AD	HF	HD	PHA
1%	57.8	50.4	57.7	**59.2**
3%	61.3	51.3	59.6	**67.6**
5%	60.3	54.3	59.9	**69.3**

Table 4: Results on 4 datasets (BoolQ, WiC, CB, WSC) from SuperGLUE for the various proportions of training samples per task (1%,3%,5%). We report the average accuracy. AD, HF and HD denote Adapter, HyperFormer and HyperDecoder.

Method		XSUM	Ro-En	En-Ro	Avg
Small	FT	7.2	24.1	20.1	17.1
	AD	2.9	25.1	9.7	12.6
	PHA	7.6	24.9	19.7	**17.4**
Base	FT	10.9	25.3	24.2	20.1
	AD	8.2	26.5	19.5	18.1
	PHA	10.8	26.6	24.8	**20.7**
Large	FT	12.3	24.7	29.7	22.2
	AD	12.6	27.1	25.9	21.9
	PHA	14.2	27.0	28.6	**23.3**

Table 5: Result for NLU task. We report the ROUGE2 score for XSUM and BLEU score for Ro-En/En-Ro. The best result on each block is in **bold**.

C Additional Results

Here, we conduct more experiments to demonstrate the sample efficiency of our method on different datasets (BoolQ, WiC, CB, and WSC) with T5-Base pre-trained model. The result are included in Table 4. The performance PHA with T5-Base achieves the best performance in terms of accuracy and sample efficiency in the 4 datasets from SuperGLUE. In addition, to verify the effectiveness of our method on NLU tasks, we conducted sample efficiency experiments on three datasets: Xsum, WMT16 Ro-En, and WMT16 En-Ro. Due to limited computation resources, we sampled 1% of the data and conducted experiments on three backbone models of different scales(small, base, and large). The results are shown in Table 5.