



Delayed-KD: Delayed Knowledge Distillation based CTC for Low-Latency Streaming ASR

Longhao Li¹, Yangze Li¹, Hongfei Xue¹, Jie Liu², Shuai Fang², Kai Wang², Lei Xie^{1*}

¹Audio, Speech and Language Processing Group (ASLP@NPU),
Northwestern Polytechnical University, China
²Huawei Cloud, China

lhli@mail.nwpu.edu.cn, lxie@nwpu.edu.cn

Abstract

CTC-based streaming ASR has gained significant attention in real-world applications but faces two main challenges: accuracy degradation in small chunks and token emission latency. To mitigate these challenges, we propose Delayed-KD, which applies delayed knowledge distillation on CTC posterior probabilities from a non-streaming to a streaming model. Specifically, with a tiny chunk size, we introduce a Temporal Alignment Buffer (TAB) that defines a relative delay range compared to the non-streaming teacher model to align CTC outputs and mitigate non-blank token mismatches. Additionally, TAB enables fine-grained control over token emission delay. Experiments on 178-hour AISHELL-1 and 10,000-hour WenetSpeech Mandarin datasets show consistent superiority of Delayed-KD. Impressively, Delayed-KD at 40 ms latency achieves a lower character error rate (CER) of 5.42% on AISHELL-1, comparable to the competitive U2++ model running at 320 ms latency.

Index Terms: streaming speech recognition, token emission delay, knowledge distillation, Temporal Alignment Buffer

1. Introduction

Streaming automatic speech recognition (ASR) has attracted significant attention in real-world applications, aiming to ensure recognition accuracy with low latency. Currently, the predominant end-to-end ASR models include Connectionist Temporal Classification (CTC) [1], Recurrent Neural Network Transducer (RNN-T) [2] and Attention-based Encoder-Decoder (AED) [3, 4]. To facilitate streaming capabilities in the encoder of these end-to-end models, chunk-based methodologies are typically employed as a simple-yet-effective approach. Particularly, CTC is often favored for integration due to its architectural simplicity.

However, CTC-based streaming ASR faces two main challenges: accuracy degradation in small chunks [5] due to limited context [6] and the inherent non-blank token emission latency issue [7], where CTC spikes lag behind non-streaming models. To address the issue of accuracy degradation in small chunks, several methods have been proposed to mitigate chunk-induced losses. For instance, Fast-U2++ [8] employs a hybrid chunk strategy, utilizing small chunks in lower encoder layers for early partial result output and large chunks in upper layers to compensate for performance degradation. USIDE-T [9] introduces SimuNet, a history-frame-based simulation network, to approximate full-context information and enable smaller chunk sizes. While these strategies partially offset the performance decline caused by smaller chunks, they still underperform under tiny chunk conditions. To address the non-blank token emission la-

tency issue, techniques like FastEmit [10], Peak-first CTC [11], and Delay-Penalty [12] encourage early non-blank token emission in the loss function. Although these methods effectively reduce CTC spike lag, they still suffer from significant model latency. Since token emission occurs at the end time of the chunk, this means that while the CTC spike lag is reduced, the token emission lag remains unaddressed. Additionally, these techniques often offer limited accuracy improvements.

Streaming ASR often suffers from a performance gap compared to non-streaming ASR due to limited context [13]. Knowledge distillation [14, 15, 16] has demonstrated its effectiveness in reducing this gap and enabling early token emission [8]. However, the inherent token emission delay in streaming CTC training limits the effectiveness of direct frame-by-frame distillation [17, 18]. Previous studies [19, 20, 21] have primarily focused on explicit forced alignment with CTC for distillation but overlook the information discrepancy between streaming and non-streaming models, leading to limited effectiveness of the forced alignment learning. This highlights the need for a low-latency and efficient CTC alignment distillation method that can effectively balance accuracy and latency.

In this paper, we propose Delayed-KD to improve recognition accuracy in low-latency streaming scenarios. Delayed-KD introduces a Temporal Alignment Buffer (TAB) with a tiny chunk size during training, enabling delayed knowledge distillation of CTC posterior probabilities from a non-streaming teacher model to a streaming student model. Specifically, TAB introduces a controlled delay range, allowing distillation within this range to select the chunk with minimal KL divergence loss, effectively overcoming the limitations of frame-by-frame distillation. Experimental results on the AISHELL-1 [22] Mandarin dataset demonstrate that Delayed-KD achieves a 9.4% relative CER reduction compared to another competitive solution U2++ [23] at 40 ms latency. Notably, Delayed-KD at 40 ms latency achieves comparable performance to U2++ at 320 ms latency. Furthermore, Delayed-KD demonstrates superior low-latency and high-accuracy capabilities compared to other advanced streaming models. On the large-scale, multi-domain Mandarin WenetSpeech [24] dataset, in rescoring mode at 40 ms latency, Delayed-KD achieves a 27.9% and 33.1% relative CER reduction on *Test_Meeting* set and *Test_Net* set, respectively. Additionally, TAB adjustment allows fine-grained control over the trade-off between token emission delay and accuracy.

2. Method

2.1. Model architecture

As shown in Figure 1, the proposed model architecture, comprises three main components: the CTC branch of the non-

*Corresponding author.

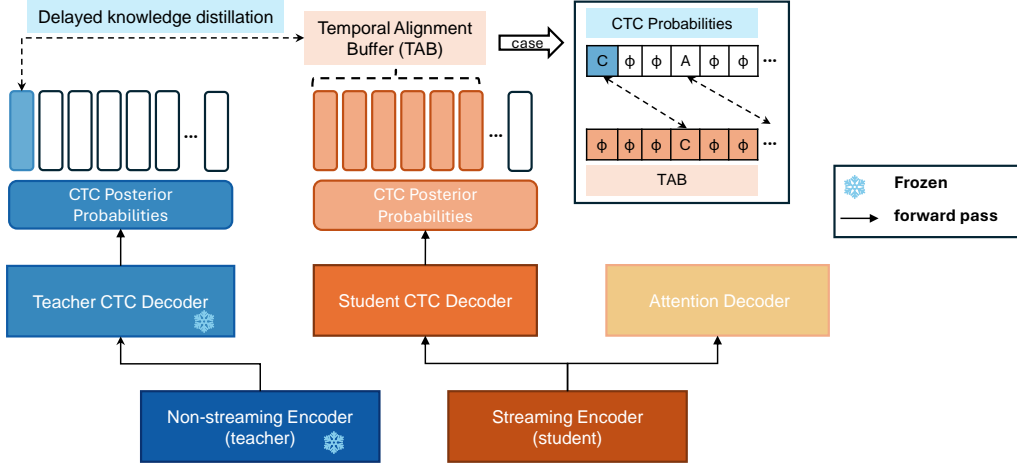


Figure 1: Model architecture of our proposed Delayed-KD

streaming teacher model, the CTC branch of the streaming student model, and the attention branch of the streaming student model. Notably, the CTC branches of both the teacher and student models share an identical structure, each consisting of a shared encoder for modeling the context of acoustic features and a CTC decoder for aligning frames and tokens. The attention branch of the student model incorporates an Attention Decoder to model dependencies among tokens.

The shared encoder is constructed with multiple Conformer [25] layers, while the CTC decoder is composed of a linear layer followed by a log softmax layer. This design serves a dual purpose: firstly, it applies the CTC loss function to the softmax output during training, and secondly, it facilitates delayed knowledge distillation of frame-level CTC posterior probabilities between the teacher and student models. The Attention Decoder adopts the same architecture as traditional Transformer decoders, ensuring robust modeling of token dependencies.

2.2. Training

In this section, we mainly discuss two training details: delayed knowledge distillation and joint training.

2.2.1. Delayed Knowledge Distillation

As shown in Figure 1, the fbank features of the audio are separately fed into the encoder of the non-streaming teacher model’s CTC branch and the encoder of the streaming student model’s CTC branch. This process generates two distinct frame-level CTC posterior probability distributions: one from the non-streaming model, which incorporates global context, and the other from the streaming model, which is limited to left-side context.

During the distillation process, we employ Kullback-Leibler (KL) divergence as the distillation metric. To address the latency of CTC spikes in the streaming model compared to the non-streaming model, we introduce a **Temporal Alignment Buffer (TAB)** to mitigate misalignment issues during distillation. Specifically, during training, we adopt a tiny chunk size. TAB maintains a controlled delay relative to the non-streaming teacher model, enabling distillation to be performed within this delay range to select the chunk with the minimum KL divergence loss. The delay range is dynamically determined

by a predefined range, which allows the selection of an optimal number of chunks to account for potential delays between the non-streaming teacher model and the streaming student model. Within this range, we calculate the KL divergence across various time delays based on chunks and select the delay that minimizes the loss as the final KL loss. By aligning the teacher model’s CTC output with the student model’s delayed CTC outputs (shifted right by a few chunks), we effectively mitigate mismatches in non-blank tokens during distillation. This approach significantly enhances the performance of the distillation process.

Based on the above process, we define the delayed knowledge distillation loss as follows:

$$L_{\text{distill}} = \frac{1}{BT} \sum_{b=1}^B \sum_{t=1}^T \min_{\tau \in \mathcal{D}} \sum_{c=1}^C \text{KL}(p_{\text{student}}^{(b,t+\tau,c)} \parallel p_{\text{teacher}}^{(b,t,c)}) \quad (1)$$

where B is the batch size, T is the number of time steps, C is the vocabulary size, and \mathcal{D} is the delayed knowledge distillation window range, which corresponds to the TAB. Specifically, $\mathcal{D} = \{0, 1, 2, \dots, d\}$ defines the allowable delay range for aligning the student and teacher model outputs, where d is the maximum delay determined by the TAB size.

The KL divergence is used to align the CTC posterior probability distributions of the student and teacher models, and it is calculated as follows:

$$\text{KL}(p_{\text{student}} \parallel p_{\text{teacher}}) = \sum_i p_{\text{student}}(i) \cdot \log \frac{p_{\text{student}}(i)}{p_{\text{teacher}}(i)} \quad (2)$$

where p represents the posterior probability distribution of the CTC output from either the teacher or student model.

2.2.2. Joint Training

The overall loss function L_{joint} is a weighted combination of the ASR task loss L_{asr} and the delayed knowledge distillation loss L_{distill} as listed in the Equation 1, defined as:

$$L_{\text{joint}} = L_{\text{asr}} + \alpha L_{\text{distill}} \quad (3)$$

where α is a hyper-parameter that controls the contribution of the delayed knowledge distillation loss to the total loss. The ASR task loss L_{asr} is further decomposed into a weighted sum of the CTC loss L_{CTC} and the attention-based encoder-decoder (AED) loss L_{AED} :

$$L_{\text{asr}}(\mathbf{x}, \mathbf{y}) = \lambda L_{\text{CTC}}(\mathbf{x}, \mathbf{y}) + (1 - \lambda)L_{\text{AED}}(\mathbf{x}, \mathbf{y}) \quad (4)$$

where λ balances the contributions of the CTC and AED losses, \mathbf{x} represents the input features, and \mathbf{y} denotes the target sequence.

2.3. Decoding

We use the same two-pass rescoring decoding strategy as in U2++ [23]. In the first pass, the frame-synchronous CTC decoder generates a set of candidate hypotheses using prefix beam search. During the second pass, the attention decoder computes the scores for these n-best candidates. The final score for each candidate is then determined by combining these scores according to Equation 5. The candidate with the highest final score is selected as the optimal output.

$$S_{\text{final}} = \lambda S_{\text{CTC}} + S_{\text{AED}} \quad (5)$$

3. Experiments

3.1. Dataset

We conduct experiments on two Mandarin Chinese datasets: AISHELL-1 [22] (178 hours) and the large-scale, multi-domain WenetSpeech [24] (10,000 hours). For AISHELL-1, the test set contains 7,176 utterances, while WenetSpeech provides two test sets, *Test_Meeting* and *Test_Net*, which collectively contain approximately 33,100 utterances.

3.2. Experimental Setup

All experiments in this study are implemented and evaluated within the WeNet toolkit [26]. We use the standard Conformer U2++ model as our baseline and conduct experiments based on its framework. The parameters of the teacher non-streaming model are frozen during the training process. The hyper-parameters for both the encoder and decoder in Delayed-KD are consistent with those of U2++ to ensure a fair comparison.

The input features are 80-dimensional fbank features, with a frameshift of 10 ms and a frame length of 25 ms. We also implemented SpecAugment and SpecSub in the same way as U2++. The modelling units consist of approximately 4,000 Chinese characters in the AISHELL-1 dataset and approximately 5,500 in the WenetSpeech experiments.

The label smoothing weight is configured to 0.01. For joint training with the cross-entropy loss, λ is assigned a value of 0.3. The Adam optimizer, combined with a Warmup learning rate scheduler, is employed for all models. The initial learning rate is set to 0.001, and gradient clipping is applied with a threshold of 5.0. The learning rate undergoes a warm-up phase over the first 25k steps. The total training steps are 200k for AISHELL-1 and 130k for WenetSpeech. During the decoding phase, for AISHELL-1, we average the top 20 models based on the lowest dev loss, while for WenetSpeech, the top 10 models are averaged.

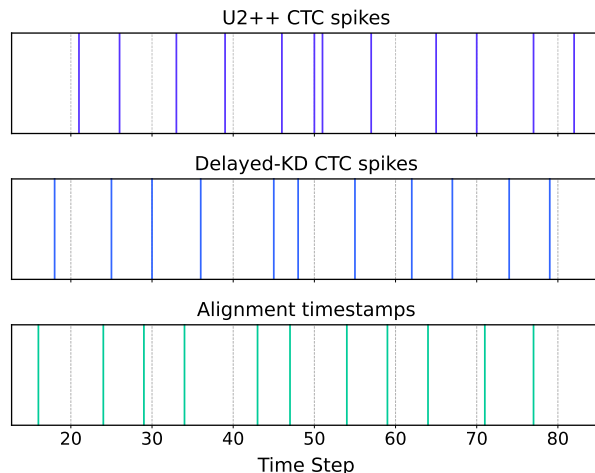


Figure 2: Comparison of CTC spike distributions between U2++ and Delayed-KD. Colored lines represent CTC spikes, and dashed lines align time axis positions.

3.3. Emission Latency Metrics

To evaluate character-level recognition latency, we calculate each token’s latency by subtracting the corresponding token’s ground-truth end time from the model output timestamp. For chunk-based streaming models, each token’s timestamp corresponds to the end time of the chunk. During inference, the model records the timestamps of all output tokens, while the character boundaries in the audio signals are extracted using the Forced Aligner tool as ground truth. Motivated by [27, 28], we measure two key metrics: (1) First Token Delay (FTD), the emission latency of the first token in each utterance, and (2) Last Token Delay (LTD), the emission latency of the last token in each utterance. To mitigate the impact of outliers, we report the 50th percentile (P50) and the 90th percentile (P90) of all utterances, excluding abnormal sentences. The estimated timestamps are obtained using a GMM/HMM model trained with the Kaldi toolkit [29].

3.4. Results on AISHELL-1

Table 1 presents the results on AISHELL-1, including latency, CER, and token emission delay, specifically FTD and LTD. Additionally, for Delayed-KD, we compare the impact of different Temporal Alignment Buffer (TAB) settings on CER and token emission delay.

Performance superiority of Delayed-KD. Delayed-KD achieves an enhanced optimization of accuracy and latency on AISHELL-1 through its TAB mechanism and delayed knowledge distillation approach. As shown in Table 1, in rescoring decoding mode, Delayed-KD achieves a CER of 5.42% at 40 ms latency, even comparable to another competitive solution U2++ running at 320 ms latency. While at the same 40 ms latency, Delayed-KD delivers a 9.34% relative CER reduction. In streaming decoding mode, Delayed-KD demonstrates enhanced frame-level decoding capabilities, achieving a 21.26% relative CER reduction over U2++ at 40 ms latency and even surpassing U2++’s performance at 320 ms latency. Through dynamic TAB size adjustment, Delayed-KD significantly reduces both FTD and LTD compared to U2++ across 320 ms and 40 ms latency settings, effectively minimizing token emission delay. As

Table 1: Comparison of CER, FTD, and LTD results for our proposed Delayed-KD with different Temporal Alignment Buffer (TAB) settings and other baseline models on AISHELL-1 test set. TAB value of 0 represents frame-by-frame distillation. Streaming refers to frame-level streaming decoding while rescoring represents an utterance-level re-scoring decoding.

Model	Temporal Alignment Buffer (TAB)	Decoding chunk size	Latency (ms)	CER (%)		FTD (ms)		LTD (ms)	
				Streaming	Rescoring	P50	P90	P50	P90
U2++ [23]	-	8	320	6.53	5.44	360	410	310	340
		1	40	7.76	5.98	170	210	120	150
Fast-U2++ [8]	-	4/24	160/960	7.34	5.06	170	220	70	120
CUSIDE-T [9]	-	10	400	6.02	5.51	-	-	-	-
Delayed-KD	0 ms			7.02	5.77	100	140	50	90
	40 ms			6.67	5.65	120	160	70	110
	80 ms	1	40	6.11	5.42	140	180	90	130
	120 ms			6.14	5.47	160	200	110	150
	160 ms			6.15	5.48	180	220	130	160

visually illustrated in Figure 2, the CTC spikes of Delayed-KD occur notably earlier than those of U2++, providing clear evidence of its improved alignment and reduced token emission latency.

Delayed-KD demonstrates superior low-latency and high-accuracy capabilities compared to other advanced streaming models. Fast-U2++ uses a hybrid block strategy (4/24 for streaming/rescoring), while CUSIDE-T operates at a minimum latency of 400 ms. In rescoring decoding mode, Delayed-KD at 40 ms latency outperforms CUSIDE-T at 400 ms latency, while its accuracy is lower than that of Fast-U2++, which is expected given the latter’s 24 times higher latency. In streaming decoding mode, Delayed-KD at 40 ms surpasses Fast-U2++ at 160 ms and approaches CUSIDE-T’s performance, demonstrating its ability to balance latency and accuracy effectively.

Analysis of TAB Settings and Trade-offs. We also investigate the impact of different TAB sizes on CER and token emission latency. As shown in Table 1, direct frame-to-frame distillation achieves the lowest token emission latency but sub-optimal CER. Increasing the TAB size improves CER significantly, with the 80 ms TAB setting yielding the best results: rescoring CER of 5.42% and streaming CER of 6.11%. This corresponds to a relative CER reduction of 6.07% and 12.96%, respectively, compared to frame-to-frame distillation. Although larger TAB sizes increase emission latency, including FTD and LTD, Delayed-KD maintains accuracy advantages while keeping latency lower than U2++, effectively balancing accuracy and latency.

Table 2: Streaming ASR results in streaming and rescoring modes across varying distillation weight (α)

Distillation Weight (α)	CER (%)	
	Streaming	Rescoring
4	6.54	5.62
20	6.35	5.56
100	6.11	5.42
200	7.10	5.87
400	7.99	6.30

Analysis of Distillation Weight Configurations. The distillation weight plays a critical role in joint training, balancing delayed distillation loss and total loss. As shown in Table 2, too low a weight weakens the effectiveness of knowledge dis-

tillation, while weights exceeding 100 degrade performance in both Streaming and rescoring modes. We hypothesize that this decline is primarily due to overfitting to the non-streaming CTC posterior probability distributions, negatively impacting generalization.

3.5. Results on WenetSpeech

We further conduct a comparative evaluation of Delayed-KD and U2++ on the large-scale WenetSpeech dataset. Using the optimal hyper-parameter settings from the AISHELL-1 experiments, we perform experiments on the *Test_Meeting* and *Test_Net* test sets of WenetSpeech. As shown in Table 3, consistent with the conclusions from the AISHELL-1 experiments, Delayed-KD achieves comparable results at 40 ms latency to U2++ at 320 ms latency, and performs significantly better under the same latency conditions. One expected observation is that Delayed-KD is particularly effective in improving the accuracy of the streaming mode, yielding better results than the rescoring mode. These results indicate that Delayed-KD maintains robust performance even on large-scale datasets.

Table 3: Streaming ASR results on WenetSpeech test set (*Test_Meeting*, *Test_Net*). CER results are reported as streaming / rescoring.

Model	Decoding chunk size	Latency (ms)	CER(%)	
			Test_Meeting	Test_Net
U2++	8	320	17.80 / 16.76	11.54 / 12.23
	1	40	25.32 / 23.52	16.51 / 19.06
Delayed-KD	1	40	17.53 / 16.96	11.37 / 12.75

4. Conclusion

In this paper, we propose Delayed-KD, a novel delayed knowledge distillation method that distills the CTC posterior probabilities from a non-streaming teacher model. By introducing a Temporal Alignment Buffer (TAB) during training, Delayed-KD aligns the CTC outputs of streaming and non-streaming models, effectively reducing mismatches in the distillation process. Experimental results on the AISHELL-1 and WenetSpeech datasets demonstrate that our method achieves an optimal balance between latency and accuracy.

5. References

- [1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proc. ICML*, 2006, pp. 369–376.
- [2] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.
- [3] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, “Listen, attend and spell,” *arXiv preprint arXiv:1508.01211*, 2015.
- [4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” *Advances in neural information processing systems*, vol. 28, 2015.
- [5] H. Tang, Y. Fu, L. Sun, J. Xue, D. Liu, Y. Li, Z. Ma, M. Wu, J. Pan, G. Wan *et al.*, “Reducing the gap between streaming and non-streaming transducer-based asr by adaptive two-stage knowledge distillation,” in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.
- [6] S. Kumar, S. Madikeri, J. Zuluaga-Gomez, E. Villatoro-Tello, I. Thorbecke, P. Motlicek, A. Ganapathiraju *et al.*, “Xlsr-transducer: Streaming asr for self-supervised pretrained models,” *arXiv preprint arXiv:2407.04439*, 2024.
- [7] Y. Fang and X. Li, “Mamba for streaming asr combined with unimodal aggregation,” *arXiv preprint arXiv:2410.00070*, 2024.
- [8] C. Liang, X.-L. Zhang, B. Zhang, D. Wu, S. Li, X. Song, Z. Peng, and F. Pan, “Fast-u2++: Fast and accurate end-to-end speech recognition in joint ctc/attention frames,” in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.
- [9] W. Zhao, Z. Li, C. Yu, and Z. Ou, “Cuside-t: Chunking, simulating future and decoding for transducer based streaming asr,” in *Proc. ICSLSP*. IEEE, 2024, pp. 11–15.
- [10] J. Yu, C.-C. Chiu, B. Li, S.-y. Chang, T. N. Sainath, Y. He, A. Narayanan, W. Han, A. Gulati, Y. Wu *et al.*, “Fastemit: Low-latency streaming asr with sequence-level emission regularization,” in *Proc. ICASSP*. IEEE, 2021, pp. 6004–6008.
- [11] Z. Tian, H. Xiang, M. Li, F. Lin, K. Ding, and G. Wan, “Peak-first ctc: reducing the peak latency of ctc models by applying peak-first regularization,” in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.
- [12] W. Kang, Z. Yao, F. Kuang, L. Guo, X. Yang, L. Lin, P. Želasko, and D. Povey, “Delay-penalized transducer for low-latency streaming asr,” in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.
- [13] A. Kojima, H. Hermansky, H. Cernocký, L. Burget, L. Lamel, and O. Scharenborg, “Knowledge distillation for streaming transformer-transducer,” in *Interspeech*, 2021, pp. 2841–2845.
- [14] G. Hinton, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [15] K. Shim, J. Lee, S. Chang, and K. Hwang, “Knowledge distillation from non-streaming to streaming asr encoder using auxiliary non-streaming layer,” *arXiv preprint arXiv:2308.16415*, 2023.
- [16] A. Seth, S. Ghosh, S. Umesh, and D. Manocha, “Stable distillation: Regularizing continued pre-training for low-resource automatic speech recognition,” in *Proc. ICASSP*. IEEE, 2024, pp. 10 821–10 825.
- [17] 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.
- [18] H. Inaguma and T. Kawahara, “Alignment knowledge distillation for online streaming attention-based speech recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1371–1385, 2021.
- [19] G. Kurata and K. Audhkhasi, “Guiding ctc posterior spike timings for improved posterior fusion and knowledge distillation,” *arXiv preprint arXiv:1904.08311*, 2019.
- [20] J. Tian, B. Yan, J. Yu, C. Weng, D. Yu, and S. Watanabe, “Bayes risk ctc: Controllable ctc alignment in sequence-to-sequence tasks,” *arXiv preprint arXiv:2210.07499*, 2022.
- [21] E. Kim, H. Kim, and K. Lee, “Guiding frame-level ctc alignments using self-knowledge distillation,” *arXiv preprint arXiv:2406.07909*, 2024.
- [22] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, “Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline,” in *2017 20th conference of the oriental chapter of the international coordinating committee on speech databases and speech I/O systems and assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.
- [23] D. Wu, B. Zhang, C. Yang, Z. Peng, W. Xia, X. Chen, and X. Lei, “U2++: Unified two-pass bidirectional end-to-end model for speech recognition,” *arXiv preprint arXiv:2106.05642*, 2021.
- [24] B. Zhang, H. Lv, P. Guo, Q. Shao, C. Yang, L. Xie, X. Xu, H. Bu, X. Chen, C. Zeng *et al.*, “Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition,” in *Proc. ICASSP*. IEEE, 2022, pp. 6182–6186.
- [25] 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.
- [26] B. Zhang, D. Wu, Z. Peng, X. Song, Z. Yao, H. Lv, L. Xie, C. Yang, F. Pan, and J. Niu, “Wenet 2.0: More productive end-to-end speech recognition toolkit,” *arXiv preprint arXiv:2203.15455*, 2022.
- [27] S.-Y. Chang, B. Li, D. Rybach, Y. He, W. Li, T. N. Sainath, and T. Strohman, “Low latency speech recognition using end-to-end prefetching,” in *Interspeech*, 2020, pp. 1962–1966.
- [28] Y. Shangguan, R. Prabhavalkar, H. Su, J. Mahadeokar, Y. Shi, J. Zhou, C. Wu, D. Le, O. Kalinli, C. Fuegen *et al.*, “Dissecting user-perceived latency of on-device e2e speech recognition,” *arXiv preprint arXiv:2104.02207*, 2021.
- [29] M. Ravanelli, T. Parcollet, and Y. Bengio, “The pytorch-kaldi speech recognition toolkit,” in *Proc. ICASSP*. IEEE, 2019, pp. 6465–6469.