INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Delay-penalized CTC implemented based on Finite State Transducer

Zengwei Yao[*], *Wei Kang*[*], *Fangjun Kuang, Liyong Guo,*
Xiaoyu Yang, Yifan Yang, Long Lin, Daniel Povey,

Xiaomi Corp., Beijing, China
{yaozengwei, kangwei1, dpovey}@xiaomi.com

Abstract

Connectionist Temporal Classification (CTC) suffers from the latency problem when applied to streaming models. We argue that in CTC lattice, the alignments that can access more future context are preferred during training, thereby leading to higher symbol delay. In this work we propose the delay-penalized CTC which is augmented with latency penalty regularization. We devise a flexible and efficient implementation based on the differentiable Finite State Transducer (FST). Specifically, by attaching a binary attribute to CTC topology, we can locate the frames that firstly emit non-blank tokens on the resulting CTC lattice, and add the frame offsets to the log-probabilities. Experimental results demonstrate the effectiveness of our proposed delay-penalized CTC, which is able to balance the delay-accuracy trade-off. Furthermore, combining the delay-penalized transducer enables the CTC model to achieve better performance and lower latency. Our work is open-sourced and publicly available[1].

Index Terms: speech recognition, delay-penalized CTC, differentiable FST, streaming, low latency

1. Introduction

Connectionist Temporal Classification (CTC) [1] is still a competitive choice among various end-to-end models [2–7] in Automatic Speech Recognition (ASR) due to its structural simplicity and computational efficiency. CTC aims to maximize the total log-probability over all enumerated alignments between feature sequence and label sequence. However, since CTC equally treats all alignments, it suffers from the symbol delay problem for training streaming ASR models [8]. It learns to strengthen the higher-score alignments that access more context, which results in high latency when deployed to real applications.

To address the CTC latency problem, a typical solution is to impose restrictions on the learned alignments by leveraging the ground-truth alignments [9–11]. The crucial limitation of this type of methods is that it needs to pre-compute the token-time alignments, which makes the end-to-end training procedure more redundant. Recently, some researches explore reducing the symbol delay without the need of external alignments [8, 12, 13]. As a prominent example, Bayes risk CTC [8] divides all alignments into different groups according to their specific symbol delay, and assigns higher risk values for those lower-delay groups during training. Our method can be explained as a form of Bayes risk CTC, but implemented in a simpler manner based on differentiable Finite State Transducer (FST), instead of modifying the CTC forward-backward algorithm [1] as in Bayes risk CTC [8].

Our work is inspired by delay-penalized transducer [14], which adds a small constant λ times the frame offsets to the log-probabilities of emitting symbols, functioning as reducing the averaged symbol delay of transducer lattice. In this paper, we propose a similar method of delay penalization for CTC, which is implemented based on the differentiable FST [15–17]. Specifically, we first attach a binary attribute to the CTC topology [17] indicating whether the arcs are with non-epsilon output labels. The propagated binary attribute on the resulting CTC lattice enables us to easily locate the frames that firstly emit non-blank tokens, so as to add the frame offsets to the log-probabilities. As a result, we can optimize the CTC objective function with latency penalty regularization by maximizing the total score of the modified CTC lattice. Experimental results validate the effectiveness of the proposed delay-penalized CTC implemented based on FST, which provides a way to adjust the trade-off between latency and performance by tuning λ.

Meanwhile, we also explore leveraging the delay-penalized transducer [14] as an auxiliary training task to achieve a better delay-accuracy trade-off for the CTC model. Experimentally, the CTC modeling capacity can be significantly improved with the shared encoder network owing to the superior performance of transducer [2,7,18]. Compared to the commonly equipped attention decoder [5,19,20], the essential advantage of the delay-penalized transducer is that it can force the encoder probability distribution to shift to the left, and thereby further reduce the symbol delay of the CTC head.

Our contributions are as follows:

- We propose the delay-penalized CTC, which can be used to train low-latency streaming models without external reference alignments.
- A simple and effective implementation based on differentiable FST is designed specifically for the proposed delay-penalized CTC.
- Experimentally, combining the delay-penalized transducer in training enables the CTC model to achieve both lower latency and better performance.

2. Connectionist Temporal Classification

Given a feature sequence $\mathbf{x} = \{x_t\}_0^{T-1}$ of length T, and a label sequence $\mathbf{y} = \{y_u \in \mathcal{V}\}_0^{U-1}$ of length U, where $\mathcal{V}$ is the vocabulary, CTC [1] aims to maximize the log-probability $\log P(\mathbf{y}|\mathbf{x})$. By introducing a blank token $\varnothing$, CTC defines the alignments between $\mathbf{x}$ and $\mathbf{y}$: $\pi = \{\pi_t\}_0^{T-1}$, $\pi_t \in \mathcal{V} \cup \{\varnothing\}$. An encoder network followed by a linear projection layer and a log-softmax function is utilized to estimate the log-probabilities $\log P(\pi_t)$.

Let $\mathcal{B}(\pi) = \mathbf{y}$ denote a many-to-one map that merges repeated contiguous tokens and removes $\varnothing$. The CTC objective

[*] stands for equal contribution
[1] https://github.com/k2-fsa/k2

10.21437/Interspeech.2023-2508

function is to maximize the total log-probability $\mathcal{L}$ over all valid alignments $\pi \in \mathcal{B}^{-1}(\mathbf{y})$:

$$\mathcal{L} = \log \sum_{\pi} \exp(s_\pi), \tag{1}$$

where s_π is the log-probability of alignment π:

$$s_\pi = \sum_{t} \log P(\pi_t). \tag{2}$$

Typically, $\mathcal{L}$ is efficiently computed employing the forward-backward algorithm [1].

Similar to transducer [14], CTC also encounters the latency problem when applied to streaming ASR models [8]. For example, in streaming Conformer models [21] trained with chunk-wise attention mask, the learned log-probabilities $\log P(\pi_t)$ only access limited future context. On the resulting CTC lattice, the higher-delay alignments that access more context would earn higher scores compared to the lower-delay counterparts, which leads to high recognition latency during decoding.

3. Delay-penalized CTC

We will first present the formulation of the proposed delay-penalized CTC in section 3.1. Then we will elaborate on the detailed FST-based implementation in section 3.2. Finally we will introduce how we utilize the delay-penalized transducer for a better delay-accuracy trade-off in section 3.3.

3.1. Formulation

Following the delay-penalized transducer [14], we aim to similarly regularize the CTC objective function $\mathcal{L}$ with an extra term $\mathcal{L}_{\text{delay}}$:

$$\mathcal{L}_{\text{aug}} = \mathcal{L} + \mathcal{L}_{\text{delay}}. \tag{3}$$

Herein, $\mathcal{L}_{\text{delay}}$ is the averaged delay scores (inversely proportional to symbol delay) of the CTC lattice, scaled by a hyper-parameter λ:

$$\mathcal{L}_{\text{delay}} = \lambda \sum_{\pi} w_\pi \cdot d_\pi, \tag{4}$$

where d_π denotes the delay score of alignment π, w_π denotes the normalized alignment weight over the whole lattice:

$$w_\pi = \frac{\exp(s_\pi)}{\sum_\pi \exp(s_\pi)}. \tag{5}$$

With the augmented objective function $\mathcal{L}_{\text{aug}}$, the CTC model is trained to assign higher log-probabilities on those lower-delay alignments with higher delay scores.

According to the mathematical proof in [14], for a small λ, we can approximately compute $\mathcal{L}_{\text{aug}}$ by replacing alignment scores s_π with $s'_\pi = s_\pi + \lambda \cdot d_\pi$ in the regular CTC function $\mathcal{L}$ in (1):

$$\mathcal{L}_{\text{aug}} \approx \log \sum_{\pi} \exp(s_\pi + \lambda \cdot d_\pi). \tag{6}$$

It can be rearranged as a Bayes risk CTC formula [8]:

$$\mathcal{L}_{\text{aug}} \approx \log \sum_{\pi} \exp(s_\pi) \cdot \exp(\lambda \cdot d_\pi). \tag{7}$$

where the delay score term $\exp(\lambda \cdot d_\pi)$ is the Bayes risk function for alignment π.

Let $\mathbf{q} = \{q_u\}_0^{U-1}$ denote the frame indexes that firstly emit non-blank tokens $\mathbf{y} = \{y_u\}_0^{U-1}$. The delay score of alignment π is defined as:

$$d_\pi = \sum_{u} \left(\frac{T-1}{2} - q_u \right), \tag{8}$$

where the middle-frame offset $\frac{T-1}{2}$ is used to keep the delay penalty in a proper numerical range. From (2), (6) and (8), before calculating $\mathcal{L}_{\text{aug}}$ with the forward-backward algorithm [1], we just need to locate those frames $\mathbf{q}$ that firstly emit non-blank tokens and add corresponding frame offsets to the CTC log-probabilities:

$$\log P'(\pi_{q_u}) = \log P(\pi_{q_u}) + \lambda \cdot \left(\frac{T-1}{2} - q_u \right). \tag{9}$$

However, to the best of our knowledge, it is a hassle to integrate this change in regular CTC implementations, such as PyTorch-based CTC[2] and warp-ctc[3].

3.2. Finite State Transducer-based Implementation

We leverage k2 framework[4] to implement the proposed delay-penalized CTC in a simple and efficient way based on the differentiable Finite State Transducer (FST). The FST-based CTC training procedure [15–17] typically involves three components: CTC topology (denoted as $\mathbf{H}$, in Figure 1(a)[5]), lexicon (denoted as $\mathbf{L}$, in Figure 1(b)), and Dense Finite State Acceptor (FSA)[6] (denoted as $\mathbf{U}$ [22], in Figure 1(c)). Specifically, $\mathbf{H}$ serves as the map $\mathcal{B}(\pi) = \mathbf{y}$ that merges repetitions and removes $\varnothing$. $\mathbf{L}$ converts sequence of tokens in $\mathcal{V}$ into word sequence. The weights on $\mathbf{U}$ correspond to the acoustic log-probabilities. Thus we can construct the CTC lattice (denoted as $\mathbf{HLU}$, in Figure 1(d)) which contains all valid alignments $\mathcal{B}^{-1}(\mathbf{y})$ by the following FST operations:

$$\mathbf{HLU} = \text{intersect}(\text{compose}(\mathbf{H}, \mathbf{L}), \mathbf{U}). \tag{10}$$

We can get the CTC loss value and propagate the gradients back to acoustic log-probabilities by computing the total score of $\mathbf{HLU}$ based on dynamic programming algorithm in a differentiable manner.[7]

In order to identify the frame indexes that firstly emit non-blank tokens, to which the corresponding penalty scores as in (9) will add, we attach a binary attribute to the arcs in CTC topology $\mathbf{H}$ (in Figure 1(a)) indicating the presence of non-epsilon output label. Thanks to the k2 mechanism, this attribute will be propagated automatically between the FST operations in (10). Hence, on the resulting lattice $\mathbf{HLU}$, we are able to easily locate those arcs where non-blank tokens are firstly emitted with the attached attribute, and add corresponding frame offsets to the log-probabilities (in Figure 1(d)). By maximizing the total score of the modified $\mathbf{HLU}$, the CTC model is regularized to enhance those lower-delay alignments.

3.3. Jointly Trained with Delay-penalized Transducer

In this paper, we also explore training the delay-penalized CTC with a delay-penalized transducer [14] as an auxiliary task to achieve higher performance and lower latency for the CTC model. Specifically, the CTC head and the transducer head share a common encoder network but own their individual linear projection layers. As in the hybrid CTC/Attention framework [5], conducting the multi-task learning will strengthen the modeling capacity of the shared encoder. More importantly,

[2] https://github.com/pytorch/pytorch

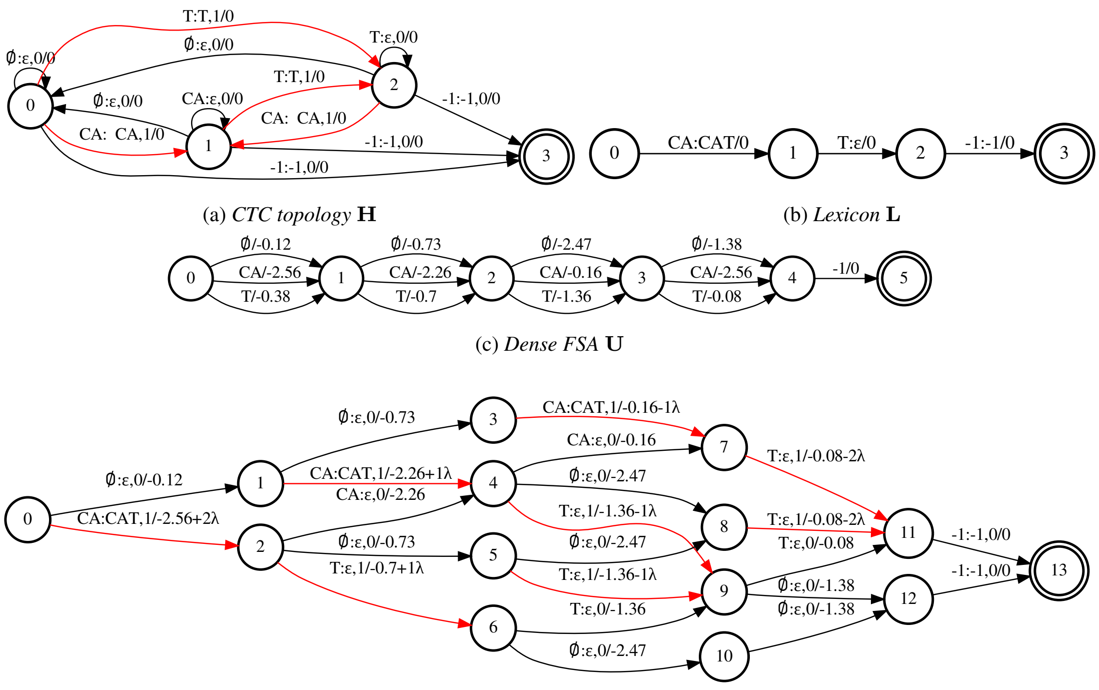
[3] https://github.com/baidu-research/warp-ctc

[4] https://github.com/k2-fsa/k2

[5] Note: In k2 framework, arcs entering the final state always have -1 as label.

[6] Search for DenseFsaVec in https://github.com/k2-fsa/k2 for more details.

[7] Search for Fsa.get_tot_scores in https://github.com/k2-fsa/k2 for more details.



(a) *CTC topology* **H**

(b) *Lexicon* **L**

(c) *Dense FSA* **U**

(d) *CTC lattice* **HLU**

Figure 1: *Delay-penalized CTC implemented based on FST. It visualizes the BPE tokens of "CAT". We attach a binary attribute to* **H**, *where* **1** *and* **0** *indicate the arcs with and without non-epsilon (ϵ) output labels, respectively. The arcs with non-epsilon output labels are highlighted in red. For example, the red arc from state 0 to state 2 on* **H** *with label T:T,***1***/0, represents input token T and output token T with the binary attribute of* **1** *and score of 0. On the resulting CTC lattice* **HLU**, *we can locate those arcs (in red) that firstly emit non-blank tokens with the attached binary attribute, and add corresponding frame offsets to the log-probabilities.*

with the delay-penalized transducer [14], it is expected to shift the output probability distribution from the shared encoder to the left along the time axis, and thereby encourage the CTC head to emit symbols earlier.

The joint objective function $\mathcal{L}^{\text{joint}}$ is formulated as:

$$\mathcal{L}^{\text{joint}} = \alpha \cdot \mathcal{L}^{\text{ctc}}_{\text{aug}} + \mathcal{L}^{\text{transducer}}_{\text{aug}} \tag{11}$$

where $\mathcal{L}^{\text{ctc}}_{\text{aug}}$ (i.e., $\mathcal{L}_{\text{aug}}$ in (3)) and $\mathcal{L}^{\text{transducer}}_{\text{aug}}$ are the delay-penalized objective functions for the CTC head and the transducer head, respectively. Note that both heads have their own delay penalty scales λ. The hyper-parameter α is used to make the loss values of two heads numerically close.

4. Experiments

4.1. Experimental Setup

Dataset and Evaluation metrics. We conduct ASR experiments on LibriSpeech dataset [23], which contains 1000 hours of English speech with sampling rate of 16 kHz. We evaluate model performance and latency on subsets *test-clean* and *test-other*. Our reference alignments are generated using the torchaudio toolkit [24]. Word Error Rate (WER) is used to evaluate the recognition performance. Lower value of WER indicates better recognition result. Two metrics are employed to measure the CTC latency, namely Mean Start Delay (MSD) and Mean End Delay (MED). Specifically, for all correctly recognized words, MSD and MED calculate the averaged start time difference and the averaged end time difference between the prediction and the reference alignment, respectively. Higher value of MSD or MED indicates higher emission latency. CTC greedy search is adopted as the decoding method for a better measure of model performance and symbol delay.

Implementation Details. Our data preparation is performed using the Lhotse [25] framework. Speed perturbation [26] with factors of 0.9 and 1.1 is applied to augment the training data. SpecAugment [27] and MUSAN [28]-based noise augmentation are utilized to improve model robustness in training. The input features are 80-channel Fbank extracted on 25 ms windows shifted by 10 ms. The classification units are 500-classes Byte Pair Encoding (BPE) [29] word pieces. We use a 12-layer Conformer [21] with 77 M parameters as the encoder. The input features are subsampled with a factor of 4 by a convolutional layer. The embedding dimension is set to 512. The kernel size of the convolution layers is set to 31. The feedforward dimension is set to 2048. To train the streaming models, the attention layers are trained with block-triangular mask to limit the future context. During streaming inference we use a chunk size of 320 ms. While training with the auxiliary transducer head, we adopt the pruned transducer [7] implementation for high efficiency and low memory consumption, in which the stateless decoder [30] is a 1-D convolution layer with a kernel size of 2. The hyper-parameter α in (11) is set to 0.2. All models are trained for 35 epochs with automatic mixed precision.

4.2. Results of delay-penalized CTC

We first perform experiments to validate the effectiveness of the proposed method of delay penalization on CTC implemented based on FST. Table 1 presents the results of CTC models in non-streaming and streaming modes respectively. Compared to the non-streaming model, the streaming model without delay penalty obtains significantly higher symbol delay, since CTC learns to augment the higher-delay alignments that access more future context during training. We also conduct an experiment to leverage the pre-computed reference token-time alignments for comparison, which performs the frame-wise BPE token

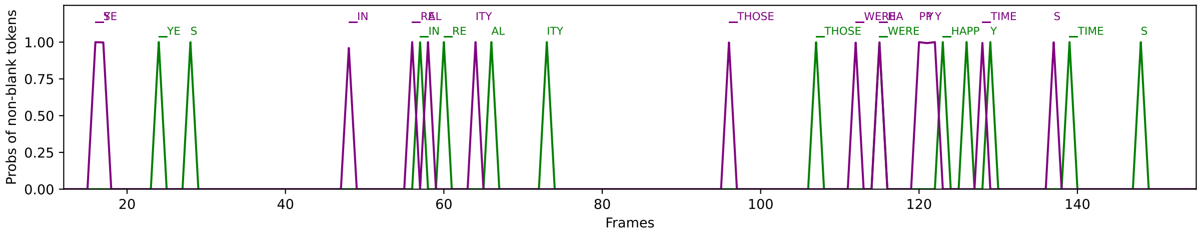


Figure 2: *Visualization of CTC output probability distributions of non-blank tokens from the streaming models with (in purple) and without (in green) delay penalization, respectively.*

Table 1: *Results of CTC models in non-streaming and streaming modes.*

Method	*test-clean*			*test-other*		
	WER (%)	MSD (ms)	MED (ms)	WER (%)	MSD (ms)	MED (ms)
Non-streaming	3.24	-28	-100	7.89	-28	-100
Streaming	4.56	273	189	12.21	275	192
+ Reference Alignment	5.45	117	33	13.26	117	34
+ Delay penalty, $\lambda = 0.005$	5.00	179	97	12.86	186	105
+ Delay penalty, $\lambda = 0.010$	5.32	108	33	13.82	121	44
+ Delay penalty, $\lambda = 0.015$	5.58	42	-36	14.04	63	-17
+ Delay penalty, $\lambda = 0.020$	5.96	3	-75	14.81	23	-56
+ Delay penalty, $\lambda = 0.025$	6.44	-18	-101	15.53	2	-81

classification with a loss scale of 0.75. The result indicates that the streaming model learns to predict non-blank tokens earlier with the frame-wise supervision by the reference alignments. For the streaming models with delay penalty, increasing λ consistently leads to lower MSD and MED, and higher WER. It manifests that the proposed method can effectively balance the trade-off between symbol delay and accuracy by tuning λ. Note that the delay-penalized model with $\lambda = 0.010$ achieves similar symbol delay compared to the model using reference alignments, which demonstrates the effectiveness of the proposed regularization method.

Figure 2 visualizes the CTC output probability distributions of non-blank tokens from the streaming models with ($\lambda = 0.025$) and without delay penalization, respectively. The CTC peaks shifting to the left along the time axis indicates that the delay penalty regularization enables the CTC model to learn to emit non-blank tokens earlier and thereby achieve lower latency.

4.3. Results of combining delay-penalized transducer

Next we conduct experiments to investigate the effect of leveraging the delay-penalized transducer [14] as an auxiliary task for the streaming CTC model. We use the commonly adopted hybrid CTC/Attention framework [5] for comparison. Specifically, we build a 6-layer transformer-based decoder with 25.7 M model parameters, which serves as a jointly learned language model. The total number of model parameters in the pruned transducer head [14] is only 1.0 M. Table 2 presents the results in terms of symbol delay and recognition accuracy of the CTC models trained with different auxiliary tasks. Training with attention decoder and transducer both improve the CTC recognition performance owing to their modeling ability in learning the dependencies of the output tokens. Attention decoder does not affect the CTC latency, due to its non-streaming modeling mechanism that accesses full encoder context [5]. Training with the transducer without delay penalty leads to a higher symbol delay, since the transducer also favors the higher-delay alignments that access more future context [14] and thereby affects the probability distribution from encoder. After applying delay penalty on the transducer [14], the CTC models achieve lower latency compared to the baseline. It demonstrates the advantage of combining the delay-penalized transducer over the attention decoder that it can force the encoder probability distribution to shift to the left along the time axis.

Table 3 presents the results of delay-penalized CTC models trained with and without the delay-penalized transducer [14] using different λ, respectively. Combining the delay-penalized transducer consistently improves the recognition performance and reduces the symbol delay for CTC models, which manifests the effectiveness of the proposed multi-task framework.

Table 2: *Results of streaming CTC models trained with different auxiliary tasks.*

Method	*test-clean*			*test-other*		
	WER (%)	MSD (ms)	MED (ms)	WER (%)	MSD (ms)	MED (ms)
Baseline	4.56	273	189	12.21	275	192
+ Attention	4.15	270	188	10.89	273	191
+ Transducer	4.00	342	271	10.42	345	274
+ Transducer, $\lambda = 0.0050$	4.19	151	72	10.95	175	97
+ Transducer, $\lambda = 0.0075$	4.42	135	49	11.21	155	72
+ Transducer, $\lambda = 0.0100$	4.51	122	37	11.40	141	58

Table 3: *Results of delay-penalized CTC models trained w/o the delay-penalized transducer respectively.*

λ in CTC	λ in Transducer	*test-clean*			*test-other*		
		WER (%)	MSD (ms)	MED (ms)	WER (%)	MSD (ms)	MED (ms)
0.010	-	5.32	108	33	13.82	121	44
0.010	0.0050	**4.74**	**77**	**-3**	**11.89**	**101**	**22**
0.015	-	5.58	42	-36	14.04	63	-17
0.015	0.0075	**4.91**	**15**	**-58**	**12.05**	**48**	**-26**
0.020	-	5.96	3	-75	14.81	23	-56
0.020	0.0100	**5.3**	**-23**	**-99**	**12.76**	**8**	**-67**

5. Conclusion

In this work, we propose the delay-penalized CTC, which is implemented based on the differentiable FST. Specifically, for the CTC lattice modeled by FST, we locate the frames that firstly emit non-blank tokens and add the corresponding frame offsets to the CTC log-probabilities. Experimental results demonstrate that it can effectively balance the trade-off between symbol delay and recognition performance. Furthermore, leveraging a delay-penalized transducer as the auxiliary task enables the CTC model to achieve a better delay-accuracy trade-off.

6. References

[1] 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.

[2] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. ICML Workshop on Representation Learning*, Edinburgh, 2012.

[3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[4] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[5] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[6] J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, "On the comparison of popular end-to-end models for large scale speech recognition," in *Proc. Interspeech*, 2020, pp. 1–5.

[7] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned RNN-T for fast, memory-efficient ASR training," in *Proc. Interspeech*, 2022, pp. 2068–2072.

[8] 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.

[9] A. Senior, H. Sak, F. de Chaumont Quitry, T. Sainath, and K. Rao, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 604–609.

[10] H. Sak, A. W. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. Interspeech*, 2015, pp. 1468–1472.

[11] H. Inaguma and T. Kawahara, "StableEmit: Selection Probability Discount for Reducing Emission Latency of Streaming Monotonic Attention ASR," in *Proc. Interspeech*, 2021, pp. 1817–1821.

[12] 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," *arXiv preprint arXiv:2211.03284*, 2022.

[13] X. Song, D. Wu, Z. Wu, B. Zhang, Y. Zhang, Z. Peng, W. Li, F. Pan, and C. Zhu, "Trimtail: Low-latency streaming asr with simple but effective spectrogram-level length penalty," *arXiv preprint arXiv:2211.00522*, 2022.

[14] W. Kang, Z. Yao, F. Kuang, L. Guo, X. Yang, P. Żelasko, D. Povey *et al.*, "Delay-penalized transducer for low-latency streaming asr," *arXiv preprint arXiv:2211.00490*, 2022.

[15] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4280–4284.

[16] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.

[17] A. Laptev, S. Majumdar, and B. Ginsburg, "CTC variations through new WFST topologies," in *Proc. Interspeech*, 2022, pp. 1041–1045.

[18] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7829–7833.

[19] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[20] 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," in *Proc. Interspeech*, 2022, pp. 1661–1665.

[21] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[22] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, "Generating exact lattices in the wfst framework," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4213–4216.

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

[24] Y.-Y. Yang, M. Hira, Z. Ni, A. Astafurov, C. Chen, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang *et al.*, "Torchaudio: Building blocks for audio and speech processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6982–6986.

[25] P. Żelasko, D. Povey, J. Trmal, S. Khudanpur *et al.*, "Lhotse: a speech data representation library for the modern deep learning ecosystem," *arXiv preprint arXiv:2110.12561*, 2021.

[26] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015, pp. 3586–3589.

[27] 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," in *Proc. Interspeech*, 2019, pp. 2613–2617.

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

[29] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1715–1725.

[30] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-transducer with stateless prediction network," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7049–7053.