HyperPrompt: Prompt-based Task-Conditioning of Transformers

Yun He [* 1] **Huaixiu Steven Zheng** [* 2] **Yi Tay** [2] **Jai Gupta** [2] **Yu Du** [2] **Vamsi Aribandi** [2] **Zhe Zhao** [2] **YaGuang Li** [2] **Zhao Chen** [3] **Donald Metzler** [2] **Heng-Tze Cheng** [2] **Ed H. Chi** [2]

Abstract

Prompt-Tuning is a new paradigm for finetuning pre-trained language models in a parameter-efficient way. Here, we explore the use of Hyper-Networks to generate hyper-prompts: we propose HyperPrompt, a novel architecture for prompt-based task-conditioning of self-attention in Transformers. The hyper-prompts are end-to-end learnable via generation by a HyperNetwork. Hyper-Prompt allows the network to learn task-specific feature maps where the hyper-prompts serve as task global memories for the queries to attend to, at the same time enabling flexible information sharing among tasks. We show that HyperPrompt is competitive against strong multi-task learning baselines with as few as 0.14% of additional task-conditioning parameters, achieving great parameter and computational efficiency. Through extensive empirical experiments, we demonstrate that HyperPrompt can achieve superior performances over strong T5 multi-task learning baselines and parameter-efficient adapter variants including Prompt-Tuning and HyperFormer++ on Natural Language Understanding benchmarks of GLUE and SuperGLUE across many model sizes.

1. Introduction

Prompt-Tuning (Lester et al., 2021), learning to condition large language models with soft learnable memory tokens, have recently garnered attention owing to their ability for parameter-efficient finetuning. Prompts are lightly tuned, allowing the model to be trained quickly since the main body of the pretrained model is kept frozen. To this end, this paradigm is strongly reminiscent of adapter layers (Houlsby

---

*Equal contribution [1]Texas A&M University, work done as an intern at Google [2]Google Research [3]Waymo LLC. Correspondence to: Huaixiu Steven Zheng <stevenzheng@google.com>, Yi Tay <yitay@google.com>.
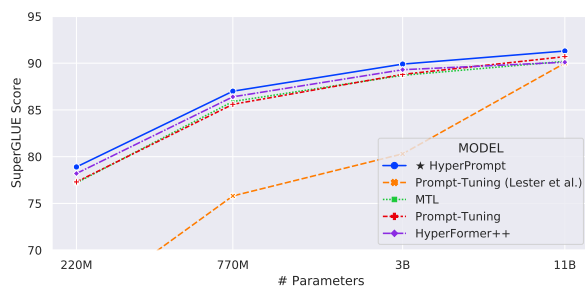
Figure 1. HyperPrompt achieves state-of-the-art performance on SuperGLUE for T5 models up to XXL. Prompt-tuning (Lester et al., 2021) with tuning prompt parameters only achieves competitive performance against multi-task learning (MTL) baseline for the 11B parameter model with a big performance gap for smaller models. HyperPrompt-Global outperforms the strong parameter-efficient adapter variant HyperFormer++ (Karimi Mahabadi et al., 2021), the MTL baseline, and the full fine-tuning of Prompt-Tuning (our implementation) across model sizes with a large margin [e.g. 91.3 vs 90.2 (MTL) for T5 XXL].

et al., 2019a; Karimi Mahabadi et al., 2021; Zaken et al., 2021; He et al., 2021) which are also efficiently finetuned.

We introduce HyperPrompt, a natural but novel extension of Prompt-Tuning to multi-task learning (MTL) for language. HyperPrompt introduces task-conditioned hyper-prompts that conditions the model on task-specific information for constructing these prompts. Hyper-prompts are injected to the keys and values in the self-attention module, reminiscent of memory augmented Transformers (Sukhbaatar et al., 2019). This mitigates the cost of having prompts pass through the standard FFN layers in Transformers and serves as additional task-specific memory tokens for queries to attend to.

We further improve upon this by introducing task-aware and layer-aware HyperNetworks (Ha et al., 2017) that parameterize and generate weights for the prompt generation process. The usage of HyperNetwork imbues our model with the necessary flexibility and expressiveness, especially when it comes to incorporating task-specific and layer-specific information to the network. Meanwhile, HyperPrompt remains very parameter and computational efficient and friendly to multi-task scaling: the additional parameters scale sub-

linearly with, and are independent of the number of tasks in practice. While Hypernetworks have enjoyed some success in learning adapters (Karimi Mahabadi et al., 2021; Tay et al., 2020) and/or continual learning (von Oswald et al., 2019), we note that this is the first exploration of HyperNetworks as a prompt generator.

Contrary to prior work, we additionally propose to finetune the entire network instead of only the hyper-prompts. We make several compelling arguments for this. Firstly, Lester et al. (2021) shows that parameter efficient Prompt-Tuning only shines for large (e.g., 11B) models and substantially pales in comparison to fine-tuning when the model is moderately parameterized (e.g., 220M). Secondly, finetuning only adaptive parameters (e.g., prompts/adapters) simply presents an illusion of efficiency (Dehghani et al., 2021). In reality, the FLOPs incurred by the model is still identical on the forward pass, which saves no compute during inference. Parameter counts, especially when including only prompts and adapters, are not the only measurement of computational efficiency. Instead, the FLOPs and training time should be considered together to provide a holistic view.

Our Contributions   Our main contributions include:

- We propose a novel HyperPrompt Transformer architecture with learnable hyper-prompts for multi-task fine-tuning with great parameter and computational efficiency.

- We demonstrate that for difficult tasks, it is crucial to fine-tune the task-specific parameters together with the backbone model to achieve Pareto efficiency on all tasks.

- We explore HyperNetworks as a prompt generator, and inject hyper-prompts into the self-attention module as global task memory tokens.

- HyperPrompt outperforms state-of-the-art parameter-efficient T5 models (Raffel et al., 2019) using Prompt-Tuning or adapters on well-established benchmarks such as SuperGLUE and GLUE, across all explored model sizes (see Figure 1).

2. Problem Statement

We consider the general setting of multi-task learning for a set of tasks $\{\mathcal{D}_\tau\}_{\tau=1}^T$, where T is the total number of tasks and $\{\mathcal{D}_\tau\} = \{x_\tau^{(n)}, y_\tau^{(n)}\}_{n=1}^{N_\tau}$ indicates the corresponding training set of the τ-th task with N_τ samples. We assume that a pre-trained Transformer model $f_\theta(\cdot)$ (e.g., T5) is given, where the model is parameterized by θ. To tackle such multi-task learning problem with $f_\theta(\cdot)$, we minimize the following objective function $\mathcal{L}(\theta) = \sum_{\tau=1}^T \sum_{n=1}^{N_\tau} C(f_\theta(x_\tau^{(n)}), y_\tau^{(n)})$, where $C(\cdot, \cdot)$ is

typically the cross-entropy loss and $f_\theta(x_\tau^{(n)})$ is the output for training sample $x_\tau^{(n)}$.

Transformer-based pre-trained language models such as T5 (Raffel et al., 2019) and BART (Lewis et al., 2020) are unified text-to-text frameworks where all tasks share the same encoder-decoder architecture – $\{\{x_\tau^{(n)}\}_{n=1}^{N_\tau}\}_{\tau=1}^T$ are fed into the same encoder and $\{\{\hat{y}_\tau^{(n)}\}_{n=1}^{N_\tau}\}_{\tau=1}^T$ are generated by the same decoder. For such universal modules, multi-task learning simply corresponds to mixing task data sets together and there is no task-specific classification or regression networks for each task as in encoder-only modules Devlin et al. (2019); Liu et al. (2019b).

Previous work Raffel et al. (2019) shows that co-learning all tasks together on a pre-trained Transformer model is inferior to fine-tuning on each task separately. A possible reason is that θ is task-agnostic (i.e., all parameters are shared) and hence task-specific information is not well captured which can be especially true for low-resource tasks. Therefore, a natural way to improve the performance of Transformers on multi-task learning is to introduce a set of task-conditioned parameters $\{\delta_\tau\}_{\tau=1}^T$ into $f_\theta(.)$. The objective function can be updated as $\mathcal{L}(\theta, \{\delta_\tau\}_{\tau=1}^T) = \sum_{\tau=1}^T \sum_{n=1}^{N_\tau} C(f_{\theta,\delta_\tau}(x_\tau^{(n)}), y_\tau^{(n)})$, where δ_τ is the task-specific parameterization for the τ-th task. During training, both θ and $\{\delta_\tau\}_{\tau=1}^T$ are updated via back-propagation because we observe a large performance drop in SuperGLUE when backbone model θ is frozen and only task-conditioned parameters are tuned, as done in Karimi Mahabadi et al. (2021), which will be detailed in Section 4.3.

To this end, our goal is to design task-conditioned parameterization of Transformer models to achieve *greater parameter and computational efficiency as well as Pareto efficiency for multi-task learning*. More explicitly, we have two goals: (1) improving the finetuning performance of most tasks in $\{\mathcal{D}_\tau\}_{\tau=1}^T$ by introducing task-conditioned parameters $\{\delta_\tau\}_{\tau=1}^T$ into $f_\theta(.)$ and (2) under the constraint that $\sum_\tau \|\{\delta_\tau\}_{\tau=1}^T\|_0 \ll \|\theta\|_0$, which means that the model capacity will not be significantly increased. And the computational cost would not increase substantially either.

3. Methods

In this section, we introduce HyperPrompt which has three variants: HyperPrompt-Share, HyperPrompt-Sep and HyperPrompt-Global (Figure 2). We follow two key design principles to formulate HyperPrompt: (1) injecting task-conditioning into self-attention module for better computational efficiency and more expressive power via token-level interactions, and (2) using HyperNetworks to simultaneously improve the parameter efficiency and allow a flexible degree of task sharing for better generalization.
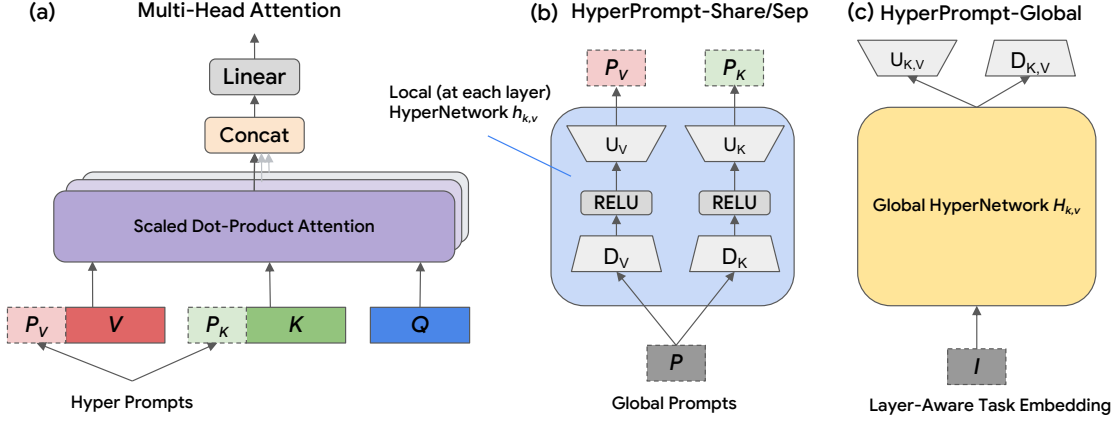
Figure 2. HyperPrompt framework: (a) in each Transformer block, task-specific hyper-prompts $P_{K,V}$ are prepended to the original key K and value V for the query Q to attend to, (b) in HyperPrompt-Share/Sep, global prompts P are used to generate the hyper-prompts $P_{K,V}$ through local HyperNetworks $h_{k,v}$ at each Transformer layer, which consists of a down-projection matrix $D_{K,V}$, a RELU layer and a up-project matrix $U_{K,V}$, (c) in HyperPrompt-Global, all the local HyperNetworks $(D_{K,V}, U_{K,V})$ are generated by global HyperNetworks $H_{k,v}$ using layer-aware task embeddings I as task-specific inputs (see Section 3.3 for details).

3.1. Prompt-Based Task-Conditioned Transformer

Previous adapter-based methods (Karimi Mahabadi et al., 2021; Tay et al., 2020) for multi-task learning normally add an adapter (i.e., dense-relu-dense network) for each task after the feed-forward layers at every Transformer block. Instead, the key idea of our approach is to prepend l task-conditioned trainable vectors to the keys and values of the multihead self-attention layer at every Transformer block, where the task-specific attention feature maps are jointly learned with the task-agnostic representation.

The idea of prepending learnable prompts to the network is explored before by Li & Liang (2021); Lester et al. (2021); Liu et al. (2021) for single-task fine-tuning. We first introduce and expand this idea for multi-task learning in this subsection. Specifically, we design a novel method called HyperPrompt following the design principle #1 of injecting hyper-prompts into self-attention and #2 using HyperNetworks as generators for hyper-prompts.

At a multihead self-attention layer, the original key, value and query are calculated as $K_\tau = X_\tau W_k$, $V_\tau = X_\tau W_v$, $Q_\tau = X_\tau W_q$, where $X_\tau \in \mathbb{R}^{L \times d}$ is the input sequence of a training sample from the τ-th task, L is the sequence length, d is the model dimension. $W_k \in \mathbb{R}^{d \times h \times d_h}$, $W_v \in \mathbb{R}^{d \times h \times d_h}$ and $W_q \in \mathbb{R}^{d \times h \times d_h}$ project the input into original key $K_\tau \in \mathbb{R}^{L \times h \times d_h}$, value $V_\tau \in \mathbb{R}^{L \times h \times d_h}$ and query $Q_\tau \in \mathbb{R}^{L \times h \times d_h}$, h is the number of heads, d_h is the dimension of each head and typically set to d/h to save parameters.

To learn the task-specific information for the τ-th task, we have l trainable d-dimensional vectors as the hyper-prompts

for the key and the value respectively, denoted as $P_{\tau,k} \in \mathbb{R}^{l \times h \times d_h}$ and $P_{\tau,v} \in \mathbb{R}^{l \times h \times d_h}$, as shown in Figure 2(a). Then, the hyper-prompts are concatenated with the original key and value:

$$K_\tau' = \text{concat}(P_{\tau,k}, K_\tau) \quad (1)$$

$$V_\tau' = \text{concat}(P_{\tau,v}, V_\tau) \quad (2)$$

where the new key (value) K_τ' $(V_\tau') \in \mathbb{R}^{(l+L) \times h \times d_h}$ are used to compute the multihead self-attention.

After that, the multihead self-attention can be operated: $O_\tau = \text{Attention}(Q_\tau, K_\tau', V_\tau') = \text{softmax}(Q_\tau K_\tau'^T) V_\tau'$ where $O_\tau \in \mathbb{R}^{L \times d}$ is the output of multihead attention.

The hyper-prompts benefit Transformers for multi-task learning in two ways: (1) Prompt for key $P_{\tau,k}$ is prepended with the original key and will participate in the calculation of attention feature map: $\text{softmax}(Q_\tau K_\tau'^T)$. $P_{\tau,k}$ directly interacts (matrix multiplication) with the original query Q_τ, allowing tokens to acquire task-specific semantics. (2) Prompt for value $P_{\tau,v}$ is prepended with the original value and will be absorbed into the self-attention output O_τ, where each position in O_τ is the weighted-sum of vectors in V_τ' with weights from the attention scores. This way, $P_{\tau,v}$ can serve as task-specific memories for multihead attention to retrieve information from.

3.2. HyperPrompt

How to obtain the prompts for the m-th Transformer block? A straightforward way is to directly initialize $P_{\tau,k}^m$ and $P_{\tau,v}^m$. However, this way is parameter-inefficient, as it scales linearly with both the number of tasks T and the number layers M as $\mathcal{O}(T \times M)$.

Instead, we initialize a global[1] prompt $\boldsymbol{P}_\tau$ for each task and apply local HyperNetworks at every Transformer block to project this prompt into $\{\boldsymbol{P}_{\tau,k}^m\}_{m=1}^M$ and $\{\boldsymbol{P}_{\tau,v}^m\}_{m=1}^M$.

Global Prompts. Specifically, we initialize a set of global prompts $\{\boldsymbol{P}_\tau\}_{\tau=1}^T$, where $\boldsymbol{P}_\tau \in \mathbb{R}^{l \times d}$ is a trainable matrix to learn the task-specific information of the τ-th task, d is the model dimension and l is the length of the prompt.

Local HyperNetworks. At the m-th Transformer block, we apply two local HyperNetworks h_k^m and h_v^m to transform the global prompt $\boldsymbol{P}_\tau$ into layer-specific and task-specific prompts as shown in Figure 2(b):

$$\boldsymbol{P}_{\tau,k}^m = h_k^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_k^m(\text{Relu}(\boldsymbol{D}_k^m(\boldsymbol{P}_\tau))), \quad (3)$$

$$\boldsymbol{P}_{\tau,v}^m = h_v^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_v^m(\text{Relu}(\boldsymbol{D}_v^m(\boldsymbol{P}_\tau))), \quad (4)$$

where $\boldsymbol{P}_{\tau,k/v}^m \in \mathbb{R}^{l \times h \times d_h}$. We call these generated prompts *hyper-prompts* to distinguish from global prompts.

In particular, to limit the number of parameters, the local HyperNetworks are designed using a bottleneck architecture: $\boldsymbol{D}_{k/v}^m \in \mathbb{R}^{d \times b}$ and $\boldsymbol{U}_{k/v}^m \in \mathbb{R}^{b \times h \times d_h}$ are down-projection and up-projection matrices, respectively. b is the bottleneck dimension satisfying $b \ll d$.

HyperPrompt-Share. We first have all tasks share the same two local HyperNetworks defined by the down-project matrices $\boldsymbol{D}_k^m$ and $\boldsymbol{D}_v^m$, and the up-project matrices $\boldsymbol{U}_k^m$ and $\boldsymbol{U}_v^m$. We refer to this design choice as HyperPrompt-Share.

Despite the saving of parameters, one drawback of HyperPrompt-Share is that the task conflicts could arise given the limited model capacity (Wu et al., 2020; Wang et al., 2020) of the shared local HyperNetworks.

HyperPrompt-Sep. In the opposite extreme of HyperPrompt-Share, each task can have its own local HyperNetworks $h_{\tau,k}^m(\boldsymbol{P}_\tau)$ and $h_{\tau,v}^m(\boldsymbol{P}_\tau)$ as following:

$$\boldsymbol{P}_{\tau,k}^m = h_{\tau,k}^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_{\tau,k}^m(\text{Relu}(\boldsymbol{D}_{\tau,k}^m(\boldsymbol{P}_\tau))), \quad (5)$$

$$\boldsymbol{P}_{\tau,v}^m = h_{\tau,v}^m(\boldsymbol{P}_\tau) = \boldsymbol{U}_{\tau,v}^m(\text{Relu}(\boldsymbol{D}_{\tau,v}^m(\boldsymbol{P}_\tau))), \quad (6)$$

where $\boldsymbol{D}_{\tau,k/v}^m$ and $\boldsymbol{U}_{\tau,k/v}^m$ are down-projection and up-projection matrices for the τ task, respectively. In this case, each task hyper-prompt is trained independently and hence there is no information sharing.

3.3. HyperPrompt-Global

We further propose a novel design of *HyperPrompt-Global* to flexibly share information and knowledge among tasks and blocks while maintaining a low parameter cost. As shown in Figure 2(c), the key idea of HyperPrompt-Global is to generate the local HyperNetworks using the same

global HyperNetwork shared by all tasks and all Transformer blocks.

Layer-Aware Task Embedding. Following the same recipe in Karimi Mahabadi et al. (2021), we define a layer-aware task embedding for better generalization. Let $k_\tau \in \mathbb{R}^{t'}$ denote the task embedding for the τ task and t' is the dimension. To capture the layer-specific information, layer embedding $z_m \in \mathbb{R}^{t'}$ is introduced. After that, a task projection network $h_t(\cdot, \cdot)$ is applied to fuse the task embedding and the layer embedding into the final layer-aware task embedding $\boldsymbol{I}_\tau^m = h_t(k_\tau, z_m)$, where $\boldsymbol{I}_\tau^m$ is the input to the shared global HyperNetworks as shown in Figure 2(c). h_t is a MLP consisting of two feed-forward layers and a ReLU non-linearity, which takes the concatenation of k_τ and z_m as input.

Global HyperNetworks. $H_k(\cdot)$ generates the weight matrices $(\boldsymbol{U}_{\tau,k}^m, \boldsymbol{D}_{\tau,k}^m)$ in the local HyperNetworks of key hyper-prompts and another global HyperNetwork $H_v(\cdot)$ generates the weight matrices $(\boldsymbol{U}_{\tau,v}^m, \boldsymbol{D}_{\tau,v}^m)$ in the local HyperNetworks of value hyper-prompts:

$$(\boldsymbol{U}_{\tau,k}^m, \boldsymbol{D}_{\tau,k}^m) = H_k(\boldsymbol{I}_\tau^m) = (\boldsymbol{W}^{U_k}, \boldsymbol{W}^{D_k})\boldsymbol{I}_\tau^m, \quad (7)$$

$$(\boldsymbol{U}_{\tau,v}^m, \boldsymbol{D}_{\tau,v}^m) = H_v(\boldsymbol{I}_\tau^m) = (\boldsymbol{W}^{U_v}, \boldsymbol{W}^{D_v})\boldsymbol{I}_\tau^m, \quad (8)$$

where $\boldsymbol{I}_\tau^m \in \mathbb{R}^t$ is the layer-aware task embedding for the τ task at the m-th block. $\boldsymbol{W}^{D_k} \in \mathbb{R}^{(d \times b) \times t}$, $\boldsymbol{W}^{D_v} \in \mathbb{R}^{(d \times b) \times t}$, $\boldsymbol{W}^{U_k} \in \mathbb{R}^{(b \times h \times d_h) \times t}$ and $\boldsymbol{W}^{U_v} \in \mathbb{R}^{(b \times h \times d_h) \times t}$ are the weight matrices of $H_k(\cdot)$ and $H_v(\cdot)$.

Given that $\boldsymbol{U}_{\tau,k/v}^m$, and $\boldsymbol{D}_{\tau,k/v}^m$ are generated by the global HyperNetworks, we project the global prompts $\boldsymbol{P}_{\tau,k/v}$ into hyper-promtps $\boldsymbol{P}_{\tau,k/v}^m$ following Eqs. 5 and 6. Finally, the hyper-prompts $\boldsymbol{P}_{\tau,k/v}^m$ are prepended with original key and value at every self-attention layer as shown in Figure 2(a) to calculate the task-conditioned attention scores.

Using global HyperNetworks to generate the projection networks has two benefits:

1. It enables a more flexible way to share information across tasks and layers: the transformation matrices are decomposed into $H_{k/v}(\cdot)$ that are shared by all tasks and all layers. Therefore, the model can adjust the degree of information sharing across tasks and layers through learning the appropriate parameter values in $H_{k/v}(\cdot)$ during the end-to-end training.

2. A parameter-efficient task conditioned parameterization is enabled. The number of extra task-conditioned parameters doesn't depend on the number of layers M, and scales sub-linearly with respect to the total number of tasks T. In practice, since task embeddings

---

[1] we term it *global* because it is independent of the layer number as opposed to layer-dependent prompt $\boldsymbol{P}_\tau^m$.

and task prompts have far fewer parameters than the global HyperNetworks, the additional task-conditioned parameters is almost independent of T.

3.4. Parameter Efficiency of HyperPrompt

As shown in A.1, the total number of additional parameters from HyperPrompt-Global is $dlT + 4(bdt) + Tt' + Mt' + (2t' + t)e$, where d is the model dimension, l is the length of the prompts, T is the total number of tasks, b is the bottleneck dimension of the weight matrices of the local HyperNetworks, d is the model dimension, t'/t is the dimension of the raw/final layer-aware task embedding, and e is the hidden dimension of $h_{k/v}$. Therefore, the space complexity is $\mathcal{O}(d(lT + 4bt))$, given that in practice $M \sim T$, $t' \ll dl$, and $e \ll bd$. This leads to a sub-linear scaling with respect to T.

Furthermore, T is typical $\sim \mathcal{O}(10)$ for multi-task learning. A reasonable $l \sim \mathcal{O}(10)$ is required to achieve the optimal performance, which will be detailed in Section 4.7. On the other hand, typical values for $b \sim 24$ and $t \geq 32$, and therefore $4bt \gg lT$ is satisfied in most cases. Hence, the space complexity could be further simplified as $\mathcal{O}(bdt)$. In conclusion, the space complexity of HyperPrompt-Global mainly comes from the global HyperNetworks and is practically independent of the prompt length l, the number of Transformer layers M, and the number of tasks T.

4. Experiments

4.1. Experimental Setup

Datasets. We evaluate the performance of the models on GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) respectively. Each of them is a collection of text classification tasks to test the general language understanding ability. Specifically, the tasks include: sentence acceptability (CoLA), sentiment analysis (SST-2), paraphrasing/sentence similarity (MRPC, STS-B and QQP), natural language inference (MNLI, QNLI, RTE and CB), coreference resolution (WSC), sentence completion (COPA), word sense disambiguation (WIC) and question answering (MultiRC and ReCoRD, BoolQ).

Transformers. Following previous work Karimi Mahabadi et al. (2021) and Tay et al. (2020), our models are built on top of the state-of-the-art Transformer model T5 (Raffel et al., 2019), which uses encoder-decoder architecture from Vaswani et al. (2017). We use already pre-trained T5 with sizes from Base (220M parameters) to XXL (11B).

Evaluation. We save a checkpoint every 2000 steps for all models and follow the same convention as Raffel et al. (2019) in selecting the best checkpoint for each task. The emphasis of our evaluation is not to find the best single

checkpoint for all tasks but to test the model's ability of transfer learning among the co-trained tasks. We first calculate the average of all metrics for each task and then report the average of all tasks for GLUE and SuperGLUE.

Baselines. We compare our proposed HyperPrompt-Share/Sep/Global with vanilla T5 models (Raffel et al., 2019) for multi-task learning, which is referred to *MTL*. Another baseline is *Vanilla Adapter* proposed in Houlsby et al. (2019b) that add adapters modules for each task after each of the the two feed-forward modules in each Transformer block of the T5 model. The state-of-the-art adapter-based method for multi-task learning is *HyperFormer++* proposed in Karimi Mahabadi et al. (2021) that use HyperNetworks to generate adapters for each task and add them after the feed-forward modules following Houlsby et al. (2019b). In addition, *Prompt-Tuning* (Lester et al., 2021) is originally for parameter-efficient single-task fine-tuning and only prepends prompts to the input word embeddings in the first layer. We slightly modify it by initializing and prepending prompts for each task respectively so that *Prompt-Tuning* can be applied to multi-task learning.

We defer additional details of the experiments to A.2

4.2. Key Results

Figure 1 provides an overall summary of the results of HyperPrompt. Previous prompt-tuning (Lester et al., 2021; Li & Liang, 2021) methods focus on parameter-efficient single-task fine-tuning and hence freeze the backbone and only fine-tune the prompts. Their experiments show that the performance of only tuning the prompts can match the full model training with a very large 11B model (Figure 1), but substantially pales for moderate model sizes.

Our HyperPrompt-Global architecture when fully fine-tuned achieves state-of-the-art performance on SuperGLUE across four different model sizes. Competitive adapter-tuning variants including Prompt-Tuning and HyperFormer++ can either match or slightly improve upon the multi-task learning (MTL) baseline on the SuperGLUE dataset. In contrast, HyperPrompt-Global outperforms the strong MTL baseline by a large margin on SuperGLUE score (78.9 vs 77.2 for T5 Base). Interestingly, such a performance gain continues all the way to model size as big as XXL (e.g. 91.3 vs 90.2) with only 0.14% additional parameters.

4.3. Tuning all vs Task-Conditioned Parameters

Recently, Karimi Mahabadi et al. (2021) show that only tuning adapters can be competitive against the full fine-tuning. However, the evaluation is conducted only on the GLUE with smaller models including T5 Small and Base.

In the experiments, we first compare tuning the full model vs. only task-conditioned parameters. Table 1 shows the com-

Tunable	Model	GLUE	SuperGLUE
All	MTL	88.3	85.9
All	HyperFormer++	88.8	86.4
All	HyperPrompt-Global	89.4	87
Task	HyperFormer++	87.3	80.5
Task	HyperPrompt-Global	87.5	81.5

Table 1. Comparison of fine-tuning all vs task-specific parameters using T5 Large. The average scores of GLUE and SuperGLUE are reported on T5 Large.

parison on the GLUE and SuperGLUE average scores using T5 large (for per-task performance, please refer to A.4). For GLUE, the observation is consistent with (Karimi Mahabadi et al., 2021), where task-specific only fine-tuning of HyperFormer++ and HyperPrompt-Global is comparable to the MTL baseline. However, on SuperGLUE, we observe a large gap: the average score drops by 5.5 and 5.9 for HyperPrompt-Global and HyperFormer++, respectively.

Therefore, these experiments show that only tuning the task-conditioned parameters is not enough to achieve competitive results as full model training for multi-task learning on high-difficulty tasks such as SuperGLUE. This is consistent with the results of Prompt-Tuning (Lester et al., 2021). Hence, the rest of the experiments are conducted with tuning all model parameters.

4.4. Computational Efficiency

Table 2 presents the computational efficiency of the Adapter/Prompt models. HyperPrompt-Global (together with HyperPrompt-Share) has the lowest # Ops since hyper-prompts are injected into self-attention and skip the standard FFN layers. In contrast, HyperFormer++ has $\sim$ 3x # Ops compared to other variants. Regarding training time, HyperPrompt-Share is fastest given that the local Hyper-Networks are shared across tasks. Vanilla Adapter and HyperPrompt-Global are comparable while HyperFormer++ and Prompt-Tuning take significant longer to do the full fine-tuning. This shows the computational efficiency of HyperPrompt for both training and inference.

Model	# Ops	Training Time (hours)
Vanilla Adapter	1.01×10^{13}	8.4
HyperFormer++	3.14×10^{13}	10.3
Prompt-Tuning	1.16×10^{13}	11.1
HyperPrompt-Sep	1.01×10^{13}	8.9
HyperPrompt-Share	9.8×10^{12}	**8.0**
HyperPrompt-Global	9.8×10^{12}	8.7

Table 2. The number of operations for a single forward pass and training time on T5 Base.

Model	#Params	GLUE	SuperGLUE
MTL	1.0x	85.5 (0.9)	77.2 (0.2)
Vanilla Adapter	1.06x	86.7 (0.3)	77.5 (0.1)
HyperFormer++	1.04x	86.5 (0.0)	78.2 (0.7)
Prompt-Tuning	1.0003x	84.8 (0.6)	77.3 (0.2)
HyperPrompt-Share	1.008x	86.4 (0.6)	78.2 (0.7)
HyperPrompt-Sep	1.06x	**86.8 (0.1)**	77.5 (0.1)
HyperPrompt-Global	1.04x	**86.8 (0.4)**	**78.9 (0.5)**

Table 3. GLUE and SuperGLUE average scores (standard deviations) over 3 runs of HyperPrompt against baselines on T5 Base.

4.5. Ablation Study

Table 3 presents the results on T5 Base and Table 4 presents the results on T5 Large (see more detailed results in A.4). HyperPrompt-Global outperforms all baselines in terms of the average score of GLUE and SuperGLUE.

HyperPrompt-Global vs. Prompt-Tuning. The original Prompt-Tuning (Lester et al., 2021) is for single-task fine-tuning. To be parameter-efficient, it only trains the prompts with the backbone frozen. To make a fair comparison, we modify Prompt-Tuning by (1) training both prompts and backbone, and (2) adding prompt to each task and co-train all tasks together. As shown in Table 3 and 4, HyperPrompt-Global outperforms Prompt-Tuning by 2.0 (0.6) and 1.6 (1.4) on GLUE and SuperGLUE using T5 Base (Large), respectively. HyperPrompt-Global improves upon Prompt-Tuning in two places: (1) Prompt-Tuning only adds prompts to the word embedding layer while HyperPrompt-Global adds hyper-prompts at every Transformer layer and hence is more expressive; and (2) Prompts of tasks are trained independently in Prompt-Tuning while HyperPrompt-Global enables a flexible information sharing via HyperNetworks.

HyperPrompt-Global vs. HyperFormer++. Our method is superior to the state-of-the-art baseline HyperFormer++ in the average score of GLUE and SuperGLUE for both Base and Large T5 model. For example, HyperPrompt-Global of T5 large achieves 87.0 on the SuperGLUE compared to 86.4 by HyperFormer++ (Table 4). Note that the main differ-

Model	#Params	GLUE	SuperGLUE
MTL	1.0x	88.3 (0.6)	85.9 (0.3)
Vanilla Adapter	1.06x	88.8 (0.2)	86.1 (0.5)
HyperFormer++	1.02x	88.8 (0.0)	86.4 (0.5)
Prompt-Tuning	1.0001x	88.8 (0.3)	85.6 (0.1)
HyperPrompt-Share	1.008x	89.3 (0.1)	86.8 (0.2)
HyperPrompt-Sep	1.06x	**89.4 (0.2)**	86.1 (0.3)
HyperPrompt-Global	1.02x	**89.4 (0.1)**	**87.0 (0.5)**

Table 4. GLUE and SuperGLUE average scores (standard deviations) over 3 runs of HyperPrompt against baselines on T5 Large.

ence between the two methods is that HyperPrompt-Global inserts the task-conditioned parameters as prompts into self-attention layers while HyperFormer++ insert adapters after each block. We believe task-conditioning in self-attention gives more expressive power than in the feed-forward network as done in adapters. Hyper-prompts that are prepended with the key and value participate in the attention interactions between different token positions, which helps the model to better capture the task-dependent semantics.

HyperPrompt-Global vs. MTL. Next, we observe that using HyperPrompt-Global can greatly improve the performance upon the vanilla Transformer model (referred to MTL): 1.7 (1.1) gain on SuperGLUE score for T5 Base (Large) with 4% (2%) additional paramters. In conclusion, the experiments show that HyperPrompt-Global is a parameter-efficient and effective task-conditioned parameterization of Transformers for multi-task learning.

HyperPrompt-Global vs. HyperPrompt-Share/Sep. Interestingly, HyperPrompt-Share is better than HyperPrompt-Sep on the SuperGLUE on both Base and Large models while the opposite is true for GLUE. Notice that all tasks share the same two projection networks in HyperPrompt-Share while each task has its own projection networks in HyperPrompt-Sep. More importantly, we observe that HyperPrompt-Global, where the projection networks are generated by the global HyperNetworks, always achieves the best performance on both GLUE and SuperGLUE. Hence, the experiments show that HyperPrompt-Global can adjust the degree of information sharing for better multi-task generalization, compared to HyperPrompt-Share/Sep.

4.6. Peeking into Hyper-Prompts

To shed light on how hyper-prompts help improve the multi-task generalization via task-conditioning, we peek into HyperPrompt-Global models by looking at the distribution of attention scores. We choose the GLUE task MRPC as an example. To avoid biasing on individual examples, we aggregate over 100 validation examples to compute the quantity of interest (see A.3 for details). First, we compute the attention mass on hyper-prompts for each encoder layer. Figure 3 (top) shows that the network has lower attention mass on hyper-prompts in the lower layers and gradually increases attention mass for higher layers. This phenomenon indicates that higher-levels of Transformer becomes more task-specialized while it is beneficial for the lower-levels to learn task-agnostic representation (Yosinski et al., 2014) by casting lower attention mass on hyper-prompts. Furthermore, we calculate the entropy of the attention scores on the tokens. For HyperPrompt-Global, we remove the hyper-prompts from the calculation and re-normalize the attention scores on the tokens to make a fair comparison with the MTL baseline. Figure 3 (bottom) shows a shift of entropy
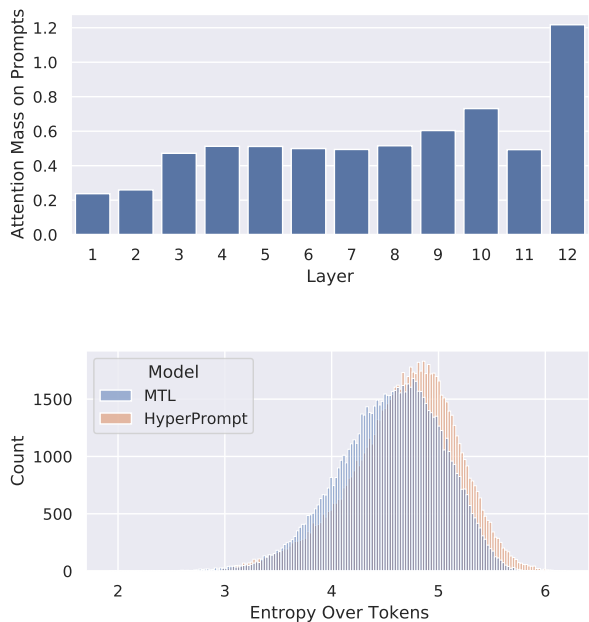

Figure 3. Visualization of attention mass and entropy distribution.

distribution towards higher values for HyperPrompt-Global. This signifies that injecting hyper-prompts encourages a more diverse attention distribution, which seems to be beneficial to model generalization.

4.7. Impact of Hyper-Prompt Length

HyperPrompt prepends l trainable hyper-prompts to the keys and values of self-attention layer at every Transformer layer. In Figure 4, we present the results of tuning the prompt length l on GLUE using T5 Base as the example for HyperPrompt-Global (similar patterns are observed on T5 Large and SuperGLUE). We first add hyper-prompts on the decoder and search the best l and then search the best l for the encoder with the fixed best decoder hyper-prompt length. As shown in Figure 4(a), $l = 6$ is the best for the decoder. As shown in Figure 4(b), HyperPrompt-Global
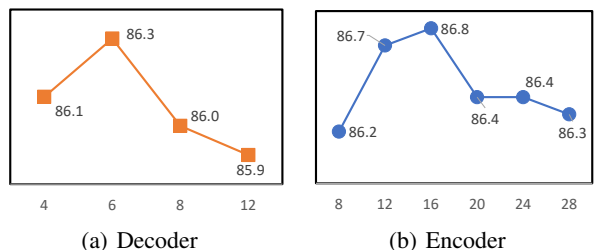

(a) Decoder   (b) Encoder

Figure 4. Impact of hyper-prompt length in HyperPrompt-Global (GLUE score on T5 Base).

achieves the best result of 86.8 when $l = 16$ on the encoder with $l = 6$ fixed for the decoder. The experiments show that hyper-prompts with length $l \sim \mathcal{O}(10)$ are good enough to achieve superior performance. Note that the original sequence length is 512 on the encoder and 32 on the decoder. Therefore, HyperPrompt does not substantially increase the time complexity of self-attention layers in practice.

4.8. Encoder vs Decoder

To understand the effect of adding task-conditioned parameters to different parts of the network, we present the results of HyperPrompt-Global and HyperFormer++ with adding hyper-prompts/adapters to: (1) encoder-only, (2) decoder-only, and (3) both encoder-decoder. As shown in Table 5, we observe adding task-conditioned parameters to encoder (encoder-only) performs better than decoder-only on GLUE. However, the opposite is true for SuperGLUE, where encoder-only is substantially worse than decoder-only. This potentially could be a trainability issue when prompts are inserted into encoders, i.e. a different learning rate might be required to learn the prompt parameters from scratch. We leave this investigation as a future work. Based on this experiment, we add task-conditioned parameters to the decoder for SuperGLUE in our experiments.

Model	#Params	GLUE	SuperGLUE
MTL	1.0x	85.5	77.2
HyperFormer++-Encoder	1.02x	85.9	74.4
HyperFormer++-Decoder	1.02x	85.7	78.2
HyperFormer++-Enc-Dec	1.04x	86.5	74.8
HyperPrompt-Encoder	1.02x	86.6	76.5
HyperPrompt-Decoder	1.02x	86.3	**78.9**
HyperPrompt-Enc-Dec	1.04x	**86.8**	78.7

Table 5. Ablation of inserting hyper-prompts or adapters into Encoder/Decoder/Enc-Dec (T5 Base).

5. Related Work

Prompt-Tuning. Prompt tuning is becoming a new paradigm for adapting pre-trained general-purpose language models to downstream tasks, as a lightweight alternative to the popular fine-tuning approach. Here, we use the term Prompt-Tuning to cover a family of methods following the prompting idea in GPT-3 (Brown et al., 2020). To avoid manually design the prompts, recent efforts have focused on search for discrete prommpting words automatically (Shin et al., 2020). On the other hand, soft prompts (Li & Liang, 2021; Hambardzumyan et al., 2021; Lester et al., 2021; Liu et al., 2021) in the form of continuous vectors are introduced to simplify the process and have shown competitive results (Lester et al., 2021; Liu et al., 2021; Li & Liang, 2021). In particular, Lester et al. (2021) show that soft

prompts can become competitive against full fine-tuning for a 11B parameters model, but with a big performance gap for moderate-size models. In our work, we close this gap in the full fine-tuning setting and demonstrate that Hyper-Prompt can outperform strong baselines across all model sizes studied.

Adapter-Tuning. Adapter tuning (Houlsby et al., 2019a;b; Karimi Mahabadi et al., 2021) is an alternative approach for parameter-efficient lightweight tuning of pre-trained langauge models. Task-specific adapter layers (Houlsby et al., 2019a) are inserted into the Transformer block for fine-tuning while the rest of the backbone model is frozen. By adding only a few percent of additional parameters, Karimi Mahabadi et al. (2021) show that competitive performance can be obtained on NLU benchmarks such as GLUE (Wang et al., 2018). However, one limitation from the existing work is the evaluation of NLU on GLUE dataset, which is known to be no longer suitable for measuring the progress of language understanding (Wang et al., 2019). In our work, we evaluate HyperPrompt on SuperGLUE in addition to GLUE, and show that indeed higher-difficulty tasks such as SuperGLUE requires full-tuning of the model beyond adapter tuning, to be competitive against state-of-the-art baselines. We also demonstrate that it is advantageous to inject prompts into self-attention than adding adapters.

Multi-task Natural Language Understanding. Multi-task learning is an important and challenge research direction in both full fine-tuning and prompt-tuning paradigms because of the competing needs of training and serving a single model while achieving Pareto efficiency in all tasks.

The T5 model (Raffel et al., 2019) renders all NLP tasks as a Text-to-Text problem. However, the best results are obtained by task-specific fine-tuning. MTDNN (multi-task deep neural network) (Liu et al., 2019a) shares parameters between several NLP tasks, and achieves strong performance on the GLUE benchmark. Aghajanyan et al. (2021) use around 50 tasks to boost the multi-task learning performance. Aribandi et al. (2021) builds an extremely diverse set of 107 NLP tasks for extreme multi-task scaling and demonstrate superior performances on a wide range of benchmarks. Recently, Wei et al. (2021); Sanh et al. (2021) also illustrated how a multi-task learning stage can greatly improve the zero-shot prompting performance of large language models.

6. Conclusion

We propose a novel architecture for prompt-based task-conditioning of self-attention in Transformers. The hyper-prompts are generated by a HyperNetwork to enable flexible information sharing among tasks while remain efficient in parameters and computation. HyperPrompt allows the network to learn task-specific feature maps where the

hyper-prompts serve as task global memories, encouraging a more diverse distribution of attention. Extensive experiments show that HyperPrompt can achieve superior performances over strong T5 multi-task learning baselines and parameter-efficient models including Prompt-Tuning and HyperFormer++ on GLUE and SuperGLUE benchmarks.

References

Aghajanyan, A., Gupta, A., Shrivastava, A., Chen, X., Zettlemoyer, L., and Gupta, S. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5799–5811, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.468. URL https://aclanthology.org/2021.emnlp-main.468.

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H., Tran, V. Q., Bahri, D., Ni, J., Gupta, J., Hui, K., Ruder, S., and Metzler, D. Ext5: Towards extreme multi-task scaling for transfer learning, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Dehghani, M., Arnab, A., Beyer, L., Vaswani, A., and Tay, Y. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe11x.

Hambardzumyan, K., Khachatrian, H., and May, J. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4921–4933, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.381. URL https://aclanthology.org/2021.acl-long.381.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019a. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019b.

Karimi Mahabadi, R., Ruder, S., Dehghani, M., and Henderson, J. 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)*, August 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mo-hamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. 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*, pp. 7871–7880, 2020.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL https://aclanthology.org/P19-1441.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, 2019b.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. Gpt understands, too, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Biderman, S., Gao, L., Bers, T., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization, 2021.

Shazeer, N., Cheng, Y., Parmar, N., Tran, D., Vaswani, A., Koanantakool, P., Hawkins, P., Lee, H., Hong, M., Young, C., et al. Mesh-tensorflow: Deep learning for supercomputers. *arXiv preprint arXiv:1811.02084*, 2018.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main. 346. URL https://aclanthology.org/2020.emnlp-main.346.

Sukhbaatar, S., Grave, E., Lample, G., Jegou, H., and Joulin, A. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Tay, Y., Zhao, Z., Bahri, D., Metzler, D., and Juan, D.-C. Hypergrid: Efficient multi-task transformers with grid-wise decomposable hyper projections. *arXiv preprint arXiv:2007.05891*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

Wang, Y., Zhao, Z., Dai, B., Fifty, C., Lin, D., Hong, L., and Chi, E. H. Small towers make big differences, 2020.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners, 2021.

Wu, S., Zhang, H. R., and Ré, C. Understanding and improving information transfer in multi-task learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylzhkBtDB.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Zaken, E. B., Ravfogel, S., and Goldberg, Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

A. Appendix

This section covers the parameter count of HyperPrompt, the experimental details, the calculation of attention mass and entropy, and per-task performance of GLUE and SuperGLUE.

A.1. Parameter Count of HyperPrompt-Global (§3.4)

Since the encoder and the decoder of Transformers have approximately the same capacity, the calculation considers only the decoder-side for simplicity. First, we have global task prompts $P_\tau \in \mathbb{R}^{l \times d}$ for the τ-th task, which contains dlT parameters for T tasks. The global HyperNetworks contain four weight matrices $W^{D_k} \in \mathbb{R}^{(d \times b) \times t}$, $W^{D_v} \in \mathbb{R}^{(d \times b) \times t}$, $W^{U_k} \in \mathbb{R}^{(b \times h \times d_h) \times t}$ and $W^{U_v} \in \mathbb{R}^{(b \times h \times d_h) \times t}$, which result in $4(bdt)$ parameters (we let $d = h \times d_h$). To obtain layer-aware task embedding, HyperPrompt-Global learns task embedding $k_\tau \in \mathbb{R}^{t'}$ for the τ task and layer embedding $z_m \in \mathbb{R}^{t'}$ for the m-th Transformer block, which in total results in $Tt' + Mt'$ parameters. Besides, a task projection network h_t is applied to fuse the task embedding and the layer embedding into the final layer-aware task embedding $I_\tau^m \in \mathbb{R}^t$. h_t is a two-layer feed-forward networks and contains $(2t' + t)e$ parameters, where e is the hidden dimension for h_t.

A.2. Experimental Details (§4.1)

Our models were implemented using Mesh Tensorflow[2] (Shazeer et al., 2018) with the T5 library[3] (Raffel et al., 2019). Following Raffel et al. (2019), all data are preprocessed as into a "sequence-to-sequence" format. The length of the sequence is 512 at the encoder and 32 at the decoder. For all experiments, we train models 300K steps with a batch size of 128 and each batch is a mixture which samples each task proportionately to the number of examples in the dataset. Learning rate is a constant of 1e-3 with Adam optimizer (Kingma & Ba, 2014).

For hyper-parameters tuning, the length of prompt l is selected from $\{12, 16, 20, 20, 24\}$ at the encoder and $\{2, 4, 6, 8, 10, 12, 14, 16\}$ at the decoder. The bottleneck dimension b in the transform matrices is set to d/r, where d is the model dimension of the T5 models and r is a reduction factor and selected from $\{16, 32, 64\}$. The dimension t of the layer-aware task embedding is selected from $\{32, 64, 128\}$. For a fair comparison, the hyper-parameters of baseline methods are set to have approximately the same numbers of parameters as HyperPrompt-Global with the exception that Prompt-Tuning and HyperPrompt-Share are extremely parameter-efficient with significantly fewer parameters.

A.3. Attention Mass and Entropy calculation (§4.6)

To calculate the attention mass over hyper-prompts per layer, we averaged the hyper-prompt attention softmax scores across 100 validation examples and each attention head in a layer, and summed across each query attending to the hyper-prompts. In other words, we aggregated the amount of attention given to hyper-prompts by queries. To calculate the attention entropy over tokens (other than hyper-prompts), we calculated the entropy of the attention distributions (averaged across attention heads) for 100 validation examples. This results in $\sum_{n=1}^{100} \sum_{L=1}^{12} |X_n|$ entropies calculated and visualized in Figure 3 (bottom). For the HyperPrompt model, this involved re-normalizing the softmax distribution after removing hyper-prompts, as we wanted to understand how the original tokens are attended to.

A.4. Per-Task Performance of GLUE and SuperGLUE

Table 6 and 7 below show the comparison of fine-tuning the entire model against task-specific parameters only on GLUE and SuperGLUE datasets. Table 8 and 9 show the detailed results of full-tuning of HyperPrompt against baselines on T5 Base. Table 10 and 11 show the detailed results of full-tuning of HyperPrompt against baselines on T5 Large.

---

[2]https://github.com/tensorflow/mesh
[3]https://github.com/google-research/text-to-text-transfer-Transformer

Tunable Parameters	Model	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
All	MTL	59.4	96.6	93.3/90.7	90.6/90.4	89.8/92.3	90.8/90.8	95.2	90.8	88.3
All	HyperFormer++-T5.1.1$_{\text{LARGE}}$	63.3	96.6	93.2/90.7	92.1/91.9	89.7/92.3	90.5/90.7	95.1	89.9	88.8
All	HyperPrompt-T5.1.1$_{\text{LARGE}}$	64.6	96.7	94.0/91.8	91.3/91.4	90.0/92.4	90.8/91.0	95.4	91.9	89.4
Task-Specific	HyperFormer++-T5.1.1$_{\text{LARGE}}$	58.9	95.7	92.7/90.0	91.6/91.5	87.7/90.7	89.8/90.0	94.5	87.0	87.3
Task-Specific	HyperPrompt-T5.1.1$_{\text{LARGE}}$	57.5	96.7	93.6/91.2	91.9/92.0	87.0/90.1	90.3/90.6	95.0	87.7	87.5

Table 6. Comparison of fine-tuning all vs task-specific parameters on GLUE.

Tunable Parameters	Model	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AVG
All	MTL	88.5	95.8/98.2	87.0	85.5/56.3	89.2/88.6	91.7	74.0	89.4	85.9
All	HyperFormer++-T5.1.1$_{\text{LARGE}}$	88.9	98.7/98.2	86.7	85.4/56.7	89.4/88.8	92.1	74.5	90.7	86.4
All	HyperPrompt-T5.1.1$_{\text{LARGE}}$	88.7	99.1/98.8	91.0	85.0/55.6	89.8/89.1	91.3	74.2	92.0	87.0
Task-Specific	HyperFormer++-T5.1.1$_{\text{LARGE}}$	85.2	90.9/94.6	76.7	81.5/48.8	87.2/86.4	87.7	67.8	82.1	80.5
Task-Specific	HyperPrompt-T5.1.1$_{\text{LARGE}}$	85.2	95.2/95.5	75.5	82.9/52.9	89.1/88.3	85.7	71.1	82.2	81.5

Table 7. Comparison of fine-tuning all vs task-specific parameters on SuperGLUE.

Model	#Params	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
MTL	1.0x	49.8	94.6	92.5/89.8	90.7/90.5	89.2/91.9	**88.8/88.5**	93.3	85.0	85.5
Vanilla Adapter	1.06x	**60.0**	**95.4**	92.7/89.8	90.2/90.2	**89.3**/91.9	88.5/88.1	**93.5**	84.4	86.7
HyperFormer++	1.04x	56.9	94.8	92.9/90.1	**91.1/90.9**	88.9/91.7	88.7/88.3	93.4	85.6	86.5
Prompt-Tuning	1.0003x	48.0	95.0	92.2/89.0	90.3/90.2	89.0/91.7	**88.8/88.5**	93.2	82.9	84.8
HyperPrompt-Share (ours)	1.008x	56.2	94.7	93.0/90.4	90.6/90.4	89.2/91.9	88.7/88.4	93.4	85.2	86.4
HyperPrompt-Sep (ours)	1.06x	57.2	94.6	**93.8/91.4**	91.0/90.8	89.2/91.9	88.5/88.4	93.4	86.6	**86.8**
HyperPrompt-Global (ours)	1.04x	57.0	95.2	93.4/90.9	90.4/90.2	89.2/**92.0**	88.7/**88.5**	93.4	**87.1**	**86.8**

Table 8. Comparison of HyperPrompt with baselines on GLUE using T5 Base.

Model	#Params	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WIC	WSC	AVG
MTL	1.0x	82.6	93.4/93.5	65.7	76.7/39.7	80.9/80.2	85.6	70.5	81.4	77.2
Vanilla Adapter	1.03x	**83.5**	93.4/94.6	65.3	77.6/**42.7**	81.0/80.2	**88.2**	71.0	76.9	77.5
HyperFormer++	1.02x	**83.5**	96.2/97.0	66.3	**77.8**/41.9	81.2/80.4	87.4	71.0	80.1	78.2
Prompt-Tuning	1.0003x	82.5	94.0/95.8	68.0	76.9/40.2	80.9/80.2	84.1	69.3	80.8	77.3
HyperPrompt-Share (ours)	1.004x	83.1	95.7/95.2	67.7	77.3/41.3	**81.9**/81.0	87.4	70.4	80.8	78.2
HyperPrompt-Sep (ours)	1.03x	83.3	**97.8/97.0**	61.7	77.6/42.3	81.5/80.6	86.8	**71.4**	78.2	77.5
HyperPrompt-Global (ours)	1.02x	83.3	96.6/96.4	**69.7**	77.5/41.0	**81.7/80.9**	86.8	70.5	**83.7**	**78.9**

Table 9. Comparison of HyperPrompt with baselines on SuperGLUE using T5 Base.

Model	#Params	CoLA	SST-2	MRPC	SST-B	QQP	MNLI	QNLI	RTE	AVG
MTL	1.0x	59.4	96.6	93.3/90.7	90.6/90.4	89.8/92.3	90.8/90.8	95.2	90.8	88.3
Vanilla Adapter	1.06x	63.8	96.5	93.7/91.3	92.0/**91.9**	**90.0/92.5**	90.6/90.5	94.9	88.7	88.8
HyperFormer++	1.02x	63.3	96.6	93.2/90.7	**92.1/91.9**	89.7/92.3	90.5/90.7	95.1	89.9	88.8
Prompt-Tuning	1.0001x	62.5	**96.7**	93.4/91.0	91.3/91.0	**90.0**/92.4	**90.9**/91.0	**95.4**	89.9	88.8
HyperPrompt-Share (ours)	1.008x	**65.0**	**96.7**	93.8/91.6	91.1/90.8	**90.0**/92.4	90.8/**91.1**	95.3	91.3	89.3
HyperPrompt-Sep (ours)	1.06x	63.9	96.6	**94.6/92.6**	92.0/91.7	**90.0**/92.4	**90.9**/91.0	95.2	91.6	**89.4**
HyperPrompt-Global (ours)	1.02x	64.6	**96.7**	94.0/91.8	91.3/91.4	**90.0**/92.4	90.8/91.0	**95.4**	91.9	**89.4**

Table 10. Comparison of HyperPrompt with baselines on GLUE using T5 Large.

Model	#Params	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WIC	WSC	AVG
MTL	1.0x	88.5	95.8/98.2	87.0	**85.5**/56.3	89.2/88.6	91.7	74.0	89.4	85.9
Vanilla Adapter	1.03x	88.8	98.3/**98.8**	86.0	85.3/56.0	89.3/88.7	91.2	73.6	91.3	86.1
HyperFormer++	1.01x	**88.9**	98.7/98.2	86.7	85.4/**56.7**	89.4/88.8	**92.1**	**74.5**	90.7	86.4
Prompt-Tuning	1.0001x	88.5	97.6/**98.8**	85.0	84.9/55.2	89.0/88.4	91.5	72.8	90.1	85.6
HyperPrompt-Share (ours)	1.004x	88.5	98.7/98.2	88.0	85.2/55.8	89.7/89.1	91.8	74.1	**93.9**	86.8
HyperPrompt-Sep (ours)	1.03x	88.6	97.6/**98.8**	87.7	85.2/56.4	89.7/89.1	91.6	73.5	89.4	86.1
HyperPrompt-Global (ours)	1.01x	88.7	**99.1/98.8**	**91.0**	85.0/55.6	**89.8/89.1**	91.3	74.2	92.0	**87.0**

Table 11. Comparison of HyperPrompt with baselines on SuperGLUE using T5 Large.