HyperLoRA: Efficient Cross-task Generalization via Constrained Low-Rank Adapters Generation

Anonymous ACL submission

Abstract

Adapting pre-trained language models (PLMs) for cross-task generalization is a crucial research area within the field of NLP. While fine-tuning and in-context learning are effective approaches for adapting LMs to emerging tasks, they can be costly and inefficient. Recently, some researchers have focused on achieving efficient task adaptation via hypernetwork, which is a meta network that generates task-specific weights based on task-oriented information without any optimization. However, the training of hypernetworks often lacks stability since the optimization signal is not straightforward, and the task information is not adequately representative. Moreover, previous works train hypenetworks with the general corpus, which is struggling with few-shot adaptation. To address these issues, we introduce `HyperLoRA`, a hypernetwork for LoRA parameters generation involving hypernetwork pretraining on instruction-following data and generalization fine-tuning on sparse task data. Furthermore, we utilize a constrained training loss and a gradient-based demonstration selection strategy to enhance the training stability and performance. Experimental results and analysis across four benchmark datasets (P3, S-NI, BBH, and SuperGLUE) demonstrate the proposed approach has flexible generalization ability and superior performance.

1 Introduction

Pre-trained language models (PLMs) have shown remarkable capabilities across a diverse spectrum of NLP tasks, encompassing understanding (Devlin et al., 2019; Liu et al., 2019), reasoning (Liu et al., 2023; Wang et al., 2023b), and generation (Raffel et al., 2020a; Brown et al., 2020). The ability of language models to effectively adapt their knowledge to unseen tasks (referred to cross-task generalization) is crucial for the broader applicability of NLP systems, which garnered significant attention from many researchers.

There are several approaches towards achieving cross-task generalization. The most straightforward way is fine-tuning LMs with a certain amount of task-specific data, which demands substantial computational costs and may cause catastrophic forgetting, degenerating the performance of LMs on the previous tasks (Chen et al., 2020). In contrast, in-context learning (ICL) provides a few demonstration examples to generalize LMs to unseen tasks without explicit optimization (Brown et al., 2020; Min et al., 2022). However, ICL requires extremely long and expensive-to-process inputs for each test example, making it both costly and inefficient (Zhou et al., 2023; Li et al., 2023). Another line of researchers explores the cheaper and more effective approach that composes the weights of new tasks by selecting and combining fine-tuned or parameter-efficient weights from a pre-existing weights pool (Vu et al., 2022; Ponti et al., 2023; Poth et al., 2023; Huang et al., 2023). While this approach is simple and effective, it necessitates a pre-existing pool of various task weights, and the effectiveness of the composed weights may be restricted by the pool of available tasks. Additionally, although the parameter-efficient weights are lightweight, there are still resource consumption issues for storage and training while the pre-existing pool is substantial in various task scenarios.

To address the above issues in the cross-task generalization scenario, (Ha et al., 2017; Phang et al., 2023; Ivison et al., 2023) proposes a meta network named hypernetwork, which performs a "text-to-weight" task converting task information (*e.g.* task instructions and task demonstrations) into task-specific parameters (*e.g.* prefixes (Li and Liang, 2021), adapters (Houlsby et al., 2019), LoRA (Hu et al., 2022)) for underlying pre-trained language models. Compared with fine-tuning and ICL, hypernetwork is a more efficient method that generates task-specific parameters in a single forward without any optimization. Furthermore, hy-

pernetwork can generate parameters when required, avoiding additional storage expenditures.

Despite the obvious strength of previous works on hypernetwork, several issues remain that have not been appropriately solved. **Firstly**, the training instability problem is one of the most challenging problems in hypernetwork training (Chang et al., 2020). Existing strategies primarily rely on back-propagating gradients from the underlying model, while lacking effective and specific measures to circumvent this instability. **Secondly**, most current works construct task demonstrations through manual crafting or random sampling from datasets, potentially affecting parameter generation and resulting in suboptimal performance. **Thirdly**, previous works typically pre-train hypernetworks on general language corpus (*e.g.* C4 (Raffel et al., 2020b)) directly, which may not be adept at handling diverse task instructions and could perform suboptimally on low-resource task data.

Considering all the above considerations, we introduce HyperLoRA, a novel method that aims to enable language models for efficient cross-task generalization. Specifically, HyperLoRA consists of a text encoder and a P-generator, designed to convert task information into parameter-efficient modules, specifically LoRA (Hu et al., 2022). To improve training stability, we propose an explicit training loss to constraint the training of HyperLoRA and utilize a gradient-based automatic demonstration selection strategy to select the most representative task examples. Furthermore, our HyperLoRA involves hypernetwork pre-training on instruction-following data to enable it to generate task-related parameters based on task information and then generalization fine-tuning on sparse task data to adapt LMs with unseen tasks. In a nutshell, the contributions of our work are as follows:

- We introduce an efficient cross-task generalization method HyperLoRA, which contains a text encoder and a P(arameters)-generator to convert task information into LoRA modules.
- To enhance the generalization ability of hypernetwork, we propose a paradigm that incorporates hypernetwork pre-training on multi-task instruction-following data and generalization fine-tuning with sparse task data.
- We develop a constrained training loss and an automatic demonstration selection strategy to improve training stability and performance.
- The experimental and analysis results across cross-task generalization and few-shot adaptation scenarios demonstrate the effectiveness of our proposed method.

2 Related Work

2.1 Efficient Cross-Task Generalization

Efficient adaptation of pre-trained LLMs to unseen tasks is an important and challenging research direction. One primary area of research focuses on prompt tuning. In this line, the T5 model (Raffel et al., 2020a) unified all NLP tasks as a Text-to-Text problem, providing a solid foundation for follow-up works. Afterwards, instruction tuning that fine-tuning LMs with various multi-task instructions is proposed (Wei et al., 2022; Sanh et al., 2022; Ouyang et al., 2022), which improves zero-shot and few-shot generalizations greatly since the fine-tuned model learned to utilize instructions to perform novel tasks. In-context learning further employs task examples as demonstrations in addition to instructions, adapting models without optimization. Nevertheless, this increases computation costs due to longer inputs from demonstrations and instructions, and the performance depends largely on the inherent ability of LLMs. Another efficient stream of research focuses on task-specific weight composition with parameter-efficient fine-tuning (PEFT). Among this, Vu et al. (2022); Su et al. (2022) explore transferring PEFT modules from source tasks and find it benefits novel downstream tasks. Chronopoulou et al. (2023); Poth et al. (2023); Pfeiffer et al. (2020); Chen et al. (2023); Huang et al. (2023) compose the weights for new tasks by selecting relevant tasks and combining their task-specific weights. While this approach is efficient, it necessitates a pre-existing pool of task-specific weights based on task information, and the task pool may limit the expressiveness of the composed weights. Meanwhile, Mahabadi et al. (2021b); He et al. (2022); Wang et al. (2023c); Phang et al. (2023) introduce employing a meta network named hypernetwork to generate task-specific weights and achieve superior cross-task performance. Our work also aligns with this direction and aims to enhance the generalization capability and training stability of hypernetworks.

2.2 Hypernetwork

Hypernetworks are meta neural networks that generate parameters for another primary network,

which gains popularity in multi-task learning scenarios. Mahabadi et al. (2021a) leverages hypernetwork with shared weights across adapters for LMs adapting. Mahabadi et al. (2021b) and He et al. (2022) further propose task-conditioned hypernetworks and enable information sharing across tasks. Moreover, Ivison et al. (2023); Phang et al. (2023); Liang et al. (2023) utilize LMs to initialize hypernetworks and propose hypernetworks pre-training on large-scale general corpus data. However, the above hypernet-based methods have limitations and struggle in few-shot adaptation scenarios. Different from those approaches, our work stands out from those approaches in several ways. We train hypernetwork with instruction-following data to improve its robustness for diverse task instructions. Furthermore, we introduce a constrained training objective to enhance training stability and develop an automatic demonstration selection strategy to further improve its performance.

3 Methodology

3.1 Revisiting the Low-Rank Adapter (LoRA) Finetuning Method

Hu et al. (2022) demonstrates that weight updates in the pre-trained models (PTMs) exhibit a low "intrinsic dimension" while adapting PTMs to specific tasks, and further proposes the Low-Rank Adapter (LoRA) finetuning method. Through updating a small set of trainable adapters and fixing full model parameters, the LoRA method substantially reduces memory requirements and achieves comparable results with full-parameter finetuning. Specifically, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA constrains its update by representing it with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times d}$, and the rank $r \ll \min(d, k)$. W_0 is frozen during training, while A and B contain trainable parameters. Considering the input as x and the operation $h = W_0 x$, the forward pass will be modified with LoRA:

$$h = W_0 x + \Delta W x = W_0 x + ABx \qquad (1)$$

In general, ΔW is scaled by $\frac{\alpha}{r}$, where α is a constant in r.

3.2 HyperLoRA

As illustrated in Figure 1, our HyperLoRA is a hypernetwork to convert task instructions to LoRA modules, which consists of three essential elements: a **text encoder** to transform task information into continuous representations, **P-generator** facilitates interaction between the encoded instructions and a collection of trainable embeddings, serving the role of synthesizing LoRA parameters.

Text Encoder To encode the task information effectively, we initialize the text encoder with an encoder of a pre-trained language model. Given the task information $\boldsymbol{x} = [\boldsymbol{x}_i; \boldsymbol{x}_e]$ as inputs, where $\boldsymbol{x}_i$ is task instruction and $\boldsymbol{x}_e$ refers to task demonstrations. We encode $\boldsymbol{x}$ as follows:

$$\mathbf{H_e} = \mathrm{Enc}(\boldsymbol{x}) \qquad (2)$$

where $\mathbf{H_e} \in \mathbb{R}^{n\times d}$ is the encoded task features, n is the length of $\boldsymbol{x}$ and d is hidden dimension.

P(arameters)-generator The P-generator assumes a pivotal role in bridging text representation space and parameter space. It extracts a fixed number of the output features from the text encoder to generate parameters. As shown in Figure 1 (a), the P-generator consists of two submodules: (1) a transformer decoder that extracts the task features. (2) a generator module to generate task parameters for the underlying model. For the inputs of the decoder, we create a set number of learnable task query embeddings, which denotes to $\mathbf{E} = (\boldsymbol{e_1}, ..., \boldsymbol{e_l}) \in \mathbb{R}^{l\times d}$, where l is the number of layers of the underlying model. The task queries interact with each other through self-attention layers and interact with the task features $\mathbf{H_e}$ through cross-attention layers:

$$\mathbf{H_d} = \mathrm{Dec}(\mathbf{H_e}; \mathbf{E}) \qquad (3)$$

where $\mathbf{H_d} \in \mathbb{R}^{l\times d}$ is the output feature of the transformer decoder, prepared to generate parameters. The generator module of the P-generator aims to conditionally generate LoRA parameters of underlying models based on the output features $\mathbf{H_e}$ through some two-layer MLP modules. We employ separate networks for distinct LoRA weights while sharing between the layers. Given the query weight W_q in the attention module as an example, we generate the low-rank parameters A and B via $\mathrm{MLP}_{q,A}$ and $\mathrm{MLP}_{q,B}$ for W_q of all layers, respectively:

$$\phi^{(i)}_{q,A} = \mathrm{MLP}_{q,A}(h^{(i)}_d) \quad \forall i \in \{1, ..., l\} \qquad (4)$$

$$\phi^{(i)}_{q,B} = \mathrm{MLP}_{q,B}(h^{(i)}_d) \quad \forall i \in \{1, ..., l\} \qquad (5)$$

where $h^{(i)}_d \in \mathbb{R}^{(1\times d)}$ is the i-th vector of $\mathbf{H}_d$, and $\phi^{(i)}_{q,A}$ is the parameter of A that is utilized to adapt the query weight W_q of the i-th attention layer.

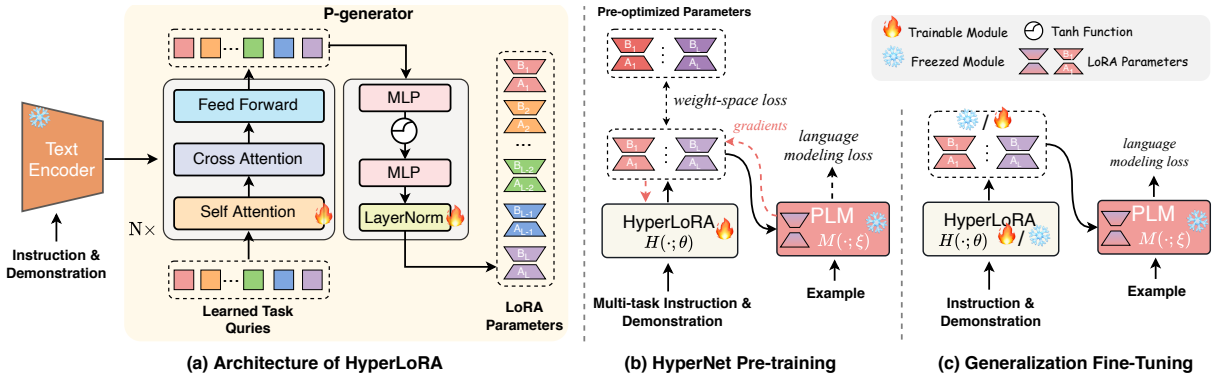


Figure 1: Overview of the proposed methods. (a) The architecture of HyperLoRA. (b) The truth-guided pre-training stage. (c) The continuous fine-tuning stage to generalize our HyperLoRA into few-shot scenario.

3.3 HyperNet Pretraining

Previous hypernet-based methods pre-train hypernetworks with general corpus, which potentially limits its generalization capacity to novel tasks. To relieve this, we design a pre-training stage with multi-task instruction data to equip HyperLoRA with the ability to convert various task information to LoRA modules. As shown in Figure 1 (b), we denote parameterized HyperLoRA by θ as $H(\cdot;\theta)$. At each training iteration, HyperLoRA receives task instruction and k-shot task demonstrations of the task τ as input x_τ and generate the LoRA parameters:

$$\phi_\tau = H(x_\tau;\theta) \tag{6}$$

The underlying model $M(\cdot;\xi)$ takes in the query q of the task τ and generates the response with the generated LoRA parameters ϕ fusion. Then HyperLoRA is optimized based on the underlying model's predictions:

$$\min_{\theta} \mathbb{E}_{\tau\in\mathcal{T},(q,a)\in\mathcal{D}_\tau} \mathcal{L}(M(q;\xi,\phi_\tau),a) \tag{7}$$

where a is the golden response of the query q, $\mathcal{T}$ is a collection of pre-training tasks. Note that only HyperLoRA is trained and the underlying model is frozen during the pre-training stage. As a consequence, the generated parameters ϕ can be computed once for a specific task information, subsequently reused for downstream predictions during inference or further tuning scenarios, which saves memory and computation. To train HyperLoRA robustly and effectively, we introduce the method of *gradient-based demonstration selection* and employ a *truth-guided training objective*.

Gradient-based Demonstration Selection Method The task instruction and task demonstrations are essential for hypernetworks to capture the task features and generate task-specific parameters. However, prior studies (Ivison et al., 2023; Mahabadi et al., 2021b; Phang et al., 2023) construct the task demonstrations through **manual crafting** or **random sampling** from datasets, potentially affecting parameter generation and resulting in suboptimal performance. We introduce an automatic demonstration selection method via gradient-based influence estimation. **Firstly**, we pre-filter demonstrations via embedding and clustering. Specifically, we convert each instance of the task $\tau \in \mathcal{T}$ into vector representations using Sentence-BERT (Reimers and Gurevych, 2019), and then we cluster the contextualized vectors utilizing the k-means clustering algorithm to produce k clusters. The instances closest to the center of the cluster are sampled as the filtered task demonstrations. **Secondly**, we follow Xia et al. (2024) and warmup training the hypernetwork using the preliminarily selected demonstrations. **Finally**, we compute the gradient-based influence score of each demonstration based on the trained hypernetwork as follows:

$$\text{Inf}(d_\tau,t_\tau) \triangleq \sum_{i=1}^{N} \bar{\eta}_i \frac{\langle \mathcal{L}(d_\tau;\theta_i), \mathcal{L}(t_\tau,\theta_i)\rangle}{\|\mathcal{L}(d_\tau;\theta_i)\|\|\mathcal{L}(t_\tau,\theta_i)\|} \tag{8}$$

where d_τ and t_τ refers to the demonstration and test example of the task τ, respectively, η_i is the learning rate during the i-th epoch and $\mathcal{L}$ is the loss function. The influence score calculated above reflects the importance of the demonstration to the test examples, thus we select the demonstrations with higher scores as the final representative demonstrations.

Truth-Guided Training Objective The most challenging in hypernetwork training is its instability, which can be attributed to multiple aspects: (1) *Weight initialization.* The choice of how the

weights are initialized significantly impacts the convergence and stability of hypernetwork training. (2) *Disparities Between Input and Output.* There are substantial distinctions between the representation space of the input text and the output parameters, which can harm the stability of hypernetwork training. (3) *Indirect Objective in an End-to-End Differentiable Manner.* During the training stage, the optimization of the hypernetwork relies on back-propagated gradients from underlying models. However, the constraint objective is tailored for underlying models rather than the hypernetwork. To resolve the above issues, we conduct experiments with various hypernet initialization configurations (including scale, type, and initialize ways), and discover that reusing the weights from the underlying model yields the most favorable results, in terms of performance and training convergence, which is consistent with (Ivison et al., 2023). Significantly, we propose a truth-guided training objective to incorporate more direct weight-space constraint loss for hypernetworks. In this approach, we pre-optimize a set of LoRA parameters $\hat{\phi}_\tau$ with the underlying model for task τ, and then utilize $\hat{\phi}_\tau$ to constrain the generation process:

$$\mathcal{L}(M(q;\xi,\phi_\tau),a) = \underbrace{\sum_{(q,a)\in\mathcal{D}^\tau} \log(p(a;q,\xi))}_{\textit{language modeling loss}} + \underbrace{\beta||\hat{\phi}_\tau - \phi_\tau||}_{\textit{weight-space loss}}$$

where β is hyperparameters that control for the relative weight of the constrained loss.

4 Experiments

4.1 Experimental Settings

Dataset We conduct experiments in cross-task generalization and few-shot adaptation settings. For the former, we do evaluations on the **Public Pool of Prompts (P3)** (Bach et al., 2022) and the instruction-based dataset **Super-Natural Instructions (S-NI)** (Wang et al., 2022). For the latter, we evaluate the multi-task benchmarks **SuperGLUE** (Wang et al., 2019a) and the diverse and challenging benchmark **BIG-Bench Hard (BBH)** (Suzgun et al., 2023). In addition, we utilize a subset of FLAN (Wei et al., 2022) following (Huang et al., 2023) in the pre-training stage to enable HyperLoRA to generate task-specific parameters. More details about the datasets can be seen in the Appendix B.

Baselines To evaluate the effectiveness of the proposed method, we compare it with several baselines, including: (1) *Full Fine-tuning methods.* We multi-task fine-tune the pre-trained language models T5 (Raffel et al., 2020a) on the provided training set and evaluate it on the held-out test set. (2) *Parameter-Efficient Fine-Tuning (PEFT) methods.* We primary focus centers on LoRA (Hu et al., 2022) and a weight composition method LoraHub (Huang et al., 2023). (3) *Hypernetwork-based methods.* This methods including HyperTuning (Phang et al., 2023), HINT (Ivison et al., 2023), and HART (Liang et al., 2023). (4) *Our methods.* HyperLoRA indicates we pre-train the hypernet on instruction data and further tune the hypernet on downstream tasks, while HyperLoRA† represents we continuously fine-tune the efficient parameters generated by the instruction pre-trained hyerpnet.

Experimental Details To conduct fair comparisons with various baselines, we utilize Flan-T5 (Chung et al., 2022) as underlying models in the BBH dataset and LM-adapted T5 (Lester et al., 2021) in other datasets. We initialize HyperLoRA with the parameters of the corresponding underlying model to achieve stable training and better performance. More details can be seen in Appendix C.

4.2 Cross-Task Generalization

We conduct cross-task generalization experiments mainly on the Public Pool of Prompts (P3) and Super-Natural Instructions (S-NI) datasets. In this scenario, we first pre-train HyperLoRA on the instruction-following data FLAN (Wei et al., 2022) with the truth-guided pre-training strategy, and then multi-task fine-tune it on the training sets of P3 and S-NI, similar to previous studies (Phang et al., 2023; Ivison et al., 2023; Liang et al., 2023).

Multi-learning on P3 benchmark. The evaluation is performed on a fixed set of P3 held-out tasks based on the multiple-choice scoring with accuracy, and the evaluation results are presented at Table 1. HyperLoRA achieves the best performance with a 1.5% Avg. score improvement than the previous SOTA hypernetwork-based method HyperTuning+. Notably, HyperTuning+ jointly trains both the hypernet and the underlying T5 model, which increases the storage and compute cost and may cause catastrophic forgetting. In contrast, our HyperLoRA adopts a more efficient way that freezes the underlying model and only trains the hypernet, obtaining superior results compared to all hypernet-based methods and the full fine-tuning methods

Method	ANLI	HSwag	CB	COPA	RTE	WiC	WSC	WGD	Avg.↑
Full Fine-tuning Methods									
T5	33.4	28.0	63.0	77.9	71.1	50.8	61.0	53.4	54.8
T5 (ICL)	**35.3**	27.5	68.6	70.5	**75.2**	51.7	62.1	52.2	55.4
Parameter-Efficient Fine-tuning Methods									
LoRA	31.8	26.3	48.6	61.4	71.3	51.5	63.0	51.1	50.6
LoraHub	33.9	26.7	56.4	59.4	53.4	51.3	59.8	50.7	49.0
Hypernetwork-based Methods									
HyperTuning	33.6	**33.0**	49.5	74.2	67.4	52.0	64.0	52.9	53.3
HyperTuning+	33.9	30.7	62.1	75.8	72.3	50.8	64.6	**54.5**	55.6
HART	33.6	28.4	70.2	70.1	72.2	50.3	62.3	53.0	55.0
HyperLoRA	34.8	28.3	**71.6**	**82.1**	70.2	**52.8**	**65.5**	53.3	**57.3**

Table 1: Performance on the P3 held-out validation set. We use T5-Large as the underlying model for all methods and report the average multiple-choice accuracy. T5 is multi-task fine-tuned without few-shot inputs while T5 (ICL) utilizes in-context learning that concatenates few-shot inputs and target examples. **Bold** and underline fonts indicate the best results and the second results in each block, respectively.

Method	Avg. ROUGE-L	
	Large	XL
Full Fine-Tuning Methods		
T5	40.6	46.6
T5 (ICL)	**47.6**	**54.0**
Parameter-Efficient Fine-Tuning Methods		
LoRA	42.9	42.9
LoraHub	13.4	-
Hypernetwork-based Methods		
HyperTuning	42.0	45.0
HINT	-	53.2
HART	46.8	50.4
HyperLoRA	47.3	52.8

Table 2: Evaluation results on the Super-Natural Instructions (S-NI) held-out test set. Compared with T5, Tk-Instruct incorporates expert-written explanations for the positive demonstrations.

Method	Needed Training	Avg. Tokens	Avg. EM
Random	No	111.6	25.7
Full Fine-Tuning Methods			
FLAN-T5	No	111.6	27.0
FLAN-T5 (ICL)	No	597.8	37.5
Llama2-7B (ICL)	No	597.8	41.2
Parameter-Efficient Fine-Tuning Methods			
LoRA	Yes	111.6	37.7
LoraHub	Yes	111.6	34.7
Hypernetwork-based Methods			
HyperLoRA	No	111.6	35.8
HyperLoRA†	Yes	111.6	**43.0**
HyperLoRA (Llama2)	No	111.6	41.4

Table 3: Experimental results on the BBH benchmark. All methods employ FLAN-T5-Large as the base language model. HyperLoRA† denotes generalization fine-tuning the generated parameters on the few-shot data. We follow the same settings as Huang et al. (2023) that leverages 5-shot examples per task for all few-shot methods and reports average exact match (EM) metric.

T5 and T5 (ICL). Additionally, our method produces robust parameter-efficient modules for unseen tasks, outperforming the PEFT methods significantly.

Generalization results on Super-Natural Instructions. We use Def+2Pos (task definition and two fixed positive examples) as input for all baselines except T5 which only receives the Def (task definition). More details about the input format can be seen in Appendix C. Table 2 shows the evaluation results of the T5-Large (∼770M) and T5-XL (∼3B) main models on the S-NI held-out test set. HyperLoRA obtains superior results than hypernet-based methods and compared with full fine-tuning methods (HINT can be regarded as a full-parameter fine-tuning method since it jointly trains the hypernet and the underlying model). Due to the limited overlap between the training set and test set in the S-NI dataset, there is a constraint on the pool of available tasks. Consequently, the weight composition methods LoraHub yield unsatisfactory results.

4.3 Few-shot Adaptation

Since the available data is limited in the few-shot adaptation scenario, we directly utilize the model

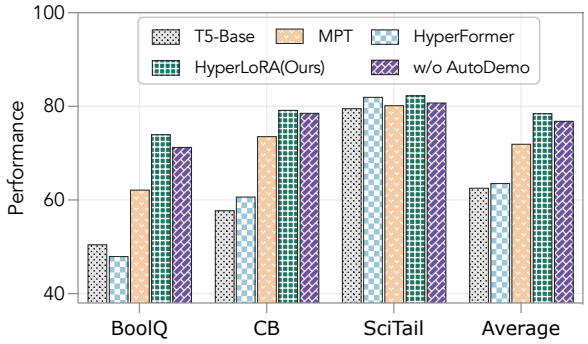


Figure 2: 4-shot learning results on the subset of SuperGLUE (BoolQ, CB, and SciTail). We report the normalized results including the average results. Our HyperLoRA obtains the best performance across all datasets.

pre-trained on instruction-following examples to generate task-specific parameters based on few-shot data without any optimization. Particularly, we also conduct continuous fine-tuning to tune the generated efficient parameters, and the result is denoted as HyperLoRA†.

Few-shot Adaptation on BBH. As shown in Table 3, our HyperLoRA demonstrates superior performance over the LoraHub method even without any training. Notably, while LoraHub employs a reduced number of tokens per example during inference compared to in-context learning, it requires the composition of multiple LoRA modules based on a gradient-free method optimization. In contrast, our HyperLoRA efficiently generates the LoRA parameters without any optimization or additional information. Although the performance of HyperLoRA does not surpass the in-context learning method, the resource consumption is significantly smaller than it (111.6 vs. 597.8 average consumed tokens per example). Moreover, after a slight fine-tuning process, HyperLoRA† outperforms FLAN-T5 (ICL) and LoRA tuning method (LoRA) significantly, which underscores the potential of our method. We present the full results of each task in BBH at Table 10 in Appendix E.3.

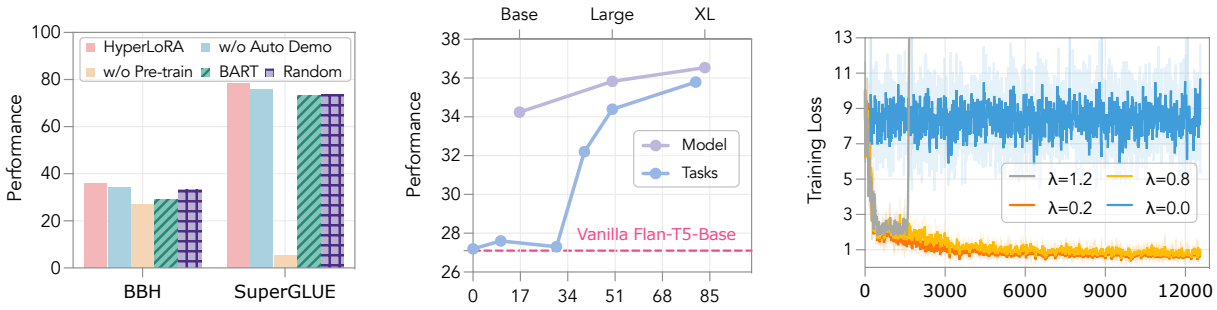
Few-shot Learning on SuperGLUE. Since previous hypernetwork-based methods are hard to address few-shot issues, we compare our methods with a multitask prompt tuning method MPT (Wang et al., 2023c) and a lightweight hypernet-based method HyperFormer (Mahabadi et al., 2021b). As shown in Figure 2, our method HyperLoRA exhibits superior performance and surpasses all compared methods. It is worth noting that our HyperLoRA is an efficient method that neither introduces an increase in consumed tokens (ICL brings doubled token consumption) nor undergoes any additional training (in contrast to other methods that are fine-tuned on few-shot examples). These findings demonstrate that our HyperLoRA is inherently suitable for few-shot adaptation since it effectively leverages the provided few-shot examples as task information to generate task-specific parameters. Moreover, when compared to the in-context learning method, HyperLoRA significantly reduces tokens consumption, and eliminates the need for fine-tuning in contrast to other fine-tuned methods.

4.4 Analysis

Ablation Study. To comprehensively understand and validate the effectiveness of our HyperLoRA, we conduct studies including the pre-training stage (*w/o pre-train*) and automatic demonstration selection strategy (*w/o AutoDemo*) ablations, as well as model configurations exploration. Based on the results in Figure 3 (a), we can condense the following conclusions: (1) The pre-training stage is instrumental in enabling task-specific parameter generation ability. Without the pre-training stage, the performance decreases significantly, especially on SuperGLUE. (2) The automatic demonstration selection strategy can improve the performance consistently. (3) While HyperLoRA is initialized with the BART model (Lewis et al., 2020) or random initialize, a decline appeared, which indicates that it is more effective to initialize the hypernets with the underlying model. The full numerical results can be seen in Appedix E.1

Scaling trends of model and pre-training tasks. We explore the performance of our model on the BBH benchmark spanning different scales of hypernet and varying numbers of pre-training tasks. The experimental results are outlined in Figure 3 (b). Our investigation spans our HyperLoRA ranging from T5-Base to T5-XL, consistently utilizing T5-Large as the underlying model. The results elucidate a positive correlation between the scale of hypernet and its overall performance. In addition, we observe that increasing the number of pre-training tasks generally improves performance, a trend more conspicuous than the improvement brought by scaling model size. This underscores the significance of pre-train hypernets with more diverse data.

Effect of the relative loss weight λ. Training instability is one major challenge for training hypernetworks. To alleviate this, we introduce the truth-



(a) Ablation Study (b) Scaling Trends (c) Effect of Loss Weight λ

Figure 3: Analysis study. (a) Ablation study of method, model type, and model initialization way. (b) Scaling trends of HyperLoRA and the number of pre-training tasks. (c) Comparison of the effect of the relative loss weight λ.

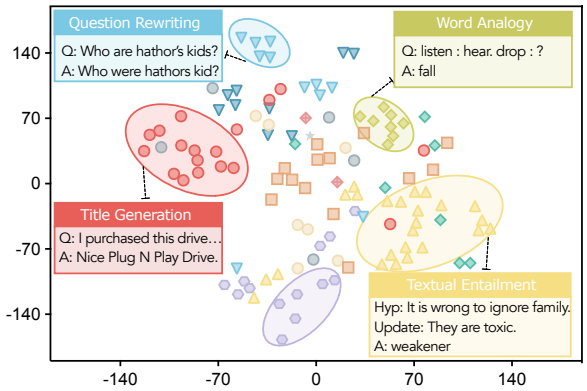


Figure 4: t-SNE visualizations of the generated parameter-efficient modules of 119 S-NI test tasks. Different colors and shapes indicate different tasks.

guided training objective in Section 3.3, which incorporates a weight-space loss controlled by the weight λ. To verify its effectiveness, we conduct experiments with varying values of the relative weight λ, ranging from 0.0 to 1.2, and observe the impact on training loss during the pre-training stage. As shown in Figure 3 (c), the model struggles to fit the training data without the weight-space loss constraint ($\lambda = 0.0$). Fortunately, after introducing the weight-space loss ($\lambda = 0.2$ and $\lambda = 0.8$), the training of the model becomes stable and efficient. However, when the weight λ is excessively large ($\lambda = 1.2$), the model experiences loss spikes, leading to training failures. One possible explanation is that imposing excessive constraints on the representation space of the generated parameters may steer the optimization in incorrect directions.

Visualization Analysis. To understand the effectiveness of our HyperLoRA, we visualize the generated LoRA parameters of 119 S-NI held-out test tasks. To be specific, we first obtain the pooled generated parameter weights for each task and then normalize the weights with L_2-Norm. Afterward, we use t-SNE (Van der Maaten and Hinton, 2008) to map the weights into two-dimensional space, as shown in Figure 4. The visualization results reveal that our HyperLoRA can generate meaningful parameters that similar tasks are closed and distinct tasks are separate. We also demarcate some obvious task clusters and visualize some example cases. However, the boundaries of some tasks are not very clear such as "summarization" and "keyword tagging", resulting in outliers in the figure.

Generalize to Large Language Models. To explore the generalization and robustness of our approach, we utilize LLaMA2-7B (Touvron et al., 2023) as the underlying model, leaving the rest unchanged. We evaluate the BBH benchmark and the results can be seen in Table 3. The surprising results indicate that our method utilized Llama obtains better performance than ICL with lower token costs, which demonstrates the powerful generalization ability of HyperLoRA for different architectures and sizes of the underlying models

5 Conclusion

In this paper, we propose HyperLoRA, a hypernetwork that generates efficient parameters for cross-task generalization. Compared with in-context learning and PEFT, our method is more efficient which decreases the training and storage costs. Furthermore, we propose a paradigm involving multi-task instruction pre-training and generalization fine-tuning for hypernetworks. Through comprehensive experiments and analysis on four benchmark datasets, we have shown HyperLoRA achieves better results than a series of multi-task learning and hypernetwork-based methods. In future, we plan to extend the proposed approach to cross-lingual and cross-modal generalization scenarios and explore the underlying models with larger scales.

Limitations

The research presented in this paper focuses on cross-task generalization in the field of Machine Learning. This work proposes a new paradigm involving pre-train hypernetworks on multi-task instruction-following data and generalization fine-tuning on sparse task data, which enhances the few-shot adaptation performance. However, there are still some limitations to our work. In terms of future societal consequences, this work could contribute to low-resource adaptation and cross-task generalization.

References

Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Févry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged Saeed AlShaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir R. Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022 - System Demonstrations, Dublin, Ireland, May 22-27, 2022*, pages 93–104. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Oscar Chang, Lampros Flokas, and Hod Lipson. 2020. Principled weight initialization for hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7870–7881. Association for Computational Linguistics.

Xiang Chen, Lei Li, Shuofei Qiao, Ningyu Zhang, Chuanqi Tan, Yong Jiang, Fei Huang, and Huajun Chen. 2023. One model for all domains: Collaborative domain-prefix tuning for cross-domain NER. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 5030–5038. ijcai.org.

Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2009–2018. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

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

Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Prakash Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, 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, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *CoRR*, abs/2307.13269.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew E. Peters. 2023. HINT: hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 11272–11288. Association for Computational Linguistics.

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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 6342–6353. Association for Computational Linguistics.

Chen Liang, Nikos Karampatziakis, Tuo Zhao, and Weizhu Chen. 2023. HART: Efficient adaptation via regularized autoregressive parameter generation. In *Submitted to The Twelfth International Conference on Learning Representations*. Under review.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

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

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1022–1035.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 565–576. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11048–11064. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: an adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7654–7673. Association for Computational Linguistics.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. 2023. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 27854–27875. PMLR.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023, pages 687–702. Association for Computational Linguistics.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulic, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023 - System Demonstrations, Singapore, December 6-10, 2023*, pages 149–160. Association for Computational Linguistics.

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

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

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. 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 V. 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 Févry, 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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. 2022. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3949–3969. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5039–5059. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Sid Wang, John Nguyen, Ke Li, and Carole-Jean Wu. 2023a. READ: recurrent adaptation of large transformers. *CoRR*, abs/2305.15348.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5085–5109. Association for Computational Linguistics.

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

Jason Wei, Maarten Bosma, Vincent Y. 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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: selecting influential data for targeted instruction tuning. *CoRR*, abs/2402.04333.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2023. Efficient prompting via dynamic in-context learning. *CoRR*, abs/2305.11170.

A Example Appendix

B Dataset Details

B.1 Dataset in Pre-training Stage

FLAN (Wei et al., 2022) is an instruction-following dataset that incorporates nearly 200 distinct tasks to instruct FLAN-T5 (Chung et al., 2022). We filter the tasks that may conflict with the evaluation datasets and select some representative tasks, resulting in 83 tasks that denote as FLAN* to pre-train our HyperLoRA. Moreover, following (Huang et al., 2023), we control the maximum number of instances per task to be 10,000. The details about the total number of FLAN* are shown in Table 4.

B.2 Evaluation Datasets

There are four datasets in our evaluation experiments: Public Pool of Prompts (P3), Super-Natural Instructions (S-NI), BIG-Bench Hard (BBH), and SuperLGUE. The statistics details of the above datasets are presented in Table 4 and we introduce the details as follows.

Public Pool of Prompts (P3) (Bach et al., 2022) is a collection of prompted English datasets containing 62 NLP tasks. The instances of each task are formatted in manually-written prompt templates which are collected using PromptSource[1]. Since there are no demonstrations in P3, previous hypernetwork-based methods HyperTuning (Phang et al., 2023), HINT (Ivison et al., 2023), and HART (Liang et al., 2023) random sample prompts from the training set and concatenate them to form the hypernetwork input. In contrast, we select 5 prompts for each training task automatically via the methods described in Section 3.3. Moreover, we follow HyperTuning and remove a number of task formulations with longer inputs. We exclude StoryCloze from evaluation as the task is not distributed with training data.

Super-Natural Instructions (S-NI) (Wang et al., 2022) consists of 1,616 tasks spanning 76 diverse categories, including translation, question answering, sentiment analysis, etc. We use v2.6 of S-NI and employ the task definition and two few-shot task examples (denoted as "Def + 2Pos") as the input of HyperLoRA which is aligned with (Phang et al., 2023; Ivison et al., 2023; Liang et al., 2023). Following (Phang et al., 2023), we select the English tasks for training and evaluation. Specifically, we limit the maximum number of training samples per task to 64 and use the first 100 samples in its test set for evaluation following (Wang et al., 2022).

BIG-Bench Hard (BBH) (Suzgun et al., 2023) is a subset of the BIG-Bench and focuses on a suite of 23 challenging tasks that require multi-step reasoning. We follow (Huang et al., 2023) that leverage different 5-shot examples per task as the demonstrations of hypernetwork and employ the exact match (EM) as the evaluation metric.

SuperGLUE (Wang et al., 2019a) is a collection of text classification tasks to test the general language understanding ability. In particular, we consider the natural language inference (NLI) datasets SciTail and CB, and the question answering (QA) dataset BoolQ from SuperGLUE.

C Implementation Details

C.1 HyperLoRA Architecture.

As described in Section 3.2, our HyperLoRA consists of a text encoder and P-generator. The P-generator contains a transformer decoder and a parameter generator. The overall architecture of HyperLoRA is an encoder-decoder and initialized with the underlying model T5 or BART. To enhance stability in the early stages of training, we initialize the parameter generator using a normal distribution with a mean of 0 and a standard deviation of 1e-7.

C.2 Experimental Details

We report the hyper-parameters in the pre-training and fine-tuning stage at Table 5. We conduct all experiments in the same environment (8× 80G A800 GPUs) with Transformers (Wolf et al., 2020) and ZeRO (Rajbhandari et al., 2020). During the pre-training stage, we freeze the underlying model as well as the encoder of HyperLoRA and only tune the P-generator. We use Adam as the optimizer with a learning rate of $5e-5$ and a global batch size of 128. We set the maximum input sequence length of HyperLoRA and the underlying model as 2,048 and 768, respectively. During the generalization fine-tuning stage, we utilize the grid search method to find the best learning rate from 1e-4 to 5e-4 and opt for the largest feasible batch size to maximize resource utilization. For all experiments, we set the rank $r = 16$, $\alpha = 0.8$, and the loss weight $\beta = 0.2$. Due to the multitude of training tasks in the cross-task generalization study, we apply the same strategy as pre-training which

[1] https://github.com/bigscience-workshop/promptsource

Dataset	# Train	# Test	# Train (Task)	# Test (Task)	Metric
FLAN*	307,771	-	83	-	-
P3	17,519,237	15,684	221	8	Multiple-Choice Accuracy
Super-Natural Instructions	48,387	11,810	757	119	ROUGEL
BBH	-	27	-	3,811	Exact Match
SuperGLUE	-	3	-	4,630	Multiple

Table 4: Details about the number of instances and tasks of the pre-train and evaluation datasets.

Hyper-parameters	Pre-training	P3	S-NI
global batch size	128	128	96
training steps	38,000	130,000	32,256
learning rate	5e-5	5e-5	5e-5
learning scheduler	cosine	cosine	cosine
sequence length	768	384	2048
hypernet sequence length	2048	1024	2048
output sequence length	512	128	512
LoRA rank	16	16	16
LoRA alpha	8	8	8

Table 5: Hyper-parameter settings in pre-training and fine-tuning stages.

only tunes the P-generator, the resulting model is denoted as HyperLoRA. In the few-shot adaptation scenario, we utilize the few-shot examples as the input demonstrations of HyperLoRA to generate parameter-efficient modules without any weights updating. In addition, we can also conduct fast task generalization fine-tuning which only tunes the generated parameters to further improve the performance and result in HyperLoRA†.

C.3 Examples of Inputs

Few-shot Inputs without Task Description

<x> Inputs 1
Target 1 <y>

<x> Inputs 2
Target 2 <y>

Few-shot Inputs with Task Description

<x> Task Description

<x> Inputs 1
Target 1 <y>

<x> Inputs 2
Target 2 <y>

D Additional Analysis of HyperLoRA

D.1 Comparison of the Hypernetwork-based methods

Table 6 summarizes the differences between our HyperLoRA and three hypernet-based methods HyperTuning, HINT, and HART. Compared with these methods, we pre-train the hypernetwork with instruction data instead of the general corpus, which endows it with the few-shot adaptation ability. While weight-freezing prevents catastrophic forgetting and saves the storage cost, we freeze the text encoder and the underlying model during the pre-train and fine-tune stages. Additionally, we design an automatic demonstration selection strategy and a weight-space constraint objective to enhance the effectiveness and training stability.

D.2 Analysis of the Computation Costs.

The large computation amount of our HyperLoRA mainly occurs during the pre-training stage, but it remains more efficient than full fine-tuning methods because only part of hypernetwork (decoder and parameter generator) is optimized. During the inference stage, both the compute cost and memory cost of hypernetwork is less than full fine-tuning methods, as the instruction is no longer processed with every sample for hypernetwork . To provide a quantitative comparison, as illustrated in HINT (Ivison et al., 2023), the full fine-tuning method requires roughly $Nn(i+t+o)$ FLOPs, while the hypernetwork-based method uses roughly $tN + nN(i+o)$ FLOPs, where t is the task instruction length, o is the output length, n is the number of same-task samples and N is the number of model parameters. These formulations highlight the compute cost of hypernetwork is $\propto t+n$ as opposed to $\propto tn$.

E Additional Experimental Results

E.1 Detailed Ablation Results

During the ablation study, we run with five different seeds (6, 42, 99, 1234, 2023, 6617) and report the

Method	Hypernet Architecture	Freeze Underlying	Instruction-Training	Demonstration Selection	Stability Training	Few-shot Adapatation
HyperTuning	Encoder-Decoder	Yes	No	Manual& Random	No	No
HyperTuning+	Encoder-Decoder	No	No	Manual& Random	No	No
HINT	Encoder-Decoder	No	No	Manual& Random	No	No
HART	Encoder-Decoder	Yes	No	Manual& Random	No	No
HyperLoRA	Encoder-Decoder	Yes	Yes	Automatic	Yes	Yes

Table 6: Comparison of the Hypernetwork-based methods.

	BBH	SuperGLUE
HyperLoRA	35.8(0.2)	78.5(0.6)
w/o AutoDemo	34.2(0.2)	76.8(0.6)
w/o Pre-train	27.2(0.8)	5.2(2.3)
BART Init.	29.0(0.2)	73.4(0.77)
Random Init.	33.3(0.2)	73.8(0.9)

Table 7: The numerical results of ablation study. For each item, we run with five random seeds (6, 42, 99, 1234, 2023, 6617) and report the mean (and standard deviation) results.

average results in Table 7. To demonstrate our gradient-based demonstration selection method, we report the full results whether we apply this method in each task at Table 8. The results reveal that our method provides a performance gain on each task consistently.

Task	w/ AutoDemo	w/o AutoDemo	Diff
P3	57.3	56.3	1.0
S-NI	47.3	46.0	1.3
BBH	35.8	34.2	1.6
SuperGLUE	78.5	76.8	1.7

Table 8: The full comparisons .

E.2 Generalization on GLUE Benchmark

To explore the effectiveness of the fast task generalization fine-tuning method, we conduct a cross-task experiment on the GLUE (Wang et al., 2019b) dataset. GLUE is a collection of text classification tasks to test the general language understanding ability. We compare our methods with full fine-tuned T5 model (Raffel et al., 2020a), PEFT methods LoRA (Hu et al., 2022), READ (Wang et al., 2023a) and MPT (Wang et al., 2023c), and hypernetwork-based methods including Compacter++ (Mahabadi et al., 2021a), HyperFormer (Mahabadi et al., 2021b) and HyperPrompt (He et al., 2022). The results can be seen in Table 9. Our method HyperLoRA achieves comparable performance with the full fine-tuning methods and is superior to all of the hypernetwork-based methods. However, direct parameter-efficient fine-tuning on downstream tasks leads to suboptimal performance, 4.2% behind HyperLoRA, which reveals that utilizing the parameters generated by HyperLoRA as initialization for downstream tasks with adequate data improves performance significantly. Additionally, we can see that the fast generalization fine-tuning method HyperLoRA† performs better than HyperLoRA and all other methods, which demonstrates the effectiveness of the approach.

E.3 Full BBH Results

We report the full evaluation results on the BIG-Bench Hard (BBH) benchmark at Table 10.

Model	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg.↑
Full Fine-Tuning Methods									
$T5_{Base}$	49.8	94.6	89.8/92.5	90.7/**90.5**	91.9/89.2	88.5	93.3	85.0	85.5
$T5_{Large}$	59.4	96.6	90.7/93.3	90.6/90.4	92.3/89.8	**90.8**	**95.2**	**90.8**	88.3
Parameter-Efficient Fine-Tuning Methods									
$LoRA_{Large}$	60.0	93.9	92.1/94.3	76.8/73.3	91.8/91.5	89.5	94.3	84.8	85.7
$READ_{Large}$	54.1	93.9	87.7/-	89.3/-	88.6/-	87.3	93.7	-	85.7
MPT_{Base}	63.5	93.3	89.2/-	90.0/-	90.4/-	84.3	93.0	82.7	85.8
Hypernetwork-based Methods									
$Compacter{+}{+}_{Base}$	61.3	93.8	90.7/93.3	90.2/86.9	90.5/90.9	85.7	93.1	74.8	86.5
$HyperFormer_{Base}$	61.3	93.8	90.6/93.3	90.1/87.2	89.6/89.0	86.3	92.8	78.3	86.6
$HyperFormer{+}{+}_{Base}$	63.7	94.0	89.7/92.6	90.3/87.2	90.0/89.7	85.7	93.0	75.4	86.5
$HyperPrompt_{Large}$	57.5	**96.7**	91.2/93.6	90.1/87.0	91.9/92.0	90.3	95.0	87.7	87.5
$HyperFormer{+}{+}_{Large}$	58.9	95.7	90.0/92.7	90.7/87.7	91.6/91.5	89.8	94.5	87.8	87.3
$\texttt{HyperLoRA}_{Large}$	60.6	95.6	88.9/92.0	90.4/87.8	91.3/91.0	89.0	94.0	87.0	88.0
$\texttt{HyperLoRA}^{\dagger}_{Large}$	**68.8**	96.4	**92.6/94.5**	**90.9**/87.9	**92.9/92.8**	89.5	94.2	89.1	**90.0**

Table 9: Performance of the models on the GLUE tasks. For MNLI, we report accuracy on the matched validation set. For MRPC and QQP, we report accuracy and F1. For STS-B, we report Pearson and Spearman correlation coefficients. For CoLA, we report Matthews correlation. For all other tasks, we report accuracy. We use T5-large as the initial model to train our HyperLoRA. **Bold** and underline fonts indicate the best results and the second results in each block, respectively.

Task	Random	T5	T5 (ICL)	LoRA	LoraHub	HyperLoRA	HyperLoRA†
Boolean Expressions	50.0	54.0	58.7	56.0	56.0	56.0	61.3
Causal Judgement	50.0	57.5	56.3	55.6	58.9	54.0	52.9
Date Understanding	17.2	15.3	22.7	35.8	29.6	28.0	76.0
Disambiguation	33.2	0.0	69.3	68.0	46.0	33.3	56.0
Dyck Languages	1.2	1.3	7.3	22.2	0.3	2.7	23.3
Formal Fallacies	25.0	51.3	58.0	53.6	52.1	52.0	57.3
Geometric Shapes	11.6	6.7	18.7	24	7.5	7.3	31.3
Hyperbaton	50.0	6.7	74.0	55.3	57.5	65.3	68.7
$\text{Logical Deduction}_{avg}$	22.5	11.3	44.4	43.6	42.7	44.9	43.6
Movie Recommendation	25.0	62.7	52.7	51.5	61.1	53.3	51.3
Multistep Arithmetic	0	0.7	0.7	0.2	0.7	0.7	0.7
Navigate	50.0	47.3	44.0	48.0	46.1	49.3	51.3
Object Counting	0.0	34.7	32.0	38.7	35.0	34.7	36.7
Penguins in a Table	0.0	43.5	39.1	36.2	43.9	50.0	34.8
Reasoning about Colored Objects	11.9	32.0	38.7	39.6	36.5	43.3	33.3
Ruin Names	25.0	23.3	18.7	37.8	21.0	24.7	65.3
Salient Translation Error Detection	16.7	37.3	46.0	16.0	37.3	46.0	18.7
Snarks	50.0	50.0	55.1	55.6	51.8	55.1	57.7
Sports Understanding	50.0	56.0	56.0	56.5	48.3	57.3	44.7
Temporal Sequences	25.0	16.7	26.7	25.1	18.7	12.7	86.0
$\text{Tracking Shuffled Objects}_{avg}$	22.5	14.5	16.5	18.2	16.0	16.5	21.8
Web of Lies	50.0	54.0	54.0	52.7	53.0	56.0	52.7
Word Sorting	0.0	1.3	0.7	4.9	1.1	1.3	4.0
Average Performance per Task	25.7	27.0	37.5	37.7	34.7	35.8	**43.0**

Table 10: Full experimental results on the BBH benchmark.