LayerConnect: Hypernetwork-Assisted Inter-Layer Connector to Enhance Parameter Efficiency

Haoxiang Shi[1]*†, Rongsheng Zhang[2*], Jiaan Wang[3†], Cen Wang[4]
Yinhe Zheng[5] and Tetsuya Sakai[1]
[1] Waseda University, Tokyo, Japan [2] Fuxi AI Lab, NetEase Inc., Hangzhou, China
[3] Soochow University, Suzhou, China [4] KDDI Research Inc., Japan [5] Lingxin AI, China
hollis.shi@toki.waseda.jp, tetsuya@waseda.jp, zhangrongsheng@corp.netease.com
jawang1@stu.suda.edu.cn, ce-wang@kddi-research.jp, zhengyinhe1@163.com

Abstract

Pre-trained Language Models (PLMs) are the cornerstone of the modern Natural Language Processing (NLP). However, as PLMs become heavier, fine tuning all their parameters loses their efficiency. In this paper, we propose LayerConnect (hypernetwork-assisted inter-layer connectors) to enhance inference efficiency. Specifically, a light-weight connector with a linear structure is inserted between two Transformer layers, and the parameters inside each connector are tuned by a hypernetwork comprising an interpolator and a down-sampler. We conduct extensive experiments on the widely used GLUE benchmark. The experimental results verify the inference efficiency of our model. Compared to Adapter, with our structure, parameters are reduced to approximately 11.75% for base PLMs and 8.82% for large PLMs, while the performance degradation is kept to less than 5% (2.07 points on average).

1 Introduction

The emergence of pre-trained language models (PLMs) has brought Natural Language Processing (NLP) to a new era (Qiu et al., 2020). Fine tuning all PLMs parameters has become the most common strategy to apply them to downstream tasks (Zheng et al., 2021a; Lai et al., 2021; Wang et al., 2021, 2022; Chen et al., 2022; He et al., 2022; Zheng et al., 2020; Wang et al., 2020; Zheng et al., 2021b,c; Zhang et al., 2020; Zhou et al., 2021). However, in this manner, one should store a full copy of a PLM for each downstream task because all its parameters are updated during the fine-tuning stage. Thus, this strategy challenges the usage of the PLMs on edge devices (e.g., mobile phone and embedded systems) that have limited storage space. Under this background, parameter efficiency has gradually attracted attention from both the research community and industries (Houlsby et al., 2019; Li and Liang, 2021; Zaken et al., 2021; Liu et al., 2021; Ding et al., 2022; Zhang et al., 2021). The core concept of parameter efficiency is to reduce the trainable parameters of PLMs. Consequently, when deploying a PLM, only the different trained parameters from multiple tasks must be stored, together with one copy of the shared frozen parameters.

Currently, there are three main types of parameter efficiency methods: (1) Specification-based methods train only a small part of the parameters in the original PLM, while the others are frozen. For example, Bitfit (Zaken et al., 2021) only updates the parameters of the bias and task-specific layers in PLMs. (2) Prompt-based methods prepend additional context (i.e., prompts) to the original input, and change only the prompt parameters during fine tuning. In this manner, P-tuning (Liu et al., 2021) adopts an extra network to functionalize the optimization of the continuous prompt embedding. Then, Prefix-tuning (Li and Liang, 2021) is proposed to use trainable prefixes to accomplish a parameter-efficiency task. (3) The Adapter-based methods inject small-scale neural modules to PLMs and update only these modules during fine tuning. Adapter (Houlsby et al., 2019) simply injects two linearly trainable projectors (i.e., Adapter layers) into each transformer layer of PLMs. Karimi Mahabadi et al. (2021) further make the weights of Adapter layers conditioned on hypernetworks (Ha et al., 2016) facilitating parameter sharing across multiple tasks. Later, Hu et al. (2021) propose Low-Rank Adaptation (LoRA), a layer-parallel structure which contains less parameters than Adapter.

In view of hypernetwork (Ha et al., 2016) which generates weights for another network, although its superiority has been proved in multi-task parameter efficiency (Karimi Mahabadi et al., 2021), the single linear projector used in their hypernetwork continues to have under-explored abilities (e.g., avoid-

*These two authors contributed equally.
† This work is mainly done when SHI and WANG were interning at NetEase Inc..

ing local optima and performance compensation when linear structure size is reduced). Furthermore, the PLM used in Karimi Mahabadi et al. (2021) is T5 (Raffel et al., 2020), a pre-trained encoder-decoder model. Therefore, the effectiveness of the hypernetwork remains unknown in encoder-only PLMs (e.g., BERT and RoBERTa). To this end, we propose LayerConnect, a hypernetwork-assisted inter-layer connector. Specifically, in the proposed architecture, we first insert a connector between every adjacent transformer layer in a PLM. Only the parameters of the inserted connector are updated during the fine-tuning stage. Then, to share parameters across multiple tasks and avoid local convergence, the weights of the connector are conditioned on a well-designed hypernetwork that is equipped with an interpolator and a down-sampler. The proposed hypernetwork reduces the required parameters in the connectors and ensures sufficient optimization space. We conduct extensive experiments on the GLUE benchmark (Wang et al., 2018). The experimental results show that compared with Adapter, our LayerConnect requires only 11.75% of its parameters, while keeping the performance degradation to less than 5%.

2 Method

We propose hypernetwork-assisted inter-layer connectors (i.e., LayerConnect), as shown in Figure 1, the Transformer encoder (Vaswani et al., 2017) is used as the backbone of our LayerConnect and a connector is inserted between Transformer layers, named inter-layer connector. Specifically, the connectors linearly transform the hidden states of each Transformer layer:

$$C^l(X_l) = X_l \cdot A^l + B^l \qquad (1)$$

where $\boldsymbol{X}_l \in \mathbb{R}^{n\times d}$ is the hidden state of the l-th Transformer layer ($l \in \{1, 2, \cdots, L\}$), n and d are the input size and the hidden size of Transformer, respectively. Further, $A^l \in \mathbb{R}^{n\times n}$ and $B^l \in \mathbb{R}^{n\times d}$ are the trainable parameters where each row in A^l or B^l is a repetition of a vector a^l or b^l with length n. In the entire model, the only tunable parameters are $\boldsymbol{A} = [a^1; a^2; \cdots; a^L]$ and $\boldsymbol{B} = [b^1; b^2; \cdots; b^L]$. Compared with previous intra-layer Adapters (Houlsby et al., 2019; Karimi Mahabadi et al., 2021), our inter-layer connector contains fewer trainable parameters due to (1) the number of inserted layers in our model is L while the counterpart of others is $2L$; and (2) the

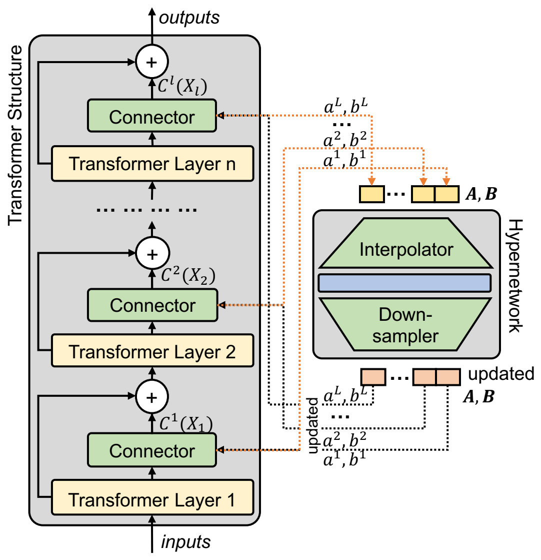

Figure 1: LayerConnect architecture.

parameters count in our connector is $n+d$, which is less than $2d \cdot d_m (1 \le d_m < d)$ existing Adapters.

To prevent the model from converging to the local optima due to the small number of parameters, the weights of $\boldsymbol{A}$ and $\boldsymbol{B}$ are conditioned on a hypernetwork. Unlike the approach of Karimi Mahabadi et al. (2021), whose hypernetwork is a single linear projector, we design a new hypernetwork equipped with an interpolator and a down-sampler. In detail, we first utilize a learned task embedding $e_t \in \mathbb{R}^{L\times d}$ for each task, and then, use an interpolator to raise dimensionality, followed by a ReLU non-linearity:

$$\boldsymbol{e}_{in} = \mathrm{ReLU}(W_{in} e_t) \qquad (2)$$

where $W_{in} \in \mathbb{R}^{mL\times L}$ is the set of trainable parameters of the interpolator and m is a hyper parameter. In this manner, sufficient interpolator parameters are provided to optimize the connectors. Subsequently, a down-sampler is used to project $\boldsymbol{X}_{in}$ back to the original dimensionality. Finally, the weights of $\boldsymbol{A}$ and $\boldsymbol{B}$ are generated from the down-sampled vector:

$$\boldsymbol{e}_{ds} = W_{ds} \boldsymbol{e}_{in} \qquad (3)$$

$$(\boldsymbol{A}^\top, \boldsymbol{B}^\top) = \boldsymbol{X}_{ds}(W^A, W^B) \qquad (4)$$

where $W_{ds} \in \mathbb{R}^{L\times mL}$, $W^A \in \mathbb{R}^{d\times n}$ and $W^B \in \mathbb{R}^{d\times d}$ are trainable parameters.

We specify the parameters of hypernetwork for different tasks during fine tuning. After that, the hypernetwork is discarded when deploying models, indicating that the parameters of the hypernetwork do not influence storage space during reasoning.

Method	Para. Size	SST-2	QNLI	M.-MM	QQP		RTE	STS-B		CoLA	MRPC	
		Acc.	Acc.	Acc.	Acc.	F1	Acc.	PCC	SCC	Acc.	Acc.	F1
Vanilla Fine-Tuning	108.31M	91.46	91.00	83.91	91.08	87.93	69.67	89.73	89.35	60.74	86.17	90.12
ADA$_1$	27.66K	90.60	88.14	78.77	85.54	80.51	63.82	86.99	86.97	51.12	77.79	85.22
ADA$_8$-random	13.06K	90.51	86.34	72.54	82.71	77.90	58.48	83.63	83.25	34.36	72.07	82.67
ADA$_8$	156.76K	91.48	90.84	81.46	88.11	83.98	69.82	**88.08**	**88.08**	**57.12**	84.46	88.94
ADA$_8$-hyper	156.76K	91.40	**90.87**	81.96	**88.58**	**84.71**	**70.51**	88.07	88.07	56.01	85.29	89.42
Bitfit	102.91K	90.78	87.70	78.15	85.50	80.90	67.22	87.22	87.22	47.92	78.14	85.67
LayerCon. (our)	18.43K	**92.45**	88.58	78.96	85.73	81.16	68.73	87.46	87.63	53.86	**86.17**	**90.11**
LayerCon. (w/o B)	9.21K	90.52	87.13	75.62	84.71	79.79	64.98	85.86	85.74	49.72	80.94	87.18
LayerCon. (w/o A)	9.21K	89.48	85.25	75.63	82.49	77.97	57.40	70.10	81.25	42.98	69.61	81.66
LayerCon. (w/o A&B)	18.43K	89.79	84.79	**82.55**	83.06	78.43	58.19	84.63	84.63	40.94	74.21	83.11

Table 1: Experimental results on the GLUE benchmark, based on BERT-base. M.-MM = MNLI-MM. Para. Size: additional parameters for PLM inference in a single task. The **bold** denotes the best results.

Method	Para. Size	SST-2	QNLI	M.-MM	QQP		RTE	STS-B		CoLA	MRPC	
		Acc.	Acc.	Acc.	Acc.	F1	Acc.	PCC	SCC	Acc.	Acc.	F1
Vanilla Fine-Tuning	125.31M	94.65	93.11	87.16	91.49	88.70	77.36	90.00	89.79	59.66	88.57	91.75
ADA$_1$	27.66K	92.09	88.19	82.48	85.74	80.95	58.34	81.93	81.99	37.82	77.09	81.92
ADA$_8$-random	13.06K	90.51	80.97	75.46	82.69	77.82	58.53	87.38	87.26	30.09	68.38	81.22
ADA$_8$	156.76K	93.85	90.90	**86.63**	**88.29**	**84.56**	70.18	**90.63**	**90.31**	58.92	**87.60**	**91.08**
ADA$_8$-hyper	156.76K	**94.50**	**91.77**	85.13	87.30	83.23	**70.21**	88.86	89.22	**61.14**	84.74	88.66
Bitfit	103.68K	90.60	88.49	75.79	85.23	81.08	67.22	84.56	84.41	54.17	80.88	86.15
LayerCon. (our)	18.43K	93.58	89.51	83.57	87.04	82.98	66.13	86.74	86.85	55.17	79.88	85.67
LayerCon. (w/o B)	9.21K	91.28	87.59	80.66	85.34	81.03	59.01	83.08	83.45	41.86	75.29	83.66
LayerCon. (w/o A)	9.21K	92.66	84.57	71.52	83.55	79.46	60.72	78.09	78.38	43.27	70.10	81.68
LayerCon. (w/o A&B)	18.43K	91.50	86.73	72.76	85.90	81.91	63.29	84.66	83.55	46.56	73.70	82.21

Table 2: Experimental results on GLUE benchmark, based on RoBERTa-base.

Therefore, we only need to restore the connector parameters, i.e., $\boldsymbol{A}$ and $\boldsymbol{B}$, for each task.

3 Experiment

3.1 Setup

The PLMs used in our experiments are BERT and RoBERTa models implemented by the *Transformers* library (Wolf et al., 2020). More details are given in Appendix A.

3.2 Dataset and Metric

Following previous work (Houlsby et al., 2019; Karimi Mahabadi et al., 2021), we evaluate model performance on the GLUE benchmark (Wang et al., 2018). Note that some test sets in GLUE are not publicly available, thus, the corresponding validation sets are used as alternatives. The main metric is accuracy (Acc.). For the QQP and MRPC, we use the F1-measure; for the STS-B, we use the Spearman and Pearson correlation coefficients (SCC and PCC).

3.3 Baselines

We compared our model with the following methods. (1) Vanilla Fine-Tuning: Fine-tuning all parameters in PLMs; (2) Adapter (Houlsby et al., 2019) under different settings: specifically, we adjust the middle size of its bottleneck structure to 1 and 8, which we denoted as ADA$_1$ and ADA$_8$, respectively; (3) ADA-random: We randomly add an Adapter layer[1] to the PLM as a baseline. (4) ADA$_8$ -hyper: We equip the original Adapter of size 8 with our hypernetwork. (5) Bitfit (Zaken et al., 2021).

Moreover, we also modify our LayerConnect (LayerCon.) into the following three variations: (1) LayerCon. (w/o B) uses only the hypernetwork to generate the weights of $\boldsymbol{A}$; (2) LayerCon. (w/o A) uses only the hypernetwork to generate the weights of $\boldsymbol{B}$; (3) LayerCon. (w/o A&B) removes the hypernetwork and randomly initializes the weights of $\boldsymbol{A}$ and $\boldsymbol{B}$.

3.4 Results

We first analyze the effectiveness of LayerConnect and hypernetwork through the experimental results of the base model (Table 1 and Table 2).

Effectiveness of LayerConnect. Our LayerConnect outperforms Bitfit in most of the benchmark datasets. For the Adapter baselines, a larger Adapter achieves the higher performance (ADA$_1$ vs. ADA$_8$). As a strong baseline, Adapter even outperforms the vanilla fine-tuning in some benchmark datasets. However, when the Adapter is equipped with parameters similar to those of LayerConnect, its performance declined (ADA$_8$-random

[1]We choose to add only one Adapter layer since the trainable parameters in this strategy are most close to our model.

Method	Para. Size	SST-2	QNLI	M.-MM	QQP		RTE	STS-B		CoLA	MRPC	
		Acc.	Acc.	Acc.	Acc.	F1	Acc.	PCC	SCC	Acc.	Acc.	F1
Vanilla Fine-Tuning	333.58M	93.12	92.36	86.55	91.44	88.43	75.45	90.08	90.17	65.12	87.74	91.28
ADA$_1$	76.82K	92.09	91.62	83.30	87.37	83.04	68.23	89.63	89.53	58.81	71.08	81.73
ADA$_8$-random	36.88K	91.97	83.56	77.52	85.25	80.83	64.34	84.58	85.13	57.54	78.43	85.53
ADA$_8$	417.98K	**92.93**	92.11	**86.60**	89.17	**86.30**	72.99	**90.19**	**90.06**	61.07	86.12	90.18
ADA$_8$-hyper	417.98K	92.84	**92.12**	85.90	**89.72**	86.28	**75.71**	90.03	89.51	**62.70**	**87.84**	**91.35**
Bitfit	272.38K	92.20	90.03	85.90	87.15	83.63	70.03	89.90	89.65	57.79	79.75	90.21
LayerCon. (our)	36.88K	92.59	91.10	83.36	85.72	84.43	72.42	90.08	87.63	60.90	87.50	91.24
LayerCon. (w/o B)	18.43K	92.09	89.71	81.11	85.03	80.76	68.95	88.12	88.87	57.02	80.15	86.61
LayerCon. (w/o A)	18.43K	91.86	89.99	80.67	84.70	80.24	67.15	88.50	88.25	58.58	82.35	82.04
LayerCon. (w/o A&B)	18.43K	91.97	89.86	81.77	84.56	80.05	59.21	87.45	87.42	55.63	76.47	84.81

Table 3: Experimental results on GLUE benchmark, based on BERT-Large.

Method	Para. Size	SST-2	QNLI	M.-MM	QQP		RTE	STS-B		CoLA	MRPC	
		Acc.	Acc.	Acc.	Acc.	F1	Acc.	PCC	SCC	Acc.	Acc.	F1
Vanilla Fine-Tuning	356.51M	96.61	94.94	89.98	91.92	89.45	85.12	92.28	92.09	65.08	88.48	91.75
ADA$_1$	76.82K	95.07	91.96	88.17	86.61	82.51	71.12	90.63	90.54	55.59	86.27	90.48
ADA$_8$-random	36.88K	95.18	75.97	84.94	84.67	80.43	73.29	88.48	88.53	52.06	83.33	88.51
ADA$_8$	417.98K	95.94	94.56	89.66	89.25	85.97	72.74	**92.01**	91.08	62.75	89.07	92.17
ADA$_8$-hyper	417.98K	**96.58**	**94.73**	**90.03**	**89.34**	**86.03**	73.19	90.05	89.88	59.06	**89.46**	**92.31**
Bitfit	273.41K	95.37	93.04	88.46	87.49	83.79	72.22	91.05	90.91	59.70	86.64	90.21
LayerCon (our)	36.88K	95.41	93.67	88.60	88.04	83.00	**76.92**	91.50	**91.46**	**63.61**	87.00	90.21
LayerCon. (w/o B)	18.43K	93.23	90.85	86.44	86.24	82.32	73.29	86.34	86.32	54.23	70.83	81.89
LayerCon. (w/o A)	18.43K	93.35	92.29	86.90	85.77	85.99	70.76	86.55	86.57	53.95	82.50	83.86
LayerCon. (w/o A&B)	36.88K	94.61	92.36	87.28	86.24	81.92	68.23	82.91	83.56	56.53	77.45	85.40

Table 4: Experimental results on GLUE benchmark, based on RoBERTa-Large.

Para. Size	CoLA(Acc.)	MNLI-MM(Acc.)
2m.	55.17	83.57
4m.	55.47	84.01
10m.	55.48	84.22

Table 5: The effect of the size of hypernetwork under the RoBERTa-base model.

vs. ADA$_8$) and is worse than ours on most datasets. Moreover, the trainable inserted parameters in our model are only 11.75% of that in ADA$_8$. The LayerConnect performance is competitive with the best Adapter, with an average performance deterioration of 1.42% (1.0 points) and 3.96% (3.3 points) for BERT-base and RoBERTa-base, respectively.

Effectiveness of the Hypernetwork. We also demonstrate that the hypernetwork is necessary for the connectors. We compare the results of our model with or without hypernetwork (LayerCon. vs. LayerCon. (w/o A&B)). The former outperforms the latter except for MNLI-MM. We analyze this because the hypernetwork provides sufficient parameters during the fine-tuning stage to tune the connectors effectively. We also remove the $\boldsymbol{A}$ and $\boldsymbol{B}$ in our hypernetwork, respectively, to further reduce the parameters. However, the performances are all worse than that of the original hypernetwork (LayerCon. vs. LayerCon. (w/o B) and LayerCon. (w/o A)), demonstrating the rationality of our hypernetwork. Moreover, we attempt to equip the best Adapter with our hypernetwork, the results are only slightly changed compared with the original

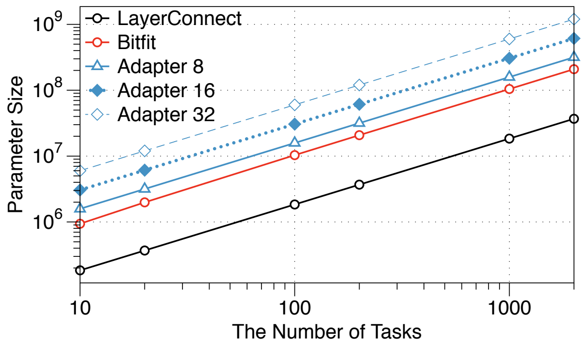

Figure 2: Parameter size as the number of tasks grows.

Adapter (ADA$_8$ vs. ADA$_8$-hyper). We conclude that our hypernetwork is more effective on fewer parameters' inserted layers since its provided parameters are more valuable to them.

We additionally test our model and baselines based on BERT-large and RoBERTa-large models. The corresponding results are shown in Table 3 and Table 4, respectively. The average performance deterioration is 2.12% (1.74 points) and 0.79% (0.85 points) for BERT-large and RoBERTa-large, respectively; both are lower than that of the base models. The trainable inserted parameters in LayerConnect are only 8.82% of that in ADA$_8$.

4 Discussion

As presented in Table 5, we adjust the middle value of the hypernetwork shape in our method to 4 and

10, respectively. A larger size resulted in a slight performance improvement. This result suggests that the hypernetwork size has a positive effect in our LayerConnect.

We compare the parameter size in the deployment for model inference as the number of tasks grows, as shown in Figure 2. In model inference, considering the scalability, for the Bitfit, one may at least store multiple copies of the bias (i.e., 1,324K). However, the structure and the bias locations are also necessary. The parameter sizes of the Adapter and the Adapter with the hypernetwork are the same (i.e., the size of the bottleneck structure). In contrast, our connectors require an extremely small parameter size. That is to say, one can easily deploy several tens of the tasks, and the total size of the parameters just arrives at the same level as that of a single task when using the others.

5 Conclusion

To enhance the parameter efficiency in the fine-tuning stage of PLMs, we propose the ultra-light connectors to be embedded into the Transformer layers. Furthermore, to keep the performance of such a small structure, we use the hypernetwork to assist the tuning of the parameters within the connectors. We compare our method with mainstream methods (Adapter and Bitfit). Experimental results show that our method outperforms Bitfit in most cases. Compared with ADA_8 (with best performance in most cases), our method reduces the trainable parameters to 11.75% for base models and 8.82% for large models, while keeping the performance degradation to less than 5% (2.07 points on average). By analyzing the results, we verify that reducing the number of parameters on the basis of Adapter will seriously reduce the performance, and the introduction of hypernetwork promises an effective way to compensate for performance. This reveals a new direction of the study on model efficiency. Additionally, LayerConnect shows scalability, especially for the memory-sensitive and storage-sensitive edge devices e.g., smartphones, embedded devices, micro containers, and IoT/IoH devices.

References

Weijie Chen, Yongzhu Chang, Rongsheng Zhang, Jiashu Pu, Guandan Chen, Le Zhang, Yadong Xi, Yijiang Chen, and Chang Su. 2022. Probing simile knowledge from pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5875–5887, Dublin, Ireland. 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, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10749–10757.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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

Siyu Lai, Hui Huang, Dong Jing, Yufeng Chen, Jinan Xu, and Jian Liu. 2021. Saliency-based multi-view mixed language training for zero-shot cross-lingual classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 599–610, Punta Cana, Dominican Republic. 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 (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *CoRR*, abs/2103.10385.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Colin Raffel, Noam M. 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. *ArXiv*, abs/1910.10683.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Jiaan Wang, Zhixu Li, Qiang Yang, Jianfeng Qu, Zhigang Chen, Qingsheng Liu, and Guoping Hu. 2021. Sportssum2.0: Generating high-quality sports news from live text commentary. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, page 3463–3467, New York, NY, USA. Association for Computing Machinery.

Jiaan Wang, Zhixu Li, Tingyi Zhang, Duo Zheng, Jianfeng Qu, An Liu, Lei Zhao, and Zhigang Chen. 2022. Knowledge enhanced sports game summarization. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 1045–1053, New York, NY, USA. Association for Computing Machinery.

Yida Wang, Pei Ke, Yinhe Zheng, Kaili Huang, Yong Jiang, Xiaoyan Zhu, and Minlie Huang. 2020. A large-scale chinese short-text conversation dataset. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 91–103. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199.

Rongsheng Zhang, Yinhe Zheng, Xiaoxi Mao, and Minlie Huang. 2021. Unsupervised domain adaptation with adapter. *arXiv preprint arXiv:2111.00667*.

Rongsheng Zhang, Yinhe Zheng, Jianzhi Shao, Xiaoxi Mao, Yadong Xi, and Minlie Huang. 2020. Dialogue distillation: Open-domain dialogue augmentation using unpaired data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3449–3460.

Duo Zheng, Zipeng Xu, Fandong Meng, Xiaojie Wang, Jiaan Wang, and Jie Zhou. 2021a. Enhancing visual dialog questioner with entity-based strategy learning and augmented guesser. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1839–1851, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinhe Zheng, Guanyi Chen, Xin Liu, and Ke Lin. 2021b. Mmchat: Multi-modal chat dataset on social media. *arXiv preprint arXiv:2108.07154*.

Yinhe Zheng, Zikai Chen, Rongsheng Zhang, Shilei Huang, Xiaoxi Mao, and Minlie Huang. 2021c. Stylized dialogue response generation using stylized unpaired texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14558–14567.

Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. 2020. A pre-training based personalized dialogue generation model with persona-sparse data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9693–9700.

Hao Zhou, Pei Ke, Zheng Zhang, Yuxian Gu, Yinhe Zheng, Chujie Zheng, Yida Wang, Chen Henry Wu, Hao Sun, Xiaocong Yang, et al. 2021. Eva: An open-domain chinese dialogue system with large-scale generative pre-training. *arXiv preprint arXiv:2108.01547*.

A Setup and Evaluation Metric

A.1 Setup

Connectors: The LayerConn.(ours), LayerConn.(w/o B) and LayerConn.(w/o A) are shown as Equation 5 to Equation 7, respectively:

$$C^l(X_l) = X_l \cdot A^l + B^l \tag{5}$$

$$C^l(X_l) = X_l \cdot A^l \tag{6}$$

$$C^l(X_l) = B^l \tag{7}$$

where $\boldsymbol{X}_l \in \mathbb{R}^{n\times d}$, $A^l \in \mathbb{R}^{n\times n}$ and $B^l \in \mathbb{R}^{n\times d}$, $l \in \{1, 2, \cdots, L\}$) is the layer ID. For both Bert-base and RoBERTa-base model, $L = 12$; for both Bert-large and RoBERTa-large model, $L = 24$. Each row in A^l or B^l is a repetition of a vector a^l or b^l with length n. The only tunable parameters are $\boldsymbol{A} = [a^1; a^2; \cdots; a^L]$ and $\boldsymbol{B} = [b^1; b^2; \cdots; b^L]$. n and d are the input size (max. length) and the hidden size of Transformer, respectively. For both Bert-base and RoBERTa-base model, n is 128, and d is 768; for both Bert-large and RoBERTa-large model, n is 128, and d is 1024.

Hypernetwork: Aa interpolator can be expressed as:

$$\boldsymbol{e}_{in} = ReLU(W_{in}e_t) \tag{8}$$

and a down-sampler can be expressed as:

$$\boldsymbol{e}_{ds} = W_{ds}\boldsymbol{e}_{in} \tag{9}$$

where $W_{in} \in \mathbb{R}^{mL\times L}$, and $W_{ds} \in \mathbb{R}^{L\times mL}$, they are the trainable parameters of the interpolator and the down-sampler, respectively. m is a hyper parameter. In the experiment, the minimum $m = 2$.

PLMs: The learning rates are 2×10^{-5} and 5×10^{-6} for BERT (base and large) and RoBERTa (base and large), respectively. We select AdamW as the optimizer. We choose random seeds for each model and report the average results of five runs.

A.2 Evaluation Metric

The metrics vary across different tasks, expect for the commonly used accuracy and F1 Score. We briefly introduce other metrics as follows:

Pearson Correlation Coefficient. The Pearson Correlation Coefficient is used to calculate the similarity of sentence pair from S_x and S_y, as shown in Equation 10:

$$\rho_{S_x,S_y} = \frac{E[(S_x - \mu_{S_x})(S_y - \mu_{S_y})]}{\sigma_{S_x}\sigma_{S_y}} \tag{10}$$

where μ and σ are the mean and standard deviation for S_x and S_y, respectively.

Spearman Correlation Coefficient. The Spearman Correlation Coefficient is another way to calculate the similarity of sentence pair from S_x and S_y. Its format is the same with Pearson one. However, the samples are transformed into the level variable, and the calculation is simplified as shown in Equation 11:

$$\rho = \frac{6\sum_{i=1}^{n}(S_x^{(i)} - S_y^{(i)})^2}{N(N^2-1)} \tag{11}$$

where $S_x^{(i)}$ and $S_y^{(i)}$ are sentence sample from S_x and S_y, and N is the total number of the samples.

Matthews Correlation Coefficient (MCC). The performance of CoLA is evaluated by MCC, as shown in Equation 12:

$$\begin{aligned} MCC &= \frac{TP\cdot TN - FP\cdot FN}{\sqrt{Q}} \\ Q &= (TP+FP)\cdot(TP+FN)\cdot \\ &\quad (TN+FP)\cdot(TN+FN) \end{aligned} \tag{12}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

A.3 Performance Degradation

The performance degradation is calculated by Equation 13:

$$\text{Degradation} = \frac{\text{BaselineScore} - \text{OurScore}}{\text{BaselineScore}} \tag{13}$$

Negative degradation means an improvement.