ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096154

FAST-U2++: FAST AND ACCURATE END-TO-END SPEECH RECOGNITION IN JOINT CTC/ATTENTION FRAMES

Chengdong Liang[1,2,3], *Xiao-Lei Zhang*[1,*], *BinBin Zhang*[2,3], *Di Wu*[2,3], *Shengqiang Li*[2,3], *Xingchen Song*[2,3], *Zhendong Peng*[2,3], *Fuping Pan*[2]

[1]School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China
[2]Horizon Robotics, Beijing, China [3]WeNet Open Source Community
liangchengdong@mail.nwpu.edu.cn, xiaolei.zhang@nwpu.edu.cn

ABSTRACT

Recently, the unified streaming and non-streaming two-pass (U2/U2++) end-to-end model for speech recognition has shown great performance in terms of streaming capability, accuracy and latency. In this paper, we present fast-U2++, an enhanced version of U2++ to further reduce *partial latency*. The core idea of fast-U2++ is to output partial results of the bottom layers in its encoder with a small chunk, while using a large chunk in the top layers of its encoder to compensate the performance degradation caused by the small chunk. Moreover, we use *knowledge distillation* method to reduce the token emission latency. We present extensive experiments on Aishell-1 dataset. Experiments and ablation studies show that compared to U2++, fast-U2++ reduces *model latency* from 320ms to 80ms, and achieves a character error rate (CER) of 5.06% with a streaming setup.

Index Terms— streaming speech recognition, fast-U2++, model latency, token emission latency

1. INTRODUCTION

In recent years, end-to-end (E2E) automatic speech recognition (ASR) received more and more attention. Compared with conventional hybrid DNN-HMM systems [1, 2], E2E models not only have a simple training and decoding process, but also have better performance than the hybrid systems. The most popular E2E models are based on the connectionist temporal classification (CTC) [3, 4], recurrent neural network transducer (RNN-T) [5, 6], and attention-based encoder-decoder (AED) [7, 8]. The hybrid CTC/attention end-to-end ASR [9] jointly minimizes the CTC and attention decoder losses, which results in faster convergence and also improves the robustness of the AED model. During the decoding, it combines the attention score and the CTC score, and performs joint decoding.

However, deploying the E2E system is not easy, and there are many practical problems to be solved [10]. The first challenge is the streaming problem. Some state-of-the-art models such as Transformer [11] and Conformer [12] could not run in streaming mode, which limits application scenarios. Second, streaming and non-streaming systems are usually developed, trained and deployed separately, and a lot of engineer efforts are required to promote E2E models to the production level.

To address the aforementioned problems, many efforts have been made. Specifically, for a unified non-streaming and streaming model, [13] proposed a dual-mode E2E for both streaming and non-streaming. It also uses knowledge distillation to train the student streaming mode from the full-context non-streaming mode. [14] proposed cascaded encoders which consists of both streaming and non-streaming encoders based on RNN-T. It first deals with input features by the streaming encoder, and then operates exclusively on the output of the streaming encoder by the non-streaming encoder. [15] proposed the dual causal/non-causal self-attention for E2E ASR. We have also proposed a unified streaming and non-streaming E2E model (U2/U2++) [16, 10], and opened a production first and production-ready E2E toolkit, named WeNet, [17, 18] for the aforementioned challenges in a simple and elegant way.

Although many efforts have been made to U2/U2++ and WeNet, low latency is always desired in real-world applications for a good customer experience. In this paper, we focus on optimizing *partial latency* of U2++, where the partial latency is defined as the time delay from the time when a user finishes speaking and the time when the first correct partial hypothesis is generated by the model. The partial latency is affected by (i) the *model latency* which is the waiting time introduced by the model structure, and (ii) the token emission latency [19]. For the U2++ model, we found that the CTC spike of the streaming model falls far behind that of the non-streaming model. This problem is mainly caused by causal convolution.

Based on the above analysis, in this paper, we propose fast-U2++ to further reduce the latency of U2++. The contributions are summarized as follows:

- First, we propose a dual-mode Conformer encoder. Compared with the conventional Conformer encoder

* Corresponding author.

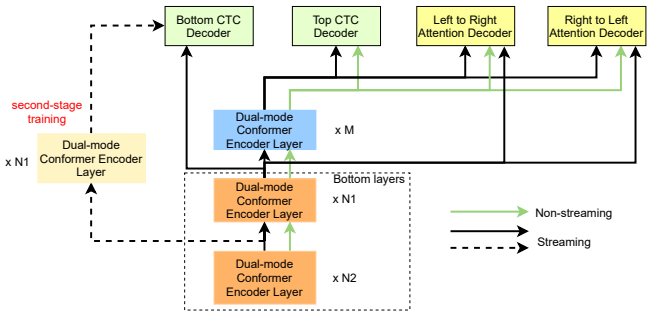

Fig. 1. Architecture of the proposed fast-U2++.

[12], we replace the conventional non-causal convolution with the dual-mode convolution. We demonstrate that the rescoring prediction of the steaming mode significantly benefits from the *joint training* of its non-streaming mode.

- Second, we use a small chunk in the bottom layers of the encoder to output partial results, which significantly reduces the *model latency*.
- Moreover, we reduce the token emission latency of the streaming mode by shifting the prediction of the non-steaming mode by *knowledge distillation* [20, 13].

With the above improvement, fast-U2++ can output partial results from the bottom layers of encoder efficiently, and can also dynamically control the trade-off between the latency and the performance, when working in the streaming mode.

2. FAST-U2++

2.1. Model architecture

The proposed model architecture is shown in Figure 1. It contains four parts: a shared dual-mode encoder that models the context of the input acoustic features, a CTC decoder that models the alignment of the frames and tokens, a Left-to-Right (L2R) attention decoder that models the dependency of the left tokens, and a Right-to-Left (R2L) attention decoder that models the dependency of the right tokens.

The shared dual-mode encoder consists of multiple dual-mode Conformer layers. The dual-mode Conformer layer uses a dual-mode convolution. The CTC decoder consists of a linear layer and a log-softmax layer. It is shared by the streaming output of the bottom layers of the encoder and the non-streaming output of the top layers of the encoder. The L2R and R2L attention decoder use the conventional Transformer decoder.

2.1.1. Dual-mode convolution

Because the convolution layer of the conventional Conformer considers both left and right context, it cannot be used in the streaming mode. To overcome this problem, as shown in Figure 2, fast-U2++ adds causal convolution [21] for the streaming mode, as a parallel branch of the conventional convolution for the non-streaming mode. Because both the causal convolution and the conventional convolution utilize the depthwise convolution structure which contains few parameters, while the other parts of the streaming and non-streaming modes are shared, the increased parameters induced by the causal convolution can be ignored.

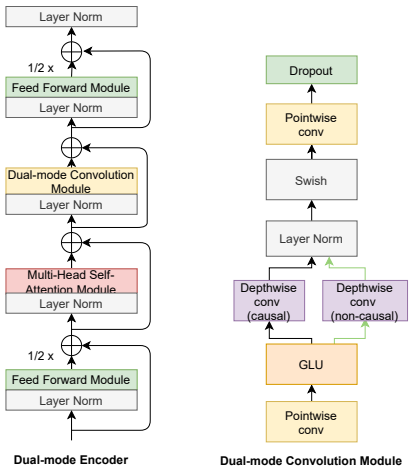

Fig. 2. Architecture of the proposed dual-mode encoder.

2.2. Training

In this section, we discuss two important training techniques: *joint training* and *knowledge distillation*.

2.2.1. Joint training

Suppose $\mathbf{x}_{\mathrm{s,l}}$, $\mathbf{x}_{\mathrm{s,h}}$ and $\mathbf{x}_{\mathrm{ns,h}}$ are the output of bottom layers in the encoder with the streaming mode, output of the top layers in the encoder with the streaming mode, and output of the top layers in encoder with the non-streaming mode, respectively. Let $\mathbf{y}$ denote the corresponding text. The training loss $\mathbf{L}_{\mathrm{joint}}(\cdot)$ for the joint training of streaming and non-streaming mode is a combination of the streaming ASR loss of the bottom layers, streaming ASR loss of the top layers, and non-streaming ASR loss of the top layers:

$$\mathbf{L}_{\mathrm{joint}} = \mathbf{L}_{\mathrm{asr}}\left(\mathbf{x}_{\mathrm{s,l}}, \mathbf{y}\right) + \mathbf{L}_{\mathrm{asr}}\left(\mathbf{x}_{\mathrm{s,h}}, \mathbf{y}\right) + \mathbf{L}_{\mathrm{asr}}\left(\mathbf{x}_{\mathrm{ns,h}}, \mathbf{y}\right) \quad (1)$$

where $\mathbf{L}_{\mathrm{asr}}(\cdot)$ is the combined CTC and AED loss for ASR:

$$\mathbf{L}_{\mathrm{asr}}\left(\mathbf{x}, \mathbf{y}\right) = \lambda \mathbf{L}_{\mathrm{CTC}}\left(\mathbf{x}, \mathbf{y}\right) + (1-\lambda)\,\mathbf{L}_{\mathrm{AED}}\left(\mathbf{x}, \mathbf{y}\right) \quad (2)$$

where λ is the weight of the CTC loss, and the AED loss $\mathbf{L}_{\mathrm{AED}}(\cdot)$ consists of a L2R AED loss $\mathbf{L}_{\mathrm{L2R}}(\cdot)$ and a R2L AED loss $\mathbf{L}_{\mathrm{R2L}}(\cdot)$:

$$\mathbf{L}_{\mathrm{AED}}\left(\mathbf{x}, \mathbf{y}\right) = (1-\alpha)\,\mathbf{L}_{\mathrm{L2R}}\left(\mathbf{x}, \mathbf{y}\right) + \alpha \mathbf{L}_{\mathrm{R2L}}\left(\mathbf{x}, \mathbf{y}\right) \quad (3)$$

with α as the weight of R2L AED loss.

2.2.2. *Knowledge distillation*

Because the prediction of the non-streaming mode usually has lower latency than the streaming mode, we can control the token emission latency of the streaming mode by shifting the prediction of the non-streaming mode in the training stage. Based on this observation, we can compute a distillation loss from $\mathbf{x}_{\mathrm{s,l}}$ and $\mathbf{x}_{\mathrm{ns,h}}$ by:

$$\mathbf{L}_{\mathrm{distill}} = \mathrm{SmoothL1Loss}\left(\mathbf{x}_{\mathrm{s,l}}, \mathbf{x}_{\mathrm{ns,h}}\right) / N_{\mathrm{frame}} \quad (4)$$

Where $\mathrm{SmoothL1Loss}(\cdot)$ is the smoothed L1 loss described in PyTorch[1], and N_{frame} is the number of speech frames.

We propose two different distillation methods:

- **D1**: Distill the last layer of the top layers in encoder and the bottom layers in encoder.
- **D2**: Distill the last two layers of the top layers in encoder and the bottom layers in encoder.

2.2.3. *Training strategy*

We propose a two-stage training strategy. In the first stage, we only conduct the joint training. In the second stage, we add $N1$ dual-mode Conformer layers in the bottom layers of the encoder as shown in Figure 1. Then, we only train the newly added Conformer layers and the corresponding CTC decoder by:

$$\mathbf{L} = \mathbf{L}_{\mathrm{joint}} + \beta \mathbf{L}_{\mathrm{distill}} \quad (5)$$

where β is the distillation weight.

2.3. Decoding

We use a similar two-pass rescoring decoding strategy with U2++. In the first pass, the bottom CTC decoder outputs the one-best candidates using greedy search decoding. At the same time, the top CTC deocder outputs n-best candidates using prefix beam search decoding.

In the second pass, the two attention decoder scores these n-best candidates. The final score is calculated by:

$$\mathbf{S}_{\mathrm{final}} = \lambda \mathbf{S}_{\mathrm{CTC}} + (1-\alpha)\,\mathbf{S}_{\mathrm{L2R}} + \alpha \mathbf{S}_{\mathrm{R2L}} \quad (6)$$

The candidate with the best score is chosen as the decoding result finally.

3. EXPERIMENTS

To evaluate the proposed fast-U2++ model, we carried out our experiments on the Chinese Mandarin speech corpus AISHELL-1[2]. We used *WeNet*[3] toolkit for all experiments.

[1]https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html
[2]http://openslr.org/33/
[3]https://github.com/wenet-e2e/wenet

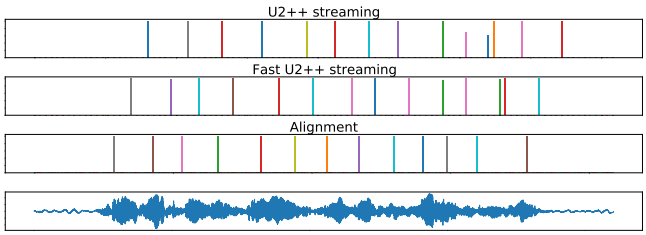

Fig. 3. Comparison of the CTC spike distributions between U2++ and the proposed fast-U2++.

We used the state-of-the-art ASR network U2++ Conformer as our baseline model. For the proposed fast-U2++, the parameter settings of the encoder and decoder were the same as U2++. The parameters $N1$, $N2$, M were set to 2, 5, 5, respectively. The input features are 80-dimensional log Mel-filter banks (FBANK) with a 25ms window and a 10ms shift. SpecAugment [22] and SpecSub [10] were applied in the same way as U2++. We set the label smoothing weight to 0.01. λ was set to 0.3 for the joint training with the Cross-entropy loss. α was set to 0.3. Adam optimizer was used, and the learning rate was warmed up with 25,000 steps. The final model averages the top-30 models with the best validation loss on the development set. The beam size of the CTC prefix search was set to 10. We used the same dynamic chunk training as U2++ for the streaming mode, which enables the model to work well in different chunk sizes. Speed perturbation with 0.9, 1.0 or 1.1 was applied on the fly.

Latency Metrics. Motivated by [19, 23], the latency metrics are *First Token emission Delay* (FTD) and *Last Token emission Delay* (LTD). FTD is defined as the timestamps difference of the following two events: (1) the first token is emitted, (2) the first token is estimated. LTD is defined as the emission latency for the final token. For all experiments, we reported both 50-th (P50) and 90-th (P90) percentile values of all utterance excluding abnormal sentences, so as to better reflect the *model latency*. We used a GMM/HMM model trained by the Kaldi[4] toolkit to get the estimated timestamps.

3.1. Main results

Table 1 reports CER, FTD and LTD of the proposed fast-U2++ in the streaming setting on the AISHELL-1 dataset. From Table 1, we see that, with the *knowledge distillation*, the CER of the CTC greedy search in the bottom layers is 7.34% when the chunk size is 4. To compensate for the performance degradation caused by small chunk size of the bottom layers, we set the chunk size of top layers to 24. Finally, the CER of rescoring is 5.06%, which is close to the performance of the U2++ with chunk size of 16. For our chunk based decoding , the average waiting time is half of the chunk theoretically [17]. Therefore, the *model latency* of fast-U2++

[4]https://github.com/kaldi-asr/kaldi

Table 1. CER, FTD and LTD of fast-U2++ on the test set of AISHELL-1. For U2++, the result of the greedy search decoding is obtained using the full 12-layers encoder. For fast-U2++, we calculate the result of the greedy search decoding with the bottom layers of its encoder. '4/24' means that the chunk size of the bottom layers in the encoder is 4, and the chunk size of the top layers in the encoder is 24. '16' means that the chunk size of the full 12-layers in the encoder is 16.

method	decoding chunk size	CER (%)		FTD (ms)		LTD (ms)	
		greedy	rescoring	P50	P90	P50	P90
U2++ baseline	16	5.81	5.05	248	300	216	250
Fast-U2++	4/24	7.47	5.08	230	280	160	210
+ D1 β=0.05		8.19		110	170	40	90
+ D1 β=0.03		7.82		140	200	50	100
+ D1 β=0.02		7.65		170	220	60	110
+ D2 β=0.05	4/24	8.27	5.06	110	170	40	80
+ D2 β=0.03		7.77		140	210	50	90
+ D2 β=0.02		7.56		160	220	50	110
+ D2 β=0.01		7.34		170	220	70	120

Table 2. CER results of fast-U2++ and the state-of-the-art streaming methods.

Method	Latency (ms)	LM	CER (%)
Sync-Transformer [24]	400	✘	8.91
SCAMA [25]	600	✘	7.39
MMA [26]	640	✘	6.60
U2 [16]	320 + Δ	✘	5.33
WNARS [27]	640	✔	5.15
U2++ [10]	320 + Δ	✘	5.05
Fast-U2++ (proposed)	80 + Δ	✘	5.06

is 80ms when chunk size of the bottom layers is 4. An example is given in Figure 3 where the CTC spike of fast-U2++ shifted significantly forward.

Moreover, we studied the effect of the weight β by setting it to 0.02, 0.03, and 0.05 respectively, together with two distillation methods, i.e. **D1** and **D2** as described in Section 2.2.2. As listed in Table 1, when β is 0.05, **D1** achieves better performance than **D2**. However, when β is getting small, **D2** tends to achieve better performance than **D1**. Both methods significantly reduce FTD and LTD, where the improvement of LTD is more apparent than FTD.

Table 2 shows the comparison results of the proposed method with other representative systems on the AISHELL-1 dataset, where Δ usually is about 100ms. From the table, we see that proposed fast-U2++ could not only have lower model latency, but also ensure good performance of final results.

3.2. Ablation study

Table 3 reports the impact of *joint training* with non-streaming loss and different CTC decoder sharing methods on performance.

Joint training. From Table 3, we see that using *joint training* with the non-streaming loss leads to better performance than that without the non-streaming loss. Moreover, according to the analysis in Section 2.1.1, the increased amount of parameters is acceptable.

Table 3. Ablation experiments without *knowledge distillation*. '7-layers' means that the result of the greedy search decoding is obtained with the bottom layers of the encoder. The number of the bottom layers in the encoder is 7 ($N1 + N2 = 7$). '12-layers' means that the result of the greedy search decoding is obtained with the full 12-layers encoder.

non-streaming loss	CTC decoder	decoding chunk size	CER (%) greedy 7-layers	CER (%) greedy 12-layers	CER (%) rescoring
✘	no shared	4/24	7.49	5.99	5.26
		16	-	5.84	5.23
✘	shared	4/24	7.46	5.82	5.34
		16	-	5.73	5.24
✔	C1	4/24	7.47	5.60	5.08
		16	-	5.53	5.00
✔	C2	4/24	7.07	5.70	5.26
		16	-	5.70	5.20
✔	C3	4/24	7.07	5.68	5.21
		16	-	5.65	5.16

CTC decoder. When using *joint training* with non-streaming, we compared three different CTC decoder sharing methods: (i) **C1**: A CTC decoder is shared by the output of the top layers in the encoder with the non-streaming mode and the output of the top layers in the encoder with the streaming mode, (ii) **C2**: A CTC decoder is shared by the output of the top layers in the encoder with the non-streaming mode and the output of the bottom layers in the encoder with the streaming mode and (iii) **C3**: all outputs of the encoder share a CTC decoder.

As listed in Table 3, the best CER after rescoring could be achieved by using the **C1**. However, the performance of the greedy search in the bottom layers of its encoder degrades slightly, compared to the other two methods.

Surprisingly, we find that using the full 12-layers of the encoder with a chunk size of 16, the CERs after the greedy search and rescoring are 5.53% and 5.00% respectively. That is to say, compared to U2++, fast-U2++ achieves a relative 5.06% (5.53% v.s. 5.81%) CER reduction using the CTC greedy search, which shows the advantage of the *joint training* with the non-streaming loss.

4. CONCLUSION

In this paper, we focus on reducing the partial latency of U2++, which leads to a new method, named fast-U2++. Fast-U2++ outputs partial results from the bottom layers of its encoder with a small chunk. It further uses *knowledge distillation* to improve the token emission latency. Experiments and ablation studies show that fast-U2++ not only significantly reduces the token emission latency, but also achieves similar performance with U2++.

5. REFERENCES

[1] Jinyu Li, Rui Zhao, Zhong Meng, Yanqing Liu, Wenning Wei, Sarangarajan Parthasarathy, Vadim Mazalov, Zhenghao Wang, Lei He, Sheng Zhao, et al., "Developing rnn-t models surpassing high-performance hybrid models with customization capability," in *Proc. Interspeech*, 2020.

[2] Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al., "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," in *ICASSP*. IEEE, 2020, pp. 6059–6063.

[3] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML*. PMLR, 2016, pp. 173–182.

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

[6] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013, pp. 6645–6649.

[7] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*. IEEE, 2016, pp. 4960–4964.

[8] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[9] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *ICASSP*. IEEE, 2017, pp. 4835–4839.

[10] Di Wu, Binbin Zhang, Chao Yang, Zhendong Peng, Wenjing Xia, Xiaoyu Chen, and Xin Lei, "U2++: Unified two-pass bidirectional end-to-end model for speech recognition," *arXiv preprint arXiv:2106.05642*, 2021.

[11] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *ICASSP*. IEEE, 2018, pp. 5884–5888.

[12] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: Convolution-augmented transformer for speech recognition," *Interspeech*, 2020.

[13] Jiahui Yu, Wei Han, Anmol Gulati, Chung-Cheng Chiu, Bo Li, Tara N Sainath, Yonghui Wu, and Ruoming Pang, "Dual-mode asr: Unify and improve streaming asr with full-context modeling," *ICLR*, 2021.

[14] Arun Narayanan, Tara N Sainath, Ruoming Pang, Jiahui Yu, Chung-Cheng Chiu, Rohit Prabhavalkar, Ehsan Variani, and Trevor Strohman, "Cascaded encoders for unifying streaming and non-streaming asr," in *ICASSP*. IEEE, 2021, pp. 5629–5633.

[15] Niko Moritz, Takaaki Hori, and Jonathan Le Roux, "Dual causal/non-causal self-attention for streaming end-to-end speech recognition," *Interspeech*, 2021.

[16] Binbin Zhang, Di Wu, Zhuoyuan Yao, Xiong Wang, Fan Yu, Chao Yang, Liyong Guo, Yaguang Hu, Lei Xie, and Xin Lei, "Unified streaming and non-streaming two-pass end-to-end model for speech recognition," *arXiv preprint arXiv:2012.05481*, 2020.

[17] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," in *Proc. Interspeech*, Brno, Czech Republic, 2021, IEEE.

[18] Binbin Zhang, Di Wu, Zhendong Peng, Xingchen Song, Zhuoyuan Yao, Hang Lv, Lei Xie, Chao Yang, Fuping Pan, and Jianwei Niu, "Wenet 2.0: More productive end-to-end speech recognition toolkit," *Interspeech*, 2022.

[19] Yuan Shangguan, Rohit Prabhavalkar, Hang Su, Jay Mahadeokar, Yangyang Shi, Jiatong Zhou, Chunyang Wu, Duc Le, Ozlem Kalinli, Christian Fuegen, et al., "Dissecting user-perceived latency of on-device e2e speech recognition," *Interspeech*, 2021.

[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[21] Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[22] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech*, 2019.

[23] Shuo-Yiin Chang, Bo Li, David Rybach, Yanzhang He, Wei Li, Tara N Sainath, and Trevor Strohman, "Low latency speech recognition using end-to-end prefetching.," in *Interspeech*, 2020, pp. 1962–1966.

[24] Zhengkun Tian, Jiangyan Yi, Ye Bai, Jianhua Tao, Shuai Zhang, and Zhengqi Wen, "Synchronous transformers for end-to-end speech recognition," in *ICASSP*. IEEE, 2020, pp. 7884–7888.

[25] Shiliang Zhang, Zhifu Gao, Haoneng Luo, Ming Lei, Jie Gao, Zhijie Yan, and Lei Xie, "Streaming chunk-aware multihead attention for online end-to-end speech recognition," *Interspeech*, 2020.

[26] Hirofumi Inaguma, Masato Mimura, and Tatsuya Kawahara, "Enhancing monotonic multihead attention for streaming asr," *Interspeech*, 2020.

[27] Zhichao Wang, Wenwen Yang, Pan Zhou, and Wei Chen, "Wnars: Wfst based non-autoregressive streaming end-to-end speech recognition," *arXiv preprint arXiv:2104.03587*, 2021.