Interspeech 2025
17-21 August 2025, Rotterdam, The Netherlands

Parameter-efficient Fine-tuning of Conformer-based Streaming Speech Recognition into Non-streaming Models

Yunjae Nam[1], *Jeong U Han*[1], *Kiyeon Kim*[1], *Jaemin Lim*[1†]

[1]PrairieSchooner Inc., Pangyo, Republic of Korea
{yjnam, ipagasoos, kykim, jmlim}@prairieschooner.io

Abstract

In this paper, we propose a parameter-efficient fine-tuning method for converting streaming speech recognition models into non-streaming models. We introduce the Full-Context Adapter, a modified residual adapter designed to incorporate full-context information, enabling non-streaming conversion while keeping the pre-trained streaming model. The proposed method integrates the Full-Context Adapter into a Conformer-based RNN-Transducer streaming model, allowing it to operate as a non-streaming model. We conduct experiments to evaluate the effectiveness of our method, demonstrating that it achieves comparable performance to a non-streaming baseline model while requiring $<10\%$ additional parameters. Furthermore, we analyze optimal hyperparameter configurations and explore performance enhancement strategies. Our results show that the proposed approach provides a simple yet effective solution for efficiently operating both streaming and non-streaming ASR.

Index Terms: residual adapter, streaming speech recognition, non-streaming speech recognition [†]

1. Introduction

Recent advances in end-to-end automatic speech recognition (ASR) have significantly improved the performance of streaming models, enabling real-time transcription in various applications [1–10]. However, streaming models inherently suffer from limited context awareness, leading to inferior recognition accuracy compared to non-streaming models. Since non-streaming models incorporate full-context acoustic information, a non-streaming model is trained separately from streaming models. This approach leads to inefficiencies, as streaming and non-streaming models are maintained separately despite their similar objectives.

To address this issue, existing studies have proposed two main approaches. The first approach is the cascade encoder, which stacks a non-streaming encoder on top of the existing encoder of the streaming model [3–5, 7–9]. In this approach, the non-streaming encoder takes the entire outputs from the streaming encoder as input, enabling the model to leverage full-context audio information for generating the final recognition output. While this approach improves speech recognition accuracy by incorporating full-context information, it also introduces a significant number of additional parameters due to the inclusion of the non-streaming encoder and decoder. The second approach is the unified encoder [11, 12], which utilizes a single encoder for both streaming and non-streaming ASR. But, achieving optimal performance for both use cases requires complex optimization techniques.

In this paper, we propose a simple and parameter-efficient approach for converting streaming ASR models into non-streaming models. Our approach leverages the residual adapter [13], which is widely known for enabling parameter-efficient fine-tuning of ASR models for specific domains [14–17]. Specifically, instead of updating the weights of the pre-trained streaming encoder, we train residual adapters to convert the streaming model into a non-streaming model. However, the conventional residual adapter inherently lacks the ability to capture full-context information, making it insufficient for direct application in non-streaming fine-tuning. To address this issue, we modified the existing residual adapter by inserting a self-attention layer to integrate full-context acoustic information. We refer to this modified adapter as the Full-Context Adapter. Our method inserts the Full-Context Adapter into a pre-trained Conformer [18]-based streaming model, operating as a non-streaming model. In the experiments, we demonstrate that our approach achieves comparable performance to a non-streaming model while requiring only $<10\%$ of the parameters.

In the following sections, we first provide an overview of the conventional residual adapter in Section 2. In Section 3, we describe the Conformer-based RNN-Transducer architecture and the modifications made to integrate full-context information into residual adapters. Section 4 describes the experimental setup, including dataset, model architecture and training details. In Section 5, we analyze the optimal hyperparameters for the Full-Context Adapter and explore methods to enhance our approach to achieve performance comparable to a non-streaming baseline model. Finally, Section 6 concludes our work and discusses potential future directions.

2. Residual Adapters

Residual Adapter [13] is a lightweight, trainable component designed to adapt large pre-trained models to specific domains or tasks. Without altering the weights or architecture of the pre-trained model, the adapter learns task or domain-specific information. By combining the output of the adapter with the original model's output via residual connections, the method retains pre-trained knowledge while accommodating new task-specific learning. This approach significantly reduces memory and computation requirements by training only the adapter layers rather than the entire model. The structure of a residual adapter typically consists of the following layers arranged in sequence: a layer normalization [19], a down-projection to reduce the input feature dimensions, a non-linear activation function such as ReLU [20], and an up-projection to restore the original dimensions. The functionality of a residual adapter can be mathemat-

[†]Corresponding author

10.21437/Interspeech.2025-685

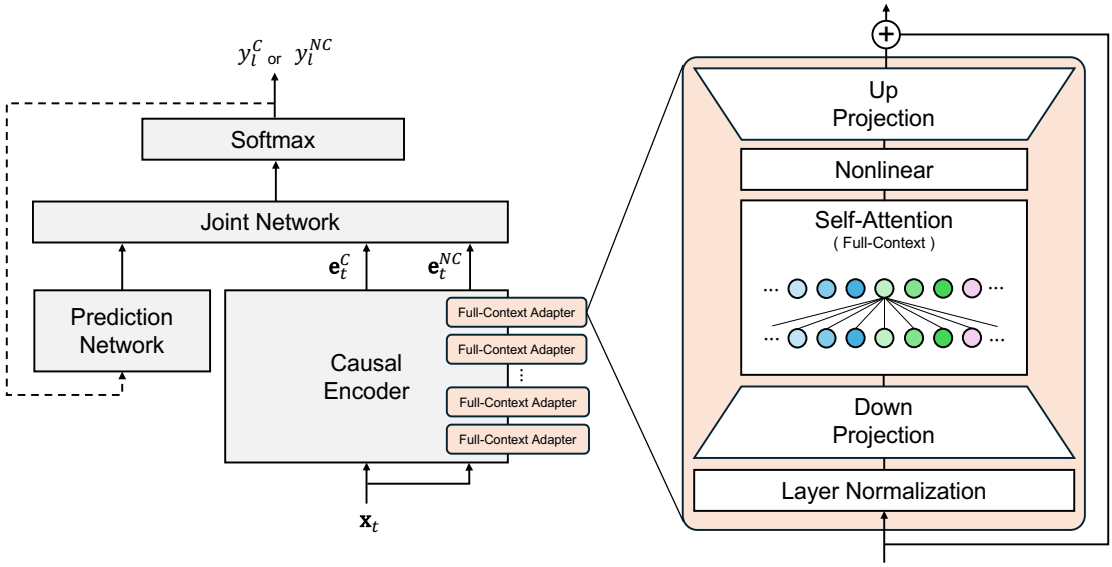


Figure 1: *Conformer-based RNN-Transducer architecture (left) and Full-Context Adapter (right).*

ically expressed as follows:

$$P_k(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W}_k + \mathbf{b}_k \quad (1)$$

$$adapter(\mathbf{x}) = \mathbf{x} + P_{up}(f(P_{down}(g(\mathbf{x})))) \quad (2)$$

where P_k is a certain projection and f, g are a non-linear activation function and the layer normalization, respectively.

The first layer of the residual adapter g, layer normalization, facilitates seamless insertion of the adapter into any part of the pre-trained network [21]. The down-projection P_{down} maximizes computational and memory efficiency by reducing the input dimensions. Especially, the size of bottleneck dimension can be adjusted depending on the complexity of the task or the size of training dataset [16, 21], and it determines additional number of parameters.

Residual adapters are typically inserted into Transformer-based architectures. Recent studies [16, 17] have extended this concept to Conformer-based architectures [18] by introducing residual adapters for speech tasks such as speech recognition and translation [15–17, 21–26]. Generally, three main methods are used for inserting residual adapter into Conformer. They are (i) *serial* that inserts residual adapters after each Conformer block; (ii) *parallel* that residually connects an adapter in parallel with the upper Feed Forward Network (FFN) of macaron-like FFNs within a Conformer block; (iii) *TPA* (Two Parallel Adapter) [16] that is similar to the *parallel* method but employs a residual connection to insert an adapter in parallel with both the lower and upper FFNs.

3. Proposed Approach

Fig. 1 shows the overall framework of our approach. In this section, we first describe speech recognition model architecture. Followed by the modification we made to the residual adapter for parameter-efficient fine-tuning of streaming models into non-streaming models.

3.1. Conformer-based RNN-Transducer

For speech recognition, we employ a recurrent neural network transducer (RNN-T) [1, 2], as shown on the left side of Fig. 1. RNN-T consists of an encoder, a prediction network (decoder) and a joint network. The encoder takes a sequence of acoustic features $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_T)$ as input, where $\mathbf{x}_t \in \mathbb{R}^d$. The prediction network takes a sequence of text tokens $\mathbf{y} = (y_1, ..., y_L)$ as input, where $y_l \in \mathbb{Z}$, the set of token indices. The joint network generates the final output by combining the outputs of the encoder and the prediction network.

In streaming applications, the RNN-T encoder is typically causal, as only past acoustic frames are observable. Consequently, the encoder is conventionally implemented using unidirectional LSTMs [2–5]. Recently, the performance of end-to-end streaming speech recognition has been improved by replacing LSTM with Conformer [6–10] which is modified to preventing the self-attention, convolution, and normalization layers from observing future context. Moreover, recent studies have proposed methods that allow the model to look ahead at a short window of future speech in addition to past inputs [7, 10, 27]. This approach enables a flexible trade-off between latency and performance, depending on the application context.

3.2. Full-Context Adapter

In our work, we use the residual adapter for fine-tuning the streaming speech recognition model into non-streaming models. However, existing residual adapter typically consists of layers that are invariant for time series. Therefore, we modify the residual adapter to incorporate acoustic full-context. To address this, as illustrated on the right side of Fig. 1, we add the self-attention layer [28] to the residual adapter*. We also maximize the parameter efficiency by inserting the self-attention layer after down-projection.

In streaming applications, the input speech features $\mathbf{x}$ pass through the causal encoder to produce causal speech embeddings $\mathbf{e}^C = (\mathbf{e}_1^C, ..., \mathbf{e}_T^C)$. In non-streaming, the speech embeddings $\mathbf{e}^{NC} = (\mathbf{e}_1^{NC}, ..., \mathbf{e}_T^{NC})$ are obtained by further passing through the Full-Context Adapter inserted into each conformer block within the causal encoder.

*Note that bidirectional LSTM underperformed compared to self-attention in training and inference speed as well as performance

4. Experiment Setup

4.1. Dataset

We train our models on the GigaSpeech dataset [29]. The set is a multi-domain English speech recognition corpus comprising high-quality labeled audio suitable for supervised training. The data is sourced from audiobooks, podcasts, and YouTube, encompassing both read and spontaneous speaking styles across a variety of topics, including arts, science, and sports. All models are trained using the *XL* training subset, which contains 10,000 hours of transcribed audio data. We also evaluate our models on the dataset's 12.5-hour *DEV* set and 40.3-hour *TEST* set. We augment the training dataset using SpecAugment [30], along with reverberation and noise addition that utilizes the MUSAN dataset [31]. All transcripts are tokenized into 4,096 word-piece [32] tokens.

4.2. Model Configuration

All models use 80-dimensional log-mel spectrograms as the input acoustic features, computed from audio with a 25ms window and a 10ms hop size. The encoder consists of 17 Conformer layers with a hidden dimension of 512. In the Conformer block, we use an 8-head attention layer, and the hidden dimension of the FFN is set to 2,048. By default, the streaming model assumes zero lookahead; however, for experiments analyzing the impact of lookahead on performance, we train streaming models with lookahead sizes of 80, 480, and 1040ms. The prediction network consists of a 2-layer LSTM with 640 units per layer, along with an embedding layer of 640 units. For the Full-Context Adapter, the bottleneck size is set to 32 by default unless experiments are conducted to determine the proper bottleneck size, and ReLU is used as the nonlinear activation function. For a non-streaming baseline model, the multi-head attention and convolution layers are configured to observe full context.

4.3. Training Setup

We optimize our models using AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and 0.001 of weight decay rate. For a learning rate scheduler, we use cosine annealing with warm-up steps. The global batch size is set to 2,048. All models are commonly trained to minimize only RNN-T loss [1]. The streaming model is trained for 100K steps, with 15K warm-up steps and a peak learning rate of 1.5e-3. The Full-Context Adapter is trained for 100K steps while keeping the pre-trained streaming model frozen, using 10K warm-up steps and a maximum learning rate of 3e-4. For non-streaming baseline comparison, we train the non-streaming model using the same setup as the streaming model. We train all models on 8×A6000 GPUs.

5. Experiment Results

5.1. Comparison between Insertion Methods

We investigate the optimal insertion method for integrating the Full-Context Adapter into the pre-trained streaming model. To this end, we compare three insertion methods—*serial, parallel*, and *TPA*—in terms of performance, measured by the word error rate (WER). In this experiment, we train only the Full-Context Adapter inserted into the encoder, while the encoder, decoder and joint network remain frozen. In the case of *TPA*, unlike the other two methods, two adapters are inserted per Conformer layer, with the bottleneck size set to 22 to maintain a similar number of parameters. As a result of the experiment in Table 1, all three methods improve over the streaming model as well as the conventional residual adapter. As expected, the conventional residual adapter provides little to no benefit for fine-tuning into the non-streaming model. In contrast, three types of the Full-Context Adapter demonstrate an absolute WER reduction of at least 1.65 compared to the baseline while using at most only 0.79% additional parameters. Among the three insertion methods, *serial* achieves an absolute WER reduction of 1.76 / 1.88 on the *DEV* / *TEST* sets even despite using the fewest additional parameters. This outperforms the other methods that insert the Full-Context Adapter in parallel to the FFN(s) within Conformer layer. Interestingly, *TPA* reports the lowest performance despite using slightly more parameters. Consequently, we confirm that parameter-efficient fine-tuning into a non-streaming model is most effective when using the *serial* insertion of the Full-Context Adapter, as it effectively increases the depth of the existing encoder.

Table 1: *Comparison between three insertion methods of the Full-Context Adapter. The number in parentheses next to the number of parameters indicates the ratio of trainable parameters relative to the streaming model.*

Methods	Num. Parameters	WER	
		DEV	*TEST*
Streaming	110M (100.0%)	10.15	10.28
conventional	0.57M (0.52%)	10.05	10.20
serial	0.65M (0.59%)	**8.39**	**8.51**
parallel	0.65M (0.59%)	8.47	8.60
TPA	0.87M (0.79%)	8.48	8.63

Table 2: *Comparison between unfrozen modules.* FCA *indicates the Full-Context Adapter.*

Unfrozen modules	Num. Parameters	WER	
		DEV	*TEST*
FCA	0.65M (0.59%)	8.39	8.51
FCA + Joint	3.27M (2.97%)	**8.02**	8.11
FCA + Pred. Net	8.19M (7.43%)	8.23	8.39
FCA + Pred. Net + Joint	10.82M (9.82%)	8.03	**8.09**

5.2. Choice of Unfrozen Modules

In previous experiments, only the Full-Context Adapter is trained while keeping the encoder, decoder, and joint network frozen. However, in addition to fine-tuning with only the Full-Context Adapter, further performance improvements are expected by unfreezing specific modules of the pre-trained streaming model. Thus, we investigate additional fine-tuning of the decoder and joint network, which typically account for less than 10% of the total parameters. To explore this, we conduct experiments comparing four training strategies: (i) adapter-only, (ii) adapter with joint network, (iii) adapter with decoder, and (iv) adapter with decoder and joint network fine-tuning. Through this analysis, we aim to determine the most effective approach for adapting a pre-trained streaming model into a non-streaming model while balancing performance gains and parameter effi-

ciency. As shown in Table 2, adapter-only fine-tuning underperforms compared to all other strategies. Notably, adapter with joint network fine-tuning achieves an absolute WER reduction of 2.13 / 2.17 on the *DEV* / *TEST* sets compared to the streaming model, while fine-tuning $< 3\%$ of the baseline parameters. This result suggests that the joint network successfully adapts to the output of the encoder fine-tuned by the Full-Context Adapter. Furthermore, fine-tuning the adapter with the joint network outperforms the adapter with decoder strategy, even achieving comparable performance to fine-tuning the adapter, decoder, and joint network together. We infer that this outcome arises because the same dataset used for training the streaming model is also used for non-streaming fine-tuning, thereby preserving the textual distribution of the source domain.

Table 3: *Comparison between different bottleneck sizes of the Full-Context Adapter.*

Bottleneck Size	**Num. Parameters**	**WER**	
		DEV	*TEST*
BS=32	3.27M (2.97%)	8.02	8.11
BS=64	4.04M (3.64%)	7.82	7.88
BS=128	5.99M (5.47%)	7.69	7.77
BS=256	10.92M (9.97%)	7.55	7.63
Non-streaming	110M (100.0%)	**6.82**	**6.93**

5.3. Choice of Bottleneck Size

We analyze the impact of the bottleneck size on performance in fine-tuning into non-streaming using the Full-Context Adapter. The experimental results are summarized in Table 3, which presents WER performance for different bottleneck sizes compared to a non-streaming baseline model. The non-streaming model consists of 110M parameters, serving as the upper bound in terms of model parameter size. In contrast, fine-tuning using the Full-Context Adapter with various bottleneck sizes (BS=32, 64, 128, and 256) have significantly fewer parameters, ranging from 3.27M (2.97%) to 10.92M (9.97%) of the full model. The non-streaming model achieves the best WER performance, with 6.82% on the development set and 6.93% on the test set. When fine-tuning with the Full-Context Adapter, increasing the bottleneck size gradually improves the WER, indicating that larger bottleneck layers can more effectively retain rich full-context information. The largest bottleneck (BS=256) model achieves the best performance among different sizes of the Full-Context Adapter (7.55 / 7.63% on *DEV* / *TEST* sets), approaching the non-streaming baseline. Overall, BS=256 emerges as a promising configuration that balances parameter-efficiency and recognition performance.

Table 4: *Comparison between different lookahead sizes.*

Lookaheds Size	**WER**	
	DEV	*TEST*
LH=0ms	7.55	7.63
LH=80ms	7.48	7.60
LH=480ms	7.37	7.48
LH=1040ms	**7.35**	**7.41**

Table 5: *Performance of the Full-Context Adapter when using multi-head attention with lookahead.*

Methods	**Num. Parameters**	**WER**	
		DEV	*TEST*
LH=0ms	10.92M (9.97%)	7.55	7.63
LH=0ms, MH=4		7.02	7.11
LH=1040ms		7.35	7.41
LH=1040ms, MH=4		**6.80**	6.94
Non-streaming	110M (100.0%)	6.82	**6.93**

5.4. Enhancing Performance

We study methods to enhance performance of the Full-Context Adapter, aiming to approach the performance of a non-streaming model without increasing parameter size. To achieve this, we first adopt a lookahead streaming encoder. Unlike a typical streaming model (zero-lookahead), the lookahead streaming encoder generates speech embeddings by utilizing not only past input audio but also future input audio with a predefined lookahead size. Thus, a causal encoder with lookahead leads to a latency in streaming applications. In contrast, in non-streaming applications where the entire audio is used as input, the lookahead encoder does not introduce any latency. The comparison between different lookahead sizes (LH=0, 80, 480 and 1040ms) is provided in Table 4. As the lookahead size increases, we observe a gradual improvement in performance. This indicates that the Full-Context Adapter benefits from speech representations obtained from the lookahead audio. Compared to the zero-lookahead, LH=1040ms achieves a WER reduction of 0.20 / 0.22 on the *DEV* / *TEST* sets. Secondly, we replace single-head self-attention with multi-head attention within the Full-Context Adapter to further enhance performance. Unlike single-head attention, multi-head attention enables the Full-Context Adapter to learn multiple patterns of acoustic features, improving both expressiveness and robustness. The dimensionality of each attention head is set to 64, and since the parameter size remains constant, the multi-head attention consists of 4 heads. We evaluate the impact of multi-head attention alone and its effect when combined with lookahead. Our results indicate a noticeable reduction in WER as shown Table 5. Introducing multi-head attention alone achieves a WER reduction of approximately 0.53 / 0.52 on the *DEV* / *TEST* sets compared to the single-head attention baseline. When using the multi-head attention in conjunction with LH=1040ms, we achieves a comparable performance with non-streaming speech recognition model.

6. Conclusion

We propose a parameter-efficient method for converting streaming ASR models into non-streaming models using the Full-Context Adapter. Our approach integrates full-context into the pre-trained streaming encoder, achieving comparable performance to non-streaming models with $<10\%$ additional parameters. Experiments show that fine-tuning the serial-inserted Full-Context Adapter with the joint network improves WER, and lookahead encoding with multi-head attention narrows the gap between streaming and non-streaming models. Future work focuses on further optimizing the Full-Context Adapter's efficiency and exploring its application in ASR domain adaptation.

7. References

[1] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[2] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[3] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu *et al.*, "Two-pass end-to-end speech recognition," *arXiv preprint arXiv:1908.10992*, 2019.

[4] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S.-y. Chang, W. Li, R. Alvarez, Z. Chen *et al.*, "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6059–6063.

[5] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6069–6073.

[6] M. Huang, J. Zhang, M. Cai, Y. Zhang, J. Yao, Y. You, Y. He, and Z. Ma, "Improving rnn transducer with normalized jointer network," *arXiv preprint arXiv:2011.01576*, 2020.

[7] A. Narayanan, T. N. Sainath, R. Pang, J. Yu, C.-C. Chiu, R. Prabhavalkar, E. Variani, and T. Strohman, "Cascaded encoders for unifying streaming and non-streaming asr," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5629–5633.

[8] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming asr," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5634–5638.

[9] J. Park, S. Jin, J. Park, S. Kim, D. Sandhyana, C. Lee, M. Han, J. Lee, S. Jung, C. Han *et al.*, "Conformer-based on-device streaming speech recognition with kd compression and two-pass architecture," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 92–99.

[10] V. Noroozi, S. Majumdar, A. Kumar, J. Balam, and B. Ginsburg, "Stateful fastconformer with cache-based inference for streaming automatic speech recognition," *arXiv preprint arXiv:2312.17279*, 2023.

[11] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode asr: Unify and improve streaming asr with full-context modeling," *arXiv preprint arXiv:2010.06030*, 2020.

[12] Z. Gao, S. Zhang, M. Lei, and I. McLoughlin, "Universal asr: Unifying streaming and non-streaming asr using a single encoder-decoder model," *arXiv preprint arXiv:2010.14099*, 2020.

[13] 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.

[14] A. Kannan, A. Datta, T. N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, and S. Lee, "Large-scale multilingual speech recognition with a streaming end-to-end model," *arXiv preprint arXiv:1909.05330*, 2019.

[15] K. Tomanek, V. Zayats, D. Padfield, K. Vaillancourt, and F. Biadsy, "Residual adapters for parameter-efficient asr adaptation to atypical and accented speech," *arXiv preprint arXiv:2109.06952*, 2021.

[16] N. Chen, I. Shafran, Y. Zhang, C.-C. Chiu, H. Soltau, J. Qin, and Y. Wu, "Efficient adapters for giant speech models," *arXiv preprint arXiv:2306.08131*, 2023.

[17] T. Rolland and A. Abad, "Shared-adapters: A novel transformer-based parameter efficient transfer learning approach for children's automatic speech recognition," *Procs. of Interspeech, Kos Island, Greece*, pp. 2370–2374, 2024.

[18] 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," *arXiv preprint arXiv:2005.08100*, 2020.

[19] J. L. Ba, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[20] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8609–8613.

[21] A. Bapna, N. Arivazhagan, and O. Firat, "Simple, scalable adaptation for neural machine translation," *arXiv preprint arXiv:1909.08478*, 2019.

[22] H. Le, J. Pino, C. Wang, J. Gu, D. Schwab, and L. Besacier, "Lightweight adapter tuning for multilingual speech translation," *arXiv preprint arXiv:2106.01463*, 2021.

[23] 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.

[24] N. M. Selvaraj, X. Guo, A. Kong, B. Shen, and A. Kot, "Adapter incremental continual learning of efficient audio spectrogram transformers," *arXiv preprint arXiv:2302.14314*, 2023.

[25] 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.

[26] K. Kim, S. Shon, Y.-T. Hsu, P. Sridhar, K. Livescu, and S. Watanabe, "Convolution-augmented parameter-efficient fine-tuning for speech recognition," in *Proc. Interspeech 2024*, 2024, pp. 2830–2834.

[27] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5904–5908.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[29] G. Chen, S. Chai, G. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang *et al.*, "Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio," *arXiv preprint arXiv:2106.06909*, 2021.

[30] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[31] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[32] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.