Interspeech 2024
1-5 September 2024, Kos, Greece

Convolution-Augmented Parameter-Efficient Fine-Tuning for Speech Recognition

Kwangyoun Kim[1], *Suwon Shon*[1], *Yi-Te Hsu*[1], *Prashant Sridhar*[1], *Karen Livescu*[2], *Shinji Watanabe*[3]

[1]ASAPP, USA
[2]Toyota Technological Institute at Chicago, USA
[3]Carnegie Mellon University, USA

kkim@asapp.com

Abstract

Parameter-efficient fine-tuning (PEFT) methods, which train only a part of a model, yield efficient and effective models. Bottleneck approaches, such as adapters and low-rank adaptation (LoRA), have been found to be beneficial in numerous studies and are widely utilized. In this work, we propose and investigate an enhanced PEFT method that adds convolution to linear projection-based bottleneck approaches. We experiment with HuBERT, a representative speech model pre-trained with self-supervised learning, and fine-tune it for the automatic speech recognition (ASR) task to examine how the proposed PEFT method impacts training and inference. We demonstrate consistent performance improvements with a minimal increase in parameters and computational complexity.

Index Terms: speech recognition, parameter-efficient fine-tuning

1. Introduction

Even today, as cutting-edge artificial intelligence models continue to emerge, there is still not a single model that performs well on a wide variety of speech tasks. Therefore, fine-tuning of pre-trained models for downstream tasks is the most common approach to achieve superior performance. In speech research, self-supervised learning (SSL) models such as wav2vec [1], HuBERT [2], and WavLM [3] are trained to extract contextual information from audio input and generate representation vector sequences. These representation vectors are then used as input for various downstream tasks including automatic speech recognition (ASR), speech translation, speaker diarization, speaker identification, and many other speech applications [4–7].

Typically, an SSL model has not learned to perform any specific downstream tasks independently. In the case of ASR, a connectionist temporal classification (CTC) [8] or other decoder can be added. For other classification tasks, a task-specific classification network is typically employed for fine-tuning. However, the size (number of parameters) of SSL models poses challenges during fine-tuning. Updating all the model parameters can lead to catastrophic forgetting [9]. In addition, the substantial GPU memory requirements for fine-tuning such a model can be prohibitive. Furthermore, it can be challenging to store several versions of a model, each fine-tuned for different tasks.

In response, many efforts have focused on developing efficient fine-tuning methods that minimize the number of trainable parameters by freezing as many of the SSL model parameters as possible while updating only a small subset or newly added parts [10–18]. Such parameter-efficient fine-tuning (PEFT) approaches often produce little or no performance degradation compared to full fine-tuning, and in some cases, they may even achieve improvements [17, 19]. Additionally, since only the task-specific parameters for each downstream task need to be individually maintained, PEFT improves efficiency by sharing most of the parameters across tasks.

In this study, we introduce an enhanced method that effectively combines convolution with widely utilized PEFT approaches. The integration of convolution into the Transformer has already demonstrated state-of-the-art performance in ASR tasks, as seen in Conformer [20, 21], Branchformer [22], and E-Branchformer [23, 24] models. There have also been studies that employ convolutions in PEFT [18, 25]. While our work shares their motivation, there are differences in methods and analysis, and we describe our distinct contributions in this paper. Our work contributes the following:

1. We present an enhanced PEFT method that directly adds convolution to commonly used PEFT methods such as adapters and low-rank adaptation (LoRA).
2. Through experiments with the Transformer-based SSL model HuBERT, we demonstrate that our proposed method is simple yet effective.
3. We conduct extensive experiments to assess the impact of incorporating convolution into PEFT on various modules of the Transformer.

2. Parameter-Efficient Fine Tuning

Most SSL models contain a very large number of parameters, learned with substantial amounts of data, which inevitably requires resource-intensive training infrastructure. Consequently, fine-tuning all model parameters for downstream tasks or target domain datasets may be inefficient. Alternatively, recent studies have been focusing on parameter-efficient fine-tuning (PEFT) methods that train only a minimal subset of SSL model parameters or a negligible number of newly added parameters, while the majority of parameters are frozen.

2.1. Adapters

One common PEFT approach is to add an adapter module to the existing model [10, 13, 26, 27]. For example, Houlsby adapters [10] were developed for Transformer-based transfer learning for NLP tasks. Two adapter layers are added to each Transformer layer, one in the multi-headed attention (MHA) module and one in the feed-forward (FF) module. An adapter layer consists of a down-projection $\boldsymbol{W_D} \in \mathbb{R}^{d \times r}$ and an up-projection $\boldsymbol{W_U} \in \mathbb{R}^{r \times d}$ layer, with a non-linear activation between them, and utilizes a residual skip connection:

$$\boldsymbol{y} = \boldsymbol{x} + f(\boldsymbol{x}\boldsymbol{W_D})\boldsymbol{W_U} \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ denote the input and the output of an adapter layer, respectively, and f is a non-linear activation function.

10.21437/Interspeech.2024-2188

Since r is tiny compared to the original dimension d, the number of trainable parameters is small. During transfer learning, only the parameters of the adapter and the original layer-normalization layers in each MHA and FF module are updated along with the newly added downstream task-specific layers. Unlike the Houlsby adapter, [26] employs an adapter that internally has its own layer-normalization before down-projection. This approach enables training multiple adapters separately for different languages in machine translation and makes it easy to switch between languages.

2.2. LoRA

Low-rank adaptation (LoRA) is also a type of adapter, but it is intended for fine-tuning dense layers. Given a dense layer in the pre-trained model with a trainable matrix $\boldsymbol{W_o} \in \mathbb{R}^{m \times n}$, the adapted matrix $\boldsymbol{W}$ is defined as

$$\boldsymbol{W} = \boldsymbol{W_o} + \Delta \boldsymbol{W} = \boldsymbol{W_o} + \boldsymbol{AB} \tag{2}$$

where $\Delta \boldsymbol{W}$ represents the update during adaptation. $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times n}$ are low-rank decomposition matrices of $\Delta \boldsymbol{W}$, and they have a similar role to the down-projection and up-projection layers in an adapter, respectively. LoRA uses a much lower rank r than the original dimensions of $\Delta \boldsymbol{W}$, m and n ($r << m, n$), so we can indirectly fine-tune the dense layer by optimizing only a small number of parameters, thereby reducing training costs. Another advantage of LoRA is that, after adaptation, $\Delta \boldsymbol{W} = \boldsymbol{BA}$ can be merged back into $\boldsymbol{W}$, meaning that there is no increase in computational costs at inference time compared to the original model. Like a residual adapter, a LoRA layer $\boldsymbol{A}, \boldsymbol{B}$ can be trained for various tasks independently, allowing us to switch between tasks easily.

2.3. Hybrid PEFT

The Hydra method [15] utilizes LoRA for a dense layer both sequentially and in parallel during fine-tuning. Another approach obtains improved results by applying an ensemble of multiple PEFT methods to each Transformer layer: A sequential adapter is added following the FF module, a parallel adapter is placed outside of the Transformer with a residual connection, and LoRA is applied in the MHA module as well [17].

3. Convolution-Augmented PEFT

Many publicly available pre-trained SSL models are structured by sequentially stacking multiple layers of a basic Transformer [1, 2]. Each Transformer layer comprises an MHA, an FF module, and layer normalization layers, but Transformer variants [20, 23, 24] that incorporate convolutions have shown superior accuracy in ASR or other speech tasks. However, training large SSL models with such convolution-augmented Transformer variants from scratch entails challenges, including significant training costs and data scarcity. Consequently, applying such convolutional operations to the pre-trained model during the fine-tuning stage can be more economical and practical.

Inspired by these observations, our study focuses on an efficient convolutional augmented PEFT method for speech. Recent studies have introduced methods that apply convolution to PEFT. Instead of a conventional linear projection-based bottleneck-style adapter, a convolutional adapter [25] employs three convolution layers and a Squeeze-Excitation module [28]. Unlike this complex structure change, we propose a method that directly attaches a convolution between bottleneck layers. In addition to the simplicity of this approach, it also allows us to understand better how the newly added convolution layer affects performance compared to the existing adapter. Another approach is Conv-LoRA [18], which was introduced for image segmentation tasks. Conv-LoRA applies a mixture of experts (MoE) using convolutions to a LoRA layer, which enables the model to utilize local priors at multiple scales during fine-tuning. MoE is critical to the success of this approach for segmentation: When a single convolution expert is used, performance is only marginally improved or is even worse than the original LoRA. Therefore, while MoE enhances performance, it introduces additional computational costs. Instead of focusing on multiple scales, we concentrate on effectively utilizing convolution-augmented PEFT methods for each module of a Transformer.

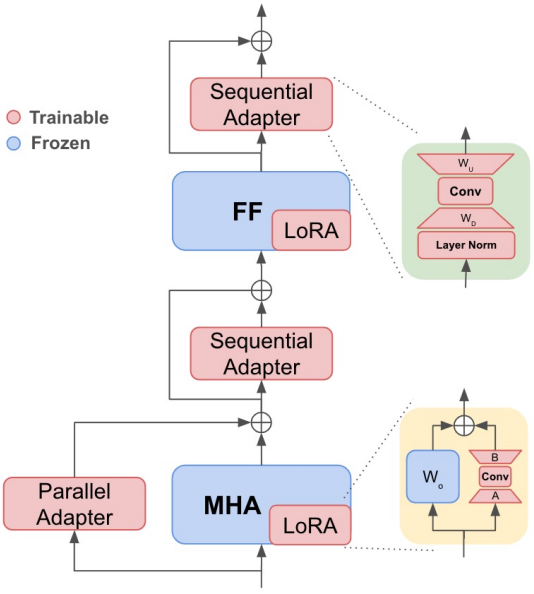

Figure 1: *Conv-augmented PEFT using adapters and LoRA for a Transformer layer, highlighting where and how they are applied. Convolution has been added to each PEFT layer. Both sequential and parallel adapters have the same structure. LoRA is applied across all four projections (query, key, value, and output) in the MHA module, and both dense layers in the FF module.*

3.1. Conv-Augmented Layer

Our proposed method of applying a convolution to an adapter and a LoRA layer is shown in Fig.1. Simply adding a 1-D convolution to the basic residual adapter [26] works as follows:

$$\boldsymbol{x'} = LayerNorm(\boldsymbol{x}) \tag{3}$$

$$\boldsymbol{y} = \boldsymbol{x} + f(\boldsymbol{c} * (\boldsymbol{x'W_D}))\boldsymbol{W_U} \tag{4}$$

where $\boldsymbol{c}$ denotes the convolution layer, and $\boldsymbol{f}$ is a non-linear activation function. The original LoRA layer works as follows:

$$\boldsymbol{y} = \boldsymbol{xW_o} + \boldsymbol{xW_D W_U} \tag{5}$$

Here, $\boldsymbol{W_D}$ and $\boldsymbol{W_U}$ are the same as $\boldsymbol{A}$ and $\boldsymbol{B}$ in Eq. 2. Then, we apply convolutions to the LoRA layer:

$$\boldsymbol{y} = \boldsymbol{xW_o} + (\boldsymbol{c} * (\boldsymbol{xW_D}))\boldsymbol{W_U} \tag{6}$$

Table 1: *The results of fine-tuning a HuBERT-base model. We compare the performance when using different PEFT methods with Full fine-tuning (Full-FT). The computational costs are profiled in the case of a 10-second input using deepspeed profiler.*[1]

	Params (M)	Trainable (M)	MACs (G)	dev-clean	dev-other	test-clean	test-other
Full FT	90.02	81.11 (90.10%)	74.07	**5.19**±0.04	13.99±0.07	**5.39**±0.04	13.82±0.03
LoRA	92.58	2.53 (2.73%)	75.39	5.81±0.10	14.41±0.10	6.07±0.07	14.29±0.15
LoRA+Conv	92.60	2.56 (2.76%)	75.40	5.71±0.05	**13.82**±0.16	5.82±0.07	13.74±0.19
Adapter	92.33	2.30 (2.50%)	75.24	5.74±0.05	14.43±0.13	6.01±0.04	14.17±0.16
Adapter+Conv	92.35	2.33 (2.52%)	75.26	5.63±0.05	13.86±0.23	5.89±0.07	**13.70**±0.25

In the case of LoRA, the added convolution precludes merging LoRA parameters with the original weight $\boldsymbol{W_o}$, which might result in a slight increase in the model size and the amount of computation. Considering that the purpose of adding convolution is to exploit local context, and the additional computational cost, we use a depth-wise convolution for both an adapter and a LoRA layer by default. However, employing more complicated and computationally expensive convolutional modules is also possible.

3.2. Conv-Augmented Layer Injection

It is common practice to place an adapter layer after the MHA or the FF module in a Transformer layer. On the other hand, LoRA is applied directly to each dense layer. As mentioned in the previous section, there are recent works [15, 17] that have studied hybrid approaches that combine sequential and parallel connections for PEFT. Within a Transformer, an MHA module leverages the global information for each input position through an attention mechanism using query, key, and value. By integrating convolution into the MHA module, we anticipate extracting the global context by simultaneously considering the input at each position with a certain range of adjacent inputs together. In the case of an adapter, adding convolutions sequentially to modules forms a similar idea to the Conformer, whereas parallel integration yields a structure similar to the Branchformer. Hence, studies demonstrating the superior performance of these Transformer variants indirectly suggest the potential for performance improvements by adding convolutions to adapter and LoRA layers. Therefore, we conduct experiments and analyze how and where convolutions can be added to the pre-trained Transformer model during PEFT to enhance performance.

4. Experiments

We conduct experiments to quantify the performance improvement afforded by incorporating depth-wise convolutions into PEFT. We utilize a pre-trained HuBERT-base[2] as the base model and conduct all experiments based on huggingface libraries. A pre-trained HuBERT model comprises a feature extractor based on convolution layers and an encoder employing Transformer layers. In our fine-tuning experiments, the feature extractor parameters remain frozen. All the parameters within the Transformer encoder are updated during full fine-tuning. Conversely, for PEFT, only the newly added parameters are trained.

Our experiments focus on ASR as the downstream task, adding a CTC layer on top of the HuBERT model and employing the alphabet characters as the output units. We use the LibriSpeech [29] clean-100 subset as the train set. Using the dev set, we explore the hyper-parameters, including the learning rate. We employ the word error rate (WER) as a criterion to choose the best model. Given that the optimal learning rate may vary between full fine-tuning and PEFT methods [19], we explore values within the range $[10^{-5}, 5 \times 10^{-3}]$; this search leads to a choice of 5×10^{-5} and 10^{-3} for full fine-tuning and PEFT, respectively. We use GELU [30] as a non-linear activation function in PEFT methods. For other hyper-parameters, we follow the original configurations of HuBERT [2] and use the publicly available model and configurations on the Hugging Face model hub. For mini-batching, rather than using a fixed batch size per device as prescribed in the Hugging Face trainer, we configure each minibatch to have 1600 seconds of input for every training step.

Table 2: *Comparison of word error rates (WER) when applying LoRA ($r = 8$) with or without convolution to each module in the Transformer. We evaluate the performance with different depth-wise convolution kernel sizes.*

		dev-clean	dev-other
only-MHA	no Conv	9.29	18.08
	+ k=3	8.88	17.51
	+ k=7	8.71	17.20
	+ k=15	8.80	**16.88**
	+ k=31	**8.43**	16.99
only-FF	no Conv	7.01	15.53
	+ k=3	6.77	**14.91**
	+ k=7	**6.75**	14.98
	+ k=15	6.94	14.99
	+ k=31	6.75	15.16

5. Results

In Table 1, we evaluate the performance of PEFT methods in three key aspects compared to full-model fine-tuning (Full-FT). First, we assess the efficiency of PEFT in training by examining the size of the entire model and the total number of trainable parameters. Second, we measure the total number of multiply-accumulate operations (MACs) of the fine-tuned model to investigate the increase in computational complexity due to PEFT methods. Lastly, we compare the recognition accuracy of the ASR model on the LibriSpeech dev and test sets. For a more precise analysis, in the table, we separately calculate the entire model size and computational costs for LoRA, which are measured before merging the LoRA parameters into the original dense layer. In actual use after fine-tuning, those values would match the Full-FT costs. Furthermore, we note that the CTC-based ASR-task layer, which is consistently used for all

[1] https://github.com/microsoft/DeepSpeed/

[2] https://huggingface.co/facebook/hubert-base-ls960

Table 3: *Comparison of word error rates (WER) when fine-tuning with an adapter ($r = 64$) applied to the Transformer. We evaluate with and without convolution, with various kernel sizes. We evaluate models on dev-clean (clean) and dev-other (other).*

	Adapter		Adapter+Conv							
			k=3		k=7		k=15		k=31	
	clean	other	clean	other	clean	other	clean	other	clean	other
FF-Seq	6.67	15.14	6.33	14.48	6.23	14.54	**6.18**	**14.44**	6.22	14.67
MHA-Seq	7.01	15.74	6.72	15.73	**6.62**	15.48	6.72	15.49	6.86	**15.39**
MHA-Par	8.13	18.97	7.57	18.12	7.38	17.42	7.21	17.65	**7.03**	**17.27**

experiments in this paper, is not included in the trainable parameter count or the computational costs. This provides a more accurate comparison between Full-FT and each PEFT method.

Compared to Full-FT, PEFT shows performance degradation, especially on dev-clean and test-clean. A higher rank r has the potential to improve performance; however, in this experiment, we use a rank of 16 for LoRA and 64 for adapters. For conv-augmented LoRA, we use a kernel size of 31 and 3 for convolutions in the MHA and the FF module, respectively. The addition of convolution appears to consistently improve accuracy. The performance improvements on dev-other and test-other are greater than those on dev-clean and test-clean. Although the difference is a small margin, conv-augmented LoRA performs better than Full-FT on dev-other and test-other. However, the computational cost slightly increases from 74.07 to 75.40 GMACs due to convolution and mostly LoRA parameters, which cannot be merged with the original dense layer. In the adapter part of Table 1, we employ sequential adapters for both the MHA and the FF modules, and fine-tuning with conv-augmented adapters that use a kernel size of 15 also achieves noticeable improvements.

In Table 2, we examine the effect on fine-tuning performance of using LoRA for each module, whether convolution is added or not. Additionally, we conduct experiments using different convolution kernel sizes to determine the optimal values for each module. For both modules, adding convolution to LoRA improves accuracy on the ASR task. An intriguing observation is that for the MHA module, performance continuously improves as the convolution kernel size increases. In contrast, for the FF module, there is a marginal difference, or smaller kernel sizes perform better. This aligns with our hypothesis that leveraging local information and global context together is beneficial for extracting information in the MHA module. As in other studies on Transformer variants using convolution before or after the MHA module [20], or in a parallel manner [22, 23], results show that larger kernel sizes are preferable. Conversely, in the case of the FF module, which performs point-wise projection, we hypothesize that a larger kernel size might dilute the intended information.

In Table 3, similarly to the previous experiments, we evaluate the recognition accuracy when convolution is applied to adapters. We conduct experiments on different cases where an adapter layer is sequentially added after the MHA or the FF modules, or when it is attached parallel to the MHA module. We note that, in other configurations, for example, when we attach an adapter in parallel next to the FF module, our attempts to fine-tune a model do not perform well. The results show that adding a conv-augmented adapter sequentially to the FF module improves performance as the kernel size increases. Even in the case of the MHA module, a sequentially added adapter achieves better performance with convolution than without. When we employ an adapter in parallel with the MHA module, although the base performance is worse than in other cases, surprisingly, the improvement from incorporating convolution is much more noticeable.

Table 4: *Comparison between word error rates (WER) for LoRA and LoRA with convolution, for different LoRA ranks. The convolution kernel size is 7. *Since the model does not converge well with a LoRA rank of 32, we use a lower learning rate, 5×10^{-4}, instead.*

	LoRA		LoRA+Conv	
	dev-clean	dev-other	dev-clean	dev-other
r=8	6.57	15.12	6.40	14.59
r=16	5.77	14.44	5.62	13.63
r=32	*5.52	*13.97	5.29	13.08

Table 5: *Comparison between word error rates (WER) for adapters and adapters with convolution, for different adapter bottleneck dimensions. The convolution kernel size is 7.*

	Adapter		Adapter+Conv	
	dev-clean	dev-other	dev-clean	dev-other
r=16	7.50	15.97	7.11	15.39
r=32	6.50	14.74	6.23	14.46
r=64	5.75	14.44	5.62	13.85

Finally, we test whether the performance improvement obtained by incorporating convolution in PEFT methods remains when the rank r increases. Tables 4 and 5 show the changes in accuracy when convolution is applied to adapters and LoRA with different ranks. The results show that, even in cases with a high rank, convolution contributes consistently to the performance after fine-tuning.

6. Conclusion

In this work, we have studied convolution-augmented PEFT for a pre-trained SSL model. We obtain consistent WER improvements by simply adding a convolution to the widely used adapter and LoRA PEFT methods. While this paper investigates the methods only on the ASR task, we are interested in applying our conv-augmented PEFT to other speech tasks and benchmarks. We also plan to explore how employing more expensive convolutional modules may impact performance in the context of PEFT.

7. References

[1] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[2] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[3] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[4] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "Superb: Speech processing universal performance benchmark," *Interspeech 2021*, 2021.

[5] S. Evain, H. Nguyen, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, A. Allauzen, Y. Estève, B. Lecouteux, F. Portet, S. Rossato, F. Ringeval, D. Schwab, and L. Besacier, " LeBenchmark: A Reproducible Framework for Assessing Self-Supervised Representation Learning from Speech," in *Proc. Interspeech 2021*, 2021, pp. 1439–1443.

[6] S. Shon, A. Pasad, F. Wu, P. Brusco, Y. Artzi, K. Livescu, and K. J. Han, "Slue: New benchmark tasks for spoken language understanding evaluation on natural speech," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7927–7931.

[7] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe *et al.*, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1179–1210, 2022.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[9] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[10] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*. PMLR, 2019, pp. 2790–2799.

[11] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2021.

[12] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *International Conference on Learning Representations*, 2021.

[13] K. Tomanek, V. Zayats, D. Padfield, K. Vaillancourt, and F. Biadsy, "Residual adapters for parameter-efficient asr adaptation to atypical and accented speech," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6751–6760.

[14] C.-L. Fu, Z.-C. Chen, Y.-R. Lee, and H.-Y. Lee, "Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks," in *Findings of the Association for Computational Linguistics: NAACL 2022*, 2022, pp. 2608–2621.

[15] S. Kim, H. Yang, Y. Kim, Y. Hong, and E. Park, "Hydra: Multi-head low-rank adaptation for parameter efficient fine-tuning," *arXiv preprint arXiv:2309.06922*, 2023.

[16] J. Huang, K. Ganesan, S. Maiti, Y. M. Kim, X. Chang, P. Liang, and S. Watanabe, "Findadaptnet: Find and insert adapters by learned layer importance," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[17] T.-H. Lin, H.-S. Wang, H.-Y. Weng, K.-C. Peng, Z.-C. Chen, and H.-y. Lee, "Peft for speech: Unveiling optimal placement, merging strategies, and ensemble techniques," *arXiv preprint arXiv:2401.02122*, 2024.

[18] Z. Zhong, Z. Tang, T. He, H. Fang, and C. Yuan, "Convolution meets lora: Parameter efficient finetuning for segment anything model," *arXiv preprint arXiv:2401.17868*, 2024.

[19] Z.-C. Chen, C.-L. Fu, C.-Y. Liu, S.-W. D. Li, and H.-y. Lee, "Exploring efficient-tuning methods in self-supervised speech models," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 1120–1127.

[20] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *Interspeech 2020*, 2020.

[21] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5874–5878.

[22] Y. Peng, S. Dalmia, I. Lane, and S. Watanabe, "Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 627–17 643.

[23] K. Kim, F. Wu, Y. Peng, J. Pan, P. Sridhar, K. J. Han, and S. Watanabe, "E-branchformer: Branchformer with enhanced merging for speech recognition," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 84–91.

[24] Y. Peng, K. Kim, F. Wu, B. Yan, S. Arora, W. Chen, J. Tang, S. Shon, P. Sridhar, and S. Watanabe, "A comparative study on e-branchformer vs conformer in speech recognition, translation, and understanding tasks," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2023, 2023, pp. 2208–2212.

[25] Y. Li, A. Mehrish, R. Bhardwaj, N. Majumder, B. Cheng, S. Zhao, A. Zadeh, R. Mihalcea, and S. Poria, "Evaluating parameter-efficient transfer learning approaches on sure benchmark for speech understanding," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[26] A. Bapna and O. Firat, "Simple, scalable adaptation for neural machine translation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 1538–1548.

[27] B. Thomas, S. Kessler, and S. Karout, "Efficient adapter transfer of self-supervised speech models for automatic speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7102–7106.

[28] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[30] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.