HyperPELT: Unified Parameter-Efficient Language Model Tuning for Both Language and Vision-and-Language Tasks

Zhengkun Zhang[1*†‡], Wenya Guo[1†], Xiaojun Meng[2†], Yasheng Wang[2], Yadao Wang[2]
Xin Jiang[2], Qun Liu[2], Zhenglu Yang[1‡]
[1]TKLNDST, CS, Nankai University, China, [2]Noah's Ark Lab, Huawei Technologies,
zhangzk2017@mail.nankai.edu.cn,wenyaguo@nankai.edu.cn
{xiaojun.meng, wangyasheng, wangyadao, Jiang.Xin, qun.liu}@huawei.com,
yangzl@nankai.edu.cn

Abstract

With the scale and capacity of pretrained models growing rapidly, parameter-efficient language model tuning has emerged as a popular paradigm for solving various NLP and Vision-and-Language (V&L) tasks. In this paper, we design a unified parameter-efficient multitask learning framework that works effectively on both NLP and V&L tasks. In particular, we use a shared hypernetwork that takes trainable hyper-embeddings and visual modality as input, and outputs weights for different modules in a pretrained language model, such as the parameters inserted into multi-head attention blocks (*i.e.,* prefix-tuning) and feed-forward blocks (*i.e.,* adapter-tuning.). Our proposed framework adds fewer trainable parameters in multi-task learning while achieving superior performances and transfer ability compared to state-of-the-art methods. Empirical results on the GLUE benchmark and multiple V&L tasks confirm the effectiveness of our framework.

1 Introduction

Pretraining and fine-tuning are now the prevalent paradigm in natural language processing, yielding state-of-the-art performances on a variety of tasks (Devlin et al., 2019). With pre-trained language models (PLMs) growing rapidly in size, it becomes increasingly infeasible to perform conventional fine-tuning on the entire model parameters. There has recently been one line of research on **P**arameter-**E**fficient **L**anguage model **T**uning (**PELT**)(Houlsby et al., 2019; Li and Liang, 2021; He et al., 2021; Mao et al., 2022). They only update a set of extra trainable task-specific parameters that are newly introduced to PLMs. Although the number of new parameters is much fewer than the original PLM, training these parameters per single task is still costly, especially when targeting a number of tasks, i.e., multi-tasking scenario.

Therefore, we are motivated to start with a unified parameter-efficient language model tuning framework (He et al., 2021) and explore on a shared hypernetwork (von Oswald et al., 2020; Mahabadi et al., 2021) that is able to take multi-task information as input, and generate weights for tuning different task-specific modules of PLMs, such as the parameters inserted into multi-head attention blocks (*i.e.,* prefix-tuning) and feed-forward blocks (*i.e.,* adapter-tuning.). We name it **HyperPELT**. Besides, we propose a novel perspective of adopting parameter-efficient multimodal fusion for PLMs via the hypernetwork. Thus we explore to use an additional separate hypernetwork handling visual input and generating visual-specific weights for multiple modules of PLMs.

Empirical results on 8 tasks of GLUE benchmark show that HyperPELT achieves superior performances (87.09 vs. 86.53) with a tenth of the parameters (0.24% vs. 2.96%) when compared to state-of-the-art alternatives. Study on the few-shot transfer learning indicates that HyperPELT is more stable and efficient than alternatives. It confirms the effectiveness of our unified parameter-efficient multitask learning framework. What's more, we evaluate our framework on V&L multi-tasks (4 tasks). Results show the promising performance of our novel fusion method on extending V&L ability on top of PLMs via hypernetworks.

In summary, we make the following contributions: (1) propose a unified parameter-efficient multitask learning framework that is able to take multi-task and multi-modality information as input, and generate weights for tuning different task-specific modules of PLMs; (2) present a novel perspective of using hypernetworks to achieve the parameter-efficient multimodal fusion on top of PLMs; (3) design various experiments to compre-

[*]Work is done at the internship of Noah's Ark Lab, Huawei Technologies.
[†]Equal Contribution.
[‡]Corresponding authors.

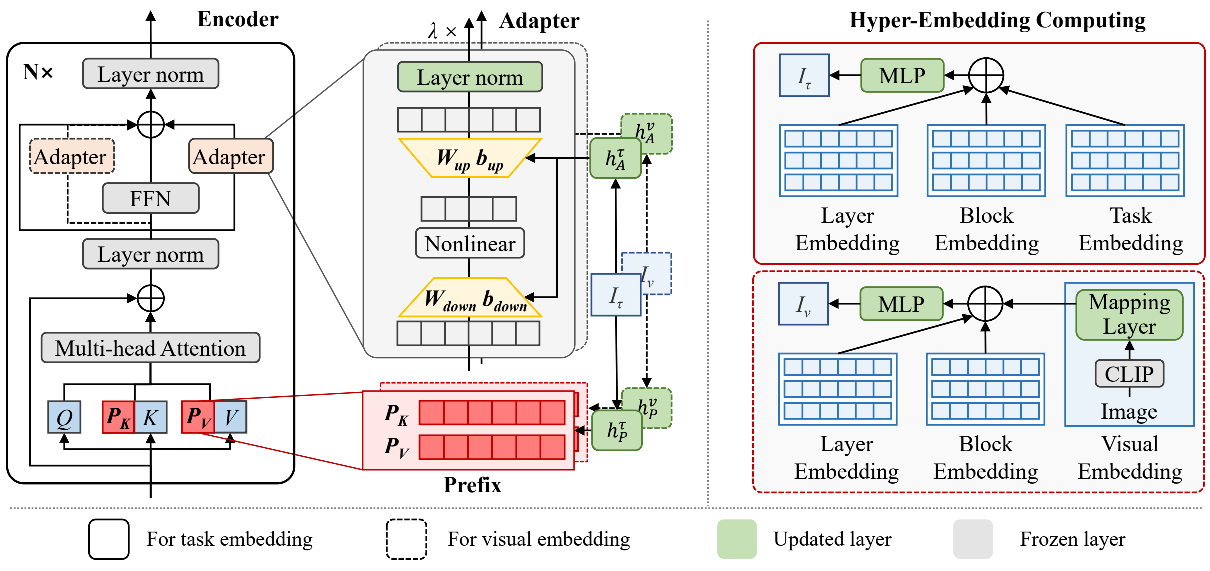

Figure 1: The model structure of the proposed unified pure language and V&L multi-task framework (left), and illustration of computing the hyper-embedding (right). We use green color to fill the trainable layers and grey color for the frozen ones. And the dashed parts denote the modules for processing visual modality.

hensively demonstrate the effectiveness of our proposed framework in multi-task learning and few-shot domain transfer scenarios.

2 Related Work

Existing research has explored a large amount of methods on parameter-efficient tuning, such as the widely used adapter-tuning (Houlsby et al., 2019), prefix-tuning (Li and Liang, 2021) and the mixed methods (He et al., 2021; Mao et al., 2022). However, it is time & space-consuming to deal with a set of tasks in multi-task learning if we simply update and save separate replicas of model parameters per single task. In this work, we explore a hypernetwork-based multi-task learning framework to generate weights for different PELT modules.

Besides, there has been a series of recent work (Cho et al., 2021; Tsimpoukelli et al., 2021; Sung et al., 2021; Alayrac et al., 2022) to equip a language model with the ability of handling visual input with a small number of trainable modules and parameters. Different from existing work, we propose a novel perspective of multimodal fusion via extending the proposed parameter-efficient multi-task learning framework. We further review recent research on parameter-efficient tuning for pure language and V&L tasks, as well as the corresponding work for multi-task learning in Appendix A.

3 Proposed Method

We target a general multi-task learning problem, which is formulated in Appendix B. In this section, we describe the hyper-embedding I for hypernetworks to generate weights $\Delta\theta$ and which modules of PLMs to insert these weights to achieve PELT. In our methods, the hyper-embedding I consists of two: task-specific hyper-embedding I_τ, and visual-specific hyper-embedding I_v. We will mostly introduce the hyper-embedding I_τ, and I_v is used in a similar parallel manner. A simple linear projection layer is employed as the hypernetwork, for example, $h_P^\tau(.)$ and $h_P^v(.)$ are used for prefix-tuning, while $h_A^\tau(.)$ and $h_A^v(.)$ are for adapter-tuning as shown in Figure 1. The hypernetwork takes the hyper-embedding I as input, and outputs weights for multiple modules of PLMs.

3.1 Hyper-Embedding for PELT

Considering a flexible parameterization of task-specific parameters for L layers of transformer, we introduce a set of layer id embeddings $\mathcal{I} = \{l_i\}_{i=1}^{L}$, and block type embeddings $\mathcal{B} = \{b_j\}_{j=1}^{5}$, which specify the position where the parameters $\Delta\theta$ are inserted to. Then, we compute a hyper-embedding $I_\tau \in \mathbb{R}^{d_I}$ for each individual task via a task projector network, which is a multi-layer perceptron consisting of two feed-forward layers and a ReLU nonlinearity: $I_\tau = \mathrm{MLP}([z_\tau, l_i, b_j])$. Thus, it learns a suitable compressed hyper-embedding from a concatenation of task embeddings $z_\tau \in \mathbb{R}^{d_\tau}$, layer id embeddings $l_i \in \mathbb{R}^{d_\tau}$, and block type embeddings $b_j \in \mathbb{R}^{d_\tau}$. In this way, the hypernetwork is able to produce distinct weights for tuning each task, and each transformer block at each layer.

3.2 HyperPELT: Incorporate with Prefix-tuning and Adapter-tuning

To further capture knowledge across tasks and transfer to others, we follow the unified parameter-efficient framework (He et al., 2021), and input the hyper-embedding to a hypernetwork for generating the weights in adapters as well as prefix vectors. We extend the dimension for different embeddings to match the prefix length N, *i.e.,* $z \in \mathbb{R}^{N\times d_\tau}$, $l_i \in \mathbb{R}^{N\times d_\tau}$, $b_j \in \mathbb{R}^{N\times d_\tau}$, and then compute the hyper-embedding $I_\tau \in \mathbb{R}^{N\times d_I}$. We finally employ a hypernetwork $h_P^\tau(.)$ with trainable parameters $\theta_{h_P^\tau}$, to project I_τ to prefix vectors $P_\tau \in \mathbb{R}^{N\times d}$: $P_\tau = h_P^\tau(\theta_{h_P^\tau}, I_\tau)$.

Besides, as depicted in Figure 1, we introduce a hypernetwork-based adapter layer with a trainable scaled parameter λ, which is inserted parallelly with feed-forward blocks. We generate adapter weights $(W_{\text{up}}^\tau, W_{\text{down}}^\tau)$ through a hypernetwork $h_A^\tau(.)$: $(W_{\text{up}}^\tau, W_{\text{down}}^\tau) := h_A^\tau(\theta_{h_A^\tau}, I_\tau)$, where $W_{\text{down}}^\tau \in \mathbb{R}^{d_{\text{mid}}\times d}$ and $W_{\text{up}}^\tau \in \mathbb{R}^{d\times d_{\text{mid}}}$.

3.3 VL-HyperPELT: Incorporate with Visual Modality

As illustrated in Fig. 1, we use CLIP (Radford et al., 2021) with a trainable visual mapping layer, which projects the visual representation to the identical dimension of task embedding, *i.e.,* $z_v \in \mathbb{R}^{N\times d_v}, d_v = d_\tau$. Then we feed this visual representation z_v to a visual projector network. In this way, we learn the visual hyper-embedding $I_v \in \mathbb{R}^{d_I}$. Finally, taking the visual-specific hyper-embeddings as input, we use visual-specific hypernetworks to generate visual-specific parameters to different modules in PLMs. Similar to the Section 3.1 & 3.2, the incorporation of visual-specific parameters to PLMs are the same as task-specific ones, *e.g.,* used as prefix vectors via a prefix hypernetwork $h_P^v(.)$ and adapter weights via an adapter hypernetwork $h_A^v(.)$. We name it *VL-HyperPELT.*

4 Results and Analysis

We conduct a series of experiments to verify the effectiveness of our proposed framework compared to existing ones.

4.1 Implementation Details

Our models are built on *T5*

Our models are built on $T5_{BASE}$ (Raffel et al., 2020) [1], which contains 12 layers and 222M parameters, and use the tokenizer of *T5* to tokenize text inputs. We set $N = 49$, $d = d_\tau = 768$, $d_I = 64$ for all the experiments. Following the training strategies from Raffel et al. (2020), we fine-tune all models with a constant learning rate of 0.001, use $2^{18} = 262144$ steps in all experiments with batch size of 128 and sample tasks via the conventional temperature-based sampler with temperature $T = 2$, i.e., sample corresponding task proportional to $p_\tau^{1/T}$, where $p_\tau = \frac{N_\tau}{\sum_{i=1}^{T} N_\tau}$ and N_τ is the number of training samples for the τ-th task. We did not experiment with other complex sampling strategies or tuning of T. For the experiments under multi-task training settings, we save a checkpoint every 1000 steps and report results on a single checkpoint with the highest average validation performance across all tasks.

In terms of the vision-and-language scenarios, we convert V&L tasks to the text generation format as Cho et al. (2021). We use *ResNet101* as our vision encoder, and initialize it with weights from pretrained CLIP (Radford et al., 2021). Input images are resized to 224×224 for memory efficiency. We extract the 7×7 grid features produced by the last convolutional layer. The percentage of updated parameters is also reported as one metric for approach efficiency, and we do not take visual encoder into account since it is frozen in our experiment.

4.2 Datasets

Our framework is evaluated on the GLUE benchmark (Wang et al., 2019b) in terms of natural language understanding. This benchmark covers multiple tasks of paraphrase detection (MRPC, QQP), sentiment classification (SST-2), natural language inference (MNLI, RTE, QNLI), and linguistic acceptability (CoLA). The original test sets are not publicly available, and following Zhang et al. (2021), for datasets fewer than 10K samples (RTE, MRPC, STS-B, CoLA), we split the original validation set into two halves, one for validation and the other for testing. For other datasets, we randomly split 1K samples from the training set for validation and test on the original validation set.

In addition, we evaluate the few-shot transfer performance on four tasks and datasets: 1) the

[1]https://huggingface.co/t5-base

Methods	#Total params	#Trained params/task	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg
Single-Task Training											
T5$_{\text{BASE}}$ †	8.0×	100%	54.85	92.19	88.18/91.61	91.46/88.61	89.55/89.41	86.49	91.60	67.39	84.67
Adapters †	1+8×0.01	0.87%	59.49	93.46	88.18/91.55	90.94/88.01	87.44/87.18	86.38	92.26	68.84	84.88
Multi-Task Training											
T5$_{\text{BASE}}$ †	1.0×	12.5%	54.88	92.54	90.15/93.01	91.13/88.07	88.84/88.53	85.66	92.04	75.36	85.47
Adapters †	1.07×	0.82%	61.53	93.00	90.15/92.91	90.47/87.26	89.86/89.44	86.09	93.17	70.29	85.83
Prefix-tuning ♣	1.14×	1.72%	56.67	93.92	89.42/92.57	90.59/87.37	89.49/89.34	85.23	93.17	79.17	86.09
MAMAdapters ♣	1.15×	2.96%	56.53	93.58	91.35/93.96	90.58/87.53	88.89/88.76	85.98	92.77	81.94	86.53
HYPERFORMER++ †	1.02×	0.29%	63.73	94.03	89.66/92.63	90.28/87.20	90.00/89.66	85.74	93.02	75.36	86.48
HyperPELT	1.02×	0.24%	65.96	93.23	89.42/92.31	90.48/87.54	89.15/89.07	85.35	92.79	82.64	**87.09**

Table 1: Performance of all models on the GLUE tasks. For each method, we report the total number of parameters across all tasks and the number of parameters that are trained for each task as a multiple and proportion respectively of the baseline single-task *T5* model. †: Results from the implementation of Mahabadi et al. (2021), ♣: We implement the methods of Li and Liang (2021) and He et al. (2021) on top of *T5*.

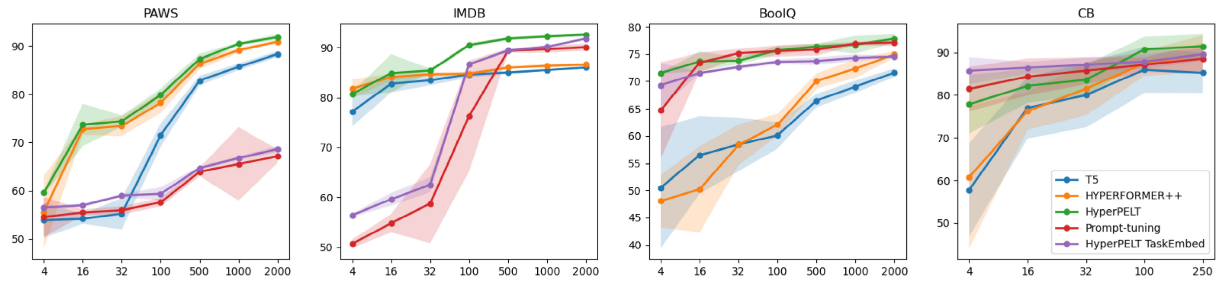

Figure 2: Few-shot domain transfer results of five different tasks averaged across 5 seeds. We compute accuracy for all tasks and datasets. HyperPELT and HyperPELT TaskEmbed are respectively fine-tuning hypernetworks with all hyper-embeddings and only task embedding in the few-shot learning.

natural language inference (NLI) datasets CB and 2) the question answering (QA) dataset BoolQ from SuperGLUE (Wang et al., 2019a); 3) the sentiment analysis datasets IMDB (Maas et al., 2011); and 4) the paraphrase detection dataset PAWS (Zhang et al., 2019). For CB and BoolQ, since the test set is not available, we split the validation set into two halves, one for validation and the other for testing. For IMDB, since the validation set is not available, we similarly split the test set to form validation. For PAWS, we report on the original test set.

To evaluate our framework on V&L tasks, we experiment on four datasets COCO (Lin et al., 2014), VQA (Goyal et al., 2017), VG-QA (Krishna et al., 2017) and GQA (Hudson and Manning, 2019). Following Cho et al. (2021), we use VQA Karpathy split, which splits the VQA dataset into 605,102 / 26,729 / 26,280 image and question pairs separately as the train/validation/test set to evaluate VQA tasks in a generative manner. We further evaluate our framework on two datasets for V&L few-shot transfer learning: OKVQA (Marino et al., 2019); SNLI-VE (Xie et al., 2018).

4.3 Results on the GLUE Benchmark

We conduct experiments on GLUE for both single- and multi-task settings, as shown in Table 1. Compared to the single-task *Adapters* that finetunes all newly introduced parameters in adapters, our method yields a significant improvement by 2.21% with much fewer trainable parameters. It illustrates the effectiveness of our proposed multi-task training framework. The comparison to *MAMAdapter* shows that using hypernetwork to tune each transformer module and thus learn the shared knowledge across multitasks, leads to an improvement in task performance (86.53 vs. 87.09) while training fewer parameters (2.96% vs. 0.24%). Overall, our *HyperPELT* obtains the best performance with less trainable parameters.

4.4 Few-shot Domain Transfer

We use the above models trained on GLUE as reported in Table 1, and evaluate them on the test set of four different tasks, i.e., PAWS, IMDB, BoolQ, and CB, after being few-shot finetuned on each target training data, as shown in Figure 2. For the

Methods	Trained Params (%)	VQAv2 test-std	VQA Karpathy test in-domain	out-domain	overall	GQA test-dev	COCO Caption B@4	M	C	S
Single-Task Training										
VL-T5 †	100%	70.3	71.4	13.1	67.9	60.0	34.6	28.8	116.1	21.9
Multi-Task Training										
VL-T5 †	100%	-	-	-	67.2	58.9	-	-	110.8	-
CLIP-T5 †	100%	-	-	-	67.3	56.5	-	-	113.1	-
CLIP-T5 ♠	100%	69.8	70.8	17.4	66.8	59.6	32.4	27.1	108.5	20.4
VL-Adapter †	7.98%	-	-	-	67.6	56.2	-	-	111.8	-
VL-Adapter ♠	7.16%	69.4	70.0	16.4	65.9	57.6	31.4	27.2	105.6	20.1
VL-HyperPELT	6.62%	69.6	70.3	16.8	66.3	57.9	32.1	27.0	108.2	20.1

Table 2: Experimental results on V&L banchmarks. We report vqa-score for VQA, gqa-score for GQA and various metrics for image captioning (B@4: BLEU@4, M: METEOR, C: CIDEr, S: SPICE). †: Results from the paper of Cho et al. (2021) and Sung et al. (2021), ♠: Our re-implementation of Sung et al. (2021).

tasks of CB and BoolQ from SuperGLUE, even though the backbone *T5* was previously trained on the train sets of these two, the performance of all methods differs a lot. The two baselines still do not work with very few samples, like 4 and 16 samples. Therefore, we assume that the two baselines suffer from catastrophic forgetting problems to some degree during multi-task training. In contrast, our proposed *HyperPELT* works effectively on these two tasks. We speculate that the reason might be the use of hypernetworks on both prefix-tuning and adapter-tuning modules of transformer. We leave this exploration to our future work.

Besides, we show the results of *Prompt-tuning* (Lester et al., 2021) and fine-tuning only the task embedding in our *HyperPELT*. Note that in this comparison, we keep the same trainable parameters between these two methods, *i.e.,* $\mathbb{R}^{N\times d_\tau}$, where N denotes the prompt length in *Prompt-tuning* method. Our *HyperPELT TaskEmbed* mostly achieves a comparable or even better performance than *Prompt-tuning*.

4.5 Results on Vision-and-Language Benchmarks

We compare the pre-trained and full fine-tuning *VL-T5* (Cho et al., 2021), and other adapter-based methods built on top of *T5*, *i.e., CLIP-T5* and *VL-Adapter* (Sung et al., 2021) in the multi-task training setting. The results and the number of trainable parameters are reported in Table 2. Since the used dataset is slightly different from Sung et al. (2021) and their checkpoint is not avaliable at this time, we re-implement *CLIP-T5* and *VL-Adpater*. Compared to which, our method achieves a comparable performance with a fewer number of trainable parameters (*e.g.*, 7.16% of *VL-Adapter* vs. 6.62% of *VL-HyperPELT*).

We further evaluate our models on multimodal few shot learning tasks and show its superiority in appendix E.1. To our best knowledge, we are the first to employ the visual modality to tune the very few parameters of different transformer blocks, instead of normally inserting image patch tokens to the input sequence. Experimental results evidence the effectiveness of our novel approach, thus providing a new perspective on how to extend the multi-modality capability on top of PLMs.

5 Discussion and Conclusion

In this paper, we propose a unified parameter-efficient tuning framework for multitasks. On the one hand, we use the hypernetwork to reduce the scale of trainable parameters of existing adapter-tuning and prefix-tuning modules. On the other hand, for the V&L tasks, we directly integrate the image features into the prefix vectors as well as adapters, which further reduces the number of trainable parameters for processing visual input. Extensive experiments on pure language and V&L tasks demonstrate the superiority of our proposed framework in both multi-tasking and few-shot settings. In the future, we plan to explore more combination of methods across tuning task-specific and visual-specific parameters for different modules of PLMs.

Limitations

Our experiments are conducted based on the *T5-base* pre-trained language model. Due to the computational resource constraints, we did not conduct experiments on other similar PLMs, such as *BART*, and *T5* model with larger scale, such as *T5-large* and *T5-3B*. Although we believe our conclusion can generalize to other backbones since T5 is a classical encoder-decoder model, we will conduct more experiments to confirm for future work.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. 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 3874–3884. Association for Computational Linguistics.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. *CoRR*, abs/2204.14198.

Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying vision-and-language tasks via text generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1931–1942. PMLR.

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.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *CoRR*, abs/2203.06904.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6325–6334. IEEE Computer Society.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *CoRR*, abs/2110.04366.

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.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6700–6709. Computer Vision Foundation / IEEE.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.*, 123(1):32–73.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *CoRR*, abs/1911.03090.

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.

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.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. 2021b. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *CoRR*, abs/2102.01386.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 565–576. Association for Computational Linguistics.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. Unipelt: A unified framework for parameter-efficient language model tuning. 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 6253–6264. Association for Computational Linguistics.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. OK-VQA: A visual question answering benchmark requiring external knowledge. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3195–3204. Computer Vision Foundation / IEEE.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2021. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. *CoRR*, abs/2112.06825.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 200–212.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. 2018. Visual entailment task for visually-grounded language learning. *CoRR*, abs/1811.10582.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1298–1308. Association for Computational Linguistics.

A Related Work

In this section, we review recent research on parameter-efficient tuning for pure language and V&L tasks, as well as the corresponding work for multi-task learning.

Method	Number of Tunable Parameters
Prompt Tuning	$N \times d$
Prefix Tuning	$N \times d + (1 + 2 \times L) \times d_{\text{mid}} \times d \times B_{\text{attn}}$
Adapter	$2 \times d_{\text{mid}} \times d \times (B_{\text{attn}} + B_{\text{ffn}}) \times L$
MAM Adapter	$N \times d + (1 + 2 \times L) \times d_{\text{mid}} \times d \times B_{\text{attn}} + 2 \times d_{\text{mid}} \times d \times B_{\text{ffn}} \times L$
HYPERFORMER++	$(N + B_{\text{attn}} + B_{\text{ffn}} + L) \times d_t + d_t \times d_I^{\text{mid}} + d_I^{\text{mid}} \times d_I + 2 \times d_I \times (d_{\text{mid}} \times d)$
HyperPELT	$(N + B_{\text{attn}} + B_{\text{ffn}} + L) \times d_t + d_t \times d_I^{\text{mid}} + d_I^{\text{mid}} \times d_I + 2 \times d_I \times d + 2 \times d_I \times (d_{\text{mid}} \times d)$

Table 3: Number of tunable parameters of various parameter-efficient tuning methods with T5 models.

A.1 Parameter-Efficient Multi-task Learning

As recent models grow rapidly in size, how to fine-tune pretrained models with a small number of trainable parameters becomes more crucial. Existing research (Liu et al., 2021a; Ding et al., 2022) has explored a large amount of methods on parameter-efficient tuning. These methods generally include two categories according to whether new trainable parameters are introduced. One category is that only a subset of model parameters can be updated while freezing the remain (Liu et al., 2021b; Lee et al., 2019). The other is introducing a few task-specific new parameters to different parts of pretrained models, such as multi-head attention (Li and Liang, 2021) and feedforward layers (Houlsby et al., 2019). In this method, a small network (often named as hypernetwork with the input embedding named as hyper-embedding) is often used to generate weights for a main network.

On the other hand, learning a unified model to perform well on multiple tasks (*i.e.,* multi-task learning) is a challenging problem. It has to address many challenges such as catastrophic forgetting, and model overfitting in low-resource tasks while underfitting in high-resource tasks (Aharoni et al., 2019). Radford et al. (2019) highlights the ability of language models to perform a wide range of multitasks in a zero-shot setting. Mahabadi et al. (2021) proposes to use a shared hypernetwork (von Oswald et al., 2020) to generate weights for a small number of parameters in adapter modules, thus to allow the model to adapt to each individual task in a parameter-efficient manner.

A range of recent work aims to unify parameter-efficient tuning methods (He et al., 2021; Mao et al., 2022), to achieve better tuning performance. We explore a framework to generate weights for different PELT methods using the hypernetwork. Compared to only generating weights for adapters, empirical results indicate that generating weights for multiple modules of PLMs achieves superior performance with fewer trainable parameters.

A.2 Parameter-Efficient Tuning towards Vision-and-Language

Building vision-and-language models on top of PLMs pretrained on pure large text corpora has led to a noticeable improvement to V&L tasks (Cho et al., 2021). There is a series of recent work that extends the ability of language models to handle multimodal input in a parameter-efficient manner. For example, *Frozen* (Tsimpoukelli et al., 2021) aligns the image representation to the text representation space of frozen *GPT* model which thus is able to generate captions for images. *VL-Adapter* (Sung et al., 2021) introduces a limited set of new trainable parameters to *T5* via the adapter-tuning approach that can match the performance of fine-tuning the entire model. Flamingo (Alayrac et al., 2022) uses an extra cross-attention module, whose keys and values are generated via visual features, thus enabling language modeling conditioned on visual inputs. Different from existing work, we propose a novel perspective of parameter-efficient multimodal fusion. We introduce a seperate visual-specific hypernetwork for handling visual input and generating weights for PLMs.

B Mutli-task Learning Problem Formulation

Our paper targets at a general multi-task learning problem, where we are given the data from a set of tasks $\{\mathcal{D}_\tau\}_{\tau=1}^{T}$. T is the total number of tasks and $\mathcal{D}_\tau = \{(x_\tau^i, y_\tau^i)\}_{i=1}^{N_\tau}$ is the training data of the τ-th task with N_τ samples. We are also given a large-scale pretrained language model, i.e., *T5*, parameterized by θ, which generates the output y_τ^i for input x_τ^i. The standard multi-task finetuning minimizes the following loss on the training set:

$$\mathcal{L}_{\text{total}} = \sum_{\tau=1}^{T} \sum_{(x_\tau^i, y_\tau^i) \in \mathcal{D}_\tau} \mathcal{L}_{\text{task}}(\theta, x_\tau^i, y_\tau^i), \quad (1)$$

Task	Input Text	Target Text
GLUE Tasks		
CoLA	cola sentence: [sentence]	acceptable/unacceptable
SST-2	sst2 sentence: [sentence]	positive/negative
MRPC	mrpc sentence1: [sentence1] sentence2: [sentence2]	equivalent/not_equivalent
QQP	qqp question1: [question1] question2: [question2]	duplicate/not_duplicate
STS-B	stsb sentence1: [sentence1] sentence2: [sentence2]	0.0 - 5.0
MNLI	mnli hypothesis: [hypothesis] premise: [premise]	entailment/neutral/contradiction
QNLI	qnli question: [question] sentence: [sentence]	entailment/not_entailment
RTE	rte sentence1: [sentence1] sentence2: [sentence2]	entailment/not_entailment
Few-shot Tasks		
CB	cb hypothesis: [hypothesis] premise: [premise]	entailment/neutral/contradiction
BoolQ	boolq question: [question] context: [context]	True/False
IMDB	imdb sentence: [sentence]	positive/negative
PAWS	paws sentence1: [sentence1] sentence2: [sentence2]	equivalent/not_equivalent
Vision-and-Language Tasks		
COCO	caption:	[caption]
VQA	vqa question: [question]	[answer]
GQA	gqa question: [question]	[answer]
Vision-and-Language Few-shot Tasks		
OKVQA	okvqa question: [question]	[answer]
SNLI-VE	snli-ve premise: [premise]	entailment/neutral/contradiction

Table 4: Input-output formats for NLU and Vision-and-Language tasks. Following Raffel et al. (2020); Cho et al. (2021), we use different prefixes (such as "cola sentence:", "vqa question:") for questions from different datasets.

where $\mathcal{L}_{\text{task}}$ is the loss function of the tasks that is usually defined as the cross-entropy loss. Our goal is to efficiently finetune the given model in this multi-task learning setting, allowing knowledge sharing across tasks, and at the same time, enabling the model to adapt to each individual task.

We aim to integrate a unified hypernetwork-based parameter-efficient transfer learning method into a multi-task transformer model. In other word, we insert the parameters generated by the hypernetworks $\Delta\theta$ into the layer and attention blocks of PLMs. During training, we only update the hypernetwork parameters θ_h with hyper-embedding $\{I_\tau\}_{\tau=1}^{T}$ and parameters in layer normalization, while the remaining model parameters in θ are fixed as in the Equation 2.

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \sum_{\tau=1}^{T} \sum_{(x_\tau^i, y_\tau^i)\in\mathcal{D}_\tau} \mathcal{L}_{\text{task}}(\Delta\theta, \theta, x_\tau^i, y_\tau^i) \\ &= \sum_{\tau=1}^{T} \sum_{(x_\tau^i, y_\tau^i)\in\mathcal{D}_\tau} \mathcal{L}_{\text{task}}(I_\tau, \theta_h, \theta, x_\tau^i, y_\tau^i)\end{aligned}, \tag{2}$$

C Number of Tunable Parameters

Following He et al. (2021), to simplify the computation of tunable parameters, we compute the sum of parameter used in one encoder layer and one decoder layer as the parameter overhead of one single layer of the pre-trained encoder-decoder model. T5 has an encoder-decoder structure that has L layers. Each layer has B_{attn} blocks and B_{ffn} blocks. For the encoder-decoder models like T5, $B_{\text{attn}} = 3$: the encoder self-attention block, the decoder self-attention block and the decoder cross-attention block and $B_{\text{ffn}} = 2$: encoder feed-forward block and decoder feed-forward block. For modifications applied at the attention blocks, the number of tunable parameters is computed by $\theta_{\text{attn}} = \theta_{\text{W}}^{\text{attn}} \times B_{\text{attn}} \times L$, where $\theta_{\text{W}}^{\text{attn}}$ denotes the number of parameters used for one attention sub-layer. Similarly, the number of tunable parameters for the FFN sub-layers is computed by $\theta_{\text{ffn}} = \theta_{\text{W}}^{\text{ffn}} \times B_{\text{ffn}} \times L$. Finally, the total number of tunable parameters for prefix tuning and adapter variants is $\theta = \theta_{\text{attn}} + \theta_{\text{ffn}}$ as applicable. Using T5 as an example, we present the number of parameters used by several representative methods throughout our paper in Tab. 3.

D Experimental Setup

D.1 Input-Output Formats

As shown in Tab. 4, we formulate the input text and labels from each task to the corresponding

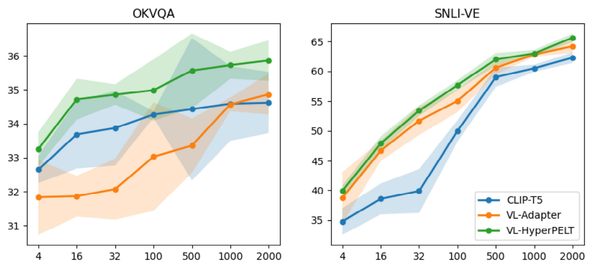

Figure 3: Few-shot domain transfer results of two different V&L tasks averaged across 5 seeds. We report the vqa-score on OKVQA validation split, and the accuracy on SNLI-VE test-P split.

target text, and we learn these different tasks by predicting target text with the language modeling objective in Eq. 2.

E Additional Results and Analysis

E.1 Multimodal Few-shot Learning

We further use the models trained on V&L tasks as reported in Figure 3 and evaluate them on the test set after few-shot fine-tuning on OKVQA (Marino et al., 2019) and SNLI-VE (Xie et al., 2018). For OKVQA, since there is no test set, we split its original validation set into two halves, one for validating and the other for testing. For SNLI-VE, we use its validation set for validating, and test-P set for testing and reporting results. We follow the methods in Section 4.4 to select samples, and report results in Figure 3.

Compared with the full parameter fine-tuning, *i.e., CLIP-T5*, and the previous parameter-efficient V&L method *VL-Adapter*, our method achieves the best performance. It is also worth noting that for the used five random seeds, the variance of our method is generally smaller than *VL-Adapter*, which indicates that our method is more robust in this few-shot learning scenario. We believe that our framework, though training less parameters, can still capture knowledge across tasks and transfer them in the multimodal few-shot setting.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 6

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Appendix D.2

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All of the datasets used in this paper use open-source licenses, and we make no modifications to the datasets in this paper. We will mark the open source licenses of the datasets in the open-source repository.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix D.2

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
These problems have been discussed in the original paper or websites which published the datasets.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
These information have been stated in the original paper or websites which published the datasets. We cite the link of each datasets used andthe reviewer can find these information there.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix D.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 and Appendix D.2

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix D.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix D.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.