INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

# BAT: Boundary aware transducer for memory-efficient and low-latency ASR

*Keyu An, Xian Shi, Shiliang Zhang*

Speech Lab of DAMO Academy, Alibaba Group, China
{ankeyu.aky, shixian.shi, sly.zsl}@alibaba-inc.com

## Abstract

Recently, recurrent neural network transducer (RNN-T) gains increasing popularity due to its natural streaming capability as well as superior performance. Nevertheless, RNN-T training requires large time and computation resources as RNN-T loss calculation is slow and consumes a lot of memory. Another limitation of RNN-T is that it tends to access more contexts for better performance, thus leading to higher emission latency in streaming ASR. In this paper we propose boundary-aware transducer (BAT) for memory-efficient and low-latency ASR. In BAT, the lattice for RNN-T loss computation is reduced to a restricted region selected by the alignment from continuous integrate-and-fire (CIF), which is jointly optimized with the RNN-T model. Extensive experiments demonstrate that compared to RNN-T, BAT reduces time and memory consumption significantly in training, and achieves good CER-latency trade-offs in inference for streaming ASR.

**Index Terms**: RNN-T, memory-efficient, low-latency, CIF

## 1. Introduction

Recently, the recurrent neural network transducer (RNN-T) [1] has emerged as a promising end-to-end ASR framework due to its competitive performance and streaming-friendly nature. RNN-T models the acoustic and language features jointly, which eliminates the drawbacks in the output-independent CTC model [2]. Nevertheless, this appealing feature comes at the cost of high memory and computation consumption during training. Specifically, the RNN-T loss calculates on a 4-D lattice of shape (N, T, U, V), where N is the batch size, T is the output length of the acoustic encoder, U is the output length of the prediction network, and V is the vocabulary size. The large memory requirements limit the RNN-T training over large batches and hence slow down the training speed, especially for languages like Mandarin, where a large vocabulary set is adopted. To overcome it, frame downsampling or skipping [3] (reducing T), restrictions on the RNN-T lattice (reducing T or U) [4, 5], sampled softmax (reducing V) [6], encoder and prediction output combination (reducing memory waste caused by padding) [7], function merging (reducing memory waste caused by calculating and storing intermediate variables) [7], and more efficient training pipeline (reducing training epochs) [8] have been studied in the previous work. In these works, the main challenge is to reduce memory consumption without degradation of accuracy.

Another important issue for RNN-T is the emission latency. RNN-T model optimized with unregularized loss tend to use more (future) context to produce better predictions, which causes significant emission delays [9] (the difference between the user speaking and the model prediction). To address it, sequence-level emission regularization [9] and token-level restrictions to enforce prediction within a reasonable time window [5, 10] were adopted in the previous work. Despite the progress, it is still challenging to obtain good WER-latency trade-offs.

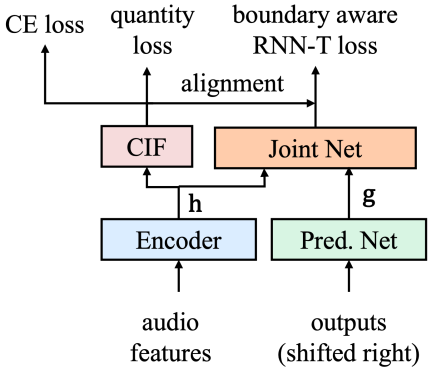


Figure 1: *Boundary-aware training of RNN-T with the CIF alignment.*

In this paper, we propose boundary-aware transducer (BAT) for memory-efficient, low-latency ASR, as illustrated in Fig. 1. In BAT, the lattice for RNN-T loss calculation is reduced to a restricted region selected by the continuous integrate-and-fire (CIF) alignment [11]. Thus, we only need to consider the forward variables $\alpha(t, u)$ and backward variables $\beta(t, u)$ within the limited region, which greatly reduces memory consumption in training. Moreover, the restricted alignment prevents the model from using unlimited context to produce the prediction, which leads to faster token emission for streaming inference. The CIF module is lightweight and fast, and thus can be jointly optimized with RNN-T from scratch, without great additional computation overhead. Results on the public AISHELL-1 datasets and non-public large-scale in-house data demonstrate that BAT achieves significant memory and latency reduction while maintaining recognition accuracy.

The paper is organized as follows. Section 2 outlines related work. Section 3 details the method of the boundary-aware transducer. Experiments are shown in Section 4. Section 5 discusses the limitations of the method and future works. Section 6 gives the conclusion.

## 2. Related work

*Reducing the memory usage* for RNN-T training by performing the forward-backward calculation on a reduced lattice has


10.21437/Interspeech.2023-770

been explored in several previous works. Alignment Restricted RNN-T (Ar-RNN-T) [5] proved that it's possible to restrict regions for each token using a pre-trained hybrid ASR model, which leads to less memory usage. In the recently proposed pruned RNN-T [4], the pruned boundary is generated by firstly computing the forward-backward algorithm on a "trivial" joint network, and then selecting the (t, u) pairs which contribute most to the final loss based on the output of the "trivial" joint network.

For *reducing the emission latency*, using a pre-generated alignment to constrain or guide the RNN-T training is also found to be beneficial. In these works, an external low-latency acoustic model (typically a hybrid model) is required to provide the frame-wise alignment. After that, the emission latency is penalized by masking out the alignment paths exceeding the predetermined threshold delay on the RNN-T lattice [5, 10], or using an auxiliary loss [12]. Recently, sequence-level emission regularization methods propose to reduce latency by modifying the RNN-T loss to find paths tending to predict vocabulary tokens instead of blanks [9, 13], without an external alignment. Nevertheless, the path may not be optimal for ASR due to a lack of alignment information, which can degrade the ASR accuracy severely [14, 13].

Inspired by the previous studies, boundary-aware transducer tries to achieve memory and latency reduction at the same time, based on the alignment information. The major differences from previous works are 1) in BAT, the alignment information is generated on-the-fly using a lightweight and fast CIF module, which is jointly optimized with the RNN-T, thus not requiring a pre-trained (hybrid) model to obtain the token alignment. 2) the alignment generated by CIF is monotonic and continuous. Thus, there is no need to apply monotonic and continuity constraints to the token boundary, as pruned RNN-T did. 3) we give a thorough analysis on both memory usage and latency for our method and demonstrate its effectiveness.

# 3. Method

## 3.1. RNN-T loss

In the training stage, the RNN Transducer (RNN-T) model [1] aims to maximize the log-probability of a conditional distribution over the target token sequences $\mathbf{y} = (y_1, y_2, ..., y_U) \in \mathcal{Y}$ given the input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$:

$$\mathcal{L} = -\log \Pr(\mathbf{y}|\mathbf{x}) = -\log \sum_{\mathbf{a} \in \mathcal{B}^{-1}(y)} \Pr(\mathbf{a}|\mathbf{x})$$

where $\mathbf{a} = (a_1, a_2, ..., a_{T+U}) \in \mathcal{Y} \cup \{\phi\}$ is the blank label $\phi$ augmented alignment sequence, and the mapping $\mathcal{B}$ is defined by removing $\phi$ in the input sequence.

$\Pr(\mathbf{a}|\mathbf{x})$ is further factorized as

$$\Pr(\mathbf{a}|\mathbf{x}) = \sum_{i=1}^{T+U} \Pr(a_i|h_{t_i}, g_{u_i}) \tag{1}$$

where $\mathbf{h} = (h_1, h_2, ..., h_T) = \mathrm{Enc}(\mathbf{x})$ is the high-level representation produced by the encoder, and $g_u$ is the prediction vector computed by the prediction network,

$$g_u = \mathrm{PredictNet}(\mathbf{y}_{[0:u-1]}) \tag{2}$$

, with the convention $y_0 = \phi$. The probability $\Pr(\cdot|h_t, g_u)$ is typically implemented as the output of the joint network:

$$\Pr(\cdot|h_t, g_u) = \mathrm{softmax}[\mathbf{W}^{out}\tanh(\mathbf{W}^{enc}h_t + \mathbf{W}^{pred}g_u + b)] \tag{3}$$

To optimize the transducer objective, $\Pr(\mathbf{y}|\mathbf{x})$ and gradients are calculated using the efficient forward-backward algorithm [1]:

$$\Pr(\mathbf{y}|\mathbf{x}) = \sum_{(t,u):t+u=n} \alpha(t,u)\beta(t,u), \forall n : 1 \le n \le T+U$$

$$\frac{\partial \Pr(\mathbf{y}|\mathbf{x})}{\partial \Pr(a|h_t, g_u)} = \alpha(t,u) \begin{cases} \beta(t,u+1), \text{if } a = y_{u+1}, \\ \beta(t+1,u), \text{if } a = \phi, \\ 0, \text{otherwise.} \end{cases}$$

where $\alpha(t,u)$ is the forward variable, defining the probability of outputting $y_{[1:u]}$ during $h_{[1:t]}$, and $\beta(t,u)$ is the backward variable, defining the probability of outputting $y_{[u+1:U]}$ during $h_{[t:T]}$. Denote

$$y(t,u) = \Pr(y_{u+1}|h_t, g_u)$$

$$\phi(t,u) = \Pr(\phi|h_t, g_u)$$

Then the forward variables and backward variables can be calculated recursively using

$$\begin{aligned} \alpha(t,u) &= \alpha(t-1,u)\phi(t-1,u) + \alpha(t,u-1)y(t,u-1) \\ \beta(t,u) &= \beta(t+1,u)\phi(t,u) + \beta(t,u+1)y(t,u) \end{aligned} \tag{4}$$

It can be seen that RNN-T loss computation can consume a lot of time and memory because it has to compute $y(t,u)$ and $\phi(t,u)$ for all $0 \le t \le T$ and $0 \le u \le U$ on a lattice of shape (N, T, U, V).

## 3.2. Boundary aware training with CIF alignment

In BAT, we use continuous integrate-and-fire (CIF) [11] to generate the monotonic alignment between the acoustic signals and token sequences and use it to restrict the paths being optimized during training, as illustrated in Fig. 2. The pipeline of boundary-aware training is detailed below.

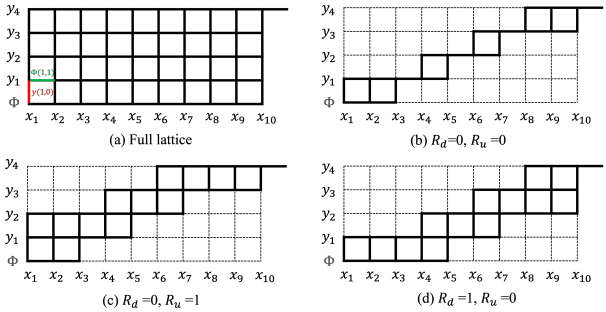


Figure 2: *Output probability lattice defined by the joint network output.* $(a)$ *is the full lattice of standard RNN-T, and* $(b)(c)(d)$ *are restricted lattices with different* $R_d$ *and* $R_u$*, given the CIF alignment* $\mathcal{C}$ = $[1, 1, 1, 2, 2, 3, 3, 4, 4, 4]$.

**CIF.** Given the encoder output $\mathbf{h} = (h_1, h_2, ..., h_T)$, CIF predicts the weights $\boldsymbol{\omega} = (\omega_1, \omega_2, ..., \omega_T)$ using

$$\boldsymbol{\omega} = \mathrm{Sigmoid}(\mathrm{Linear}(\mathrm{Conv}(\mathbf{h}))) \tag{5}$$

Then, it forwardly accumulates the weights and integrates the encoder outputs until the accumulated weight reaches a given threshold $\beta_{\mathrm{CIF}}$, which means a token boundary is located. It then instantly fires the integrated acoustic information for token

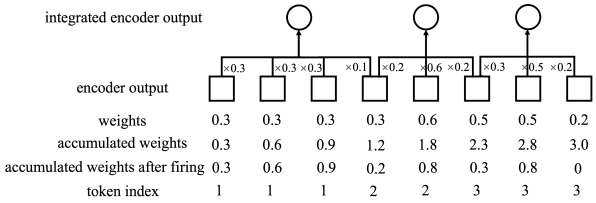


Figure 3: *Token boundary generation with CIF. The weights have been scaled to match the target token number (3) and the threshold $\beta_{\text{CIF}}$ is 1.*

prediction and updates the accumulated weights. In the training stage, the weights $\boldsymbol{\omega}$ are scaled by $\frac{U}{\sum_{t=1}^{T}\omega_t}$ so that the predicted length of the token sequence is equal to the length of the target sequence and the model can be optimized using a simple cross-entropy (CE) loss $\mathcal{L}_{\text{CIF}-\text{CE}}$, and quantity loss term

$$\mathcal{L}_{\text{CIF}-\text{QUA}} = \left| \sum_{t=1}^{T} \omega_t - U \right| \tag{6}$$

is added to the total loss to encourage the model to predict the length of labels closer to the targets. We adopt a fast parallel implementation and a lightweight neural net configuration for the CIF module so the additional parameter and computation overhead introduced by CIF is insignificant.

**Token boundary generation.** Given $\boldsymbol{\omega}$, we obtain the CIF alignment $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_T)$ sequence simply using

$$\boldsymbol{\mathcal{C}} = \text{ceil}(\text{cumsum}(\boldsymbol{\omega})) \tag{7}$$

, where $\mathcal{C}_t$ is defined as the index of the token to which $h_t$ assigns. $\text{ceil}()$ is the ceiling function and $\text{cumsum}()$ is the cumulative sum operation, as illustrated in Fig. 3. Note that the alignment given by eq. 7 is naturally monotonic ($\mathcal{C}_{t+1}$-$\mathcal{C}_t \geq 0$) and continuous ($\mathcal{C}_{t+1}$-$\mathcal{C}_t \leq 1$) so the ad hoc operations to make the pruning boundary valid in pruned RNN-T [4] are avoided.

**Boundary aware training.** In boundary-aware training, we assume that $y(t,u)$ and $\phi(t,u)$ with non-zero gradients are located in a small neighboring region of CIF alignment. Thus, instead of considering $y(t,u)$ and $\phi(t,u)$ over all $U$ tokens at each time step $t$, we only compute $y(t,u)$ for

$$\mathcal{C}_t - R_d \leq u \leq \mathcal{C}_t + R_u$$

, and $\phi(t,u)$ for

$$\mathcal{C}_t - R_d \leq u \leq \mathcal{C}_t + R_u + 1$$

. On the contrary, $y(t,u)$ for

$$u \in \{u \mid u < \mathcal{C}_t - R_d \text{ or } u > \mathcal{C}_t + R_u\}$$

and $\phi(t,u)$ for

$$u \in \{u \mid u < \mathcal{C}_t - R_d \text{ or } u > \mathcal{C}_t + R_u + 1\}$$

will be treated as 0. $R_d$ and $R_u$ are two hyper-parameters that control the ranges of the tokens that will be evaluated for time step $t$. Thus, the output lattice shape becomes (N, T, $R_d + R_u + 2$, V), which greatly reduces memory consumption. In the training stage, the CIF module is jointly optimized with the boundary-aware transducer (BAT) and the total training objective is defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{BAT}} + \mathcal{L}_{\text{CIF}-\text{CE}} + \mathcal{L}_{\text{CIF}-\text{QUA}}$$

# 4. Experiments

## 4.1. Experiment settings

We conduct our experiments on the openly available 170-hour Mandarin AISHELL-1 [15] dataset a 30000-hour in-house industrial Mandarin dataset. The code and pre-trained model will be released upon publication of the paper.

On AISHELL-1, we use 80-dim filterbanks as input features. SpecAugment [16] and 3-fold speed perturbation are used for data augmentation. The encoder is a 12-layer conformer [17]. The convolution kernel size is 31, and the number of attention heads, attention dimension, and feed-forward dimension are 8, 512, and 2048 respectively. We perform frame downsampling on 1) the input feature by a factor of 4 using stride convolution and 2) the output of the encoder by a factor of 2 to reduce the training and inference memory consumption. The prediction network is a 1-layer LSTM with 512 hidden units. The modeling units are 4233 Chinese characters. The CIF module consists of a 1D convolution layer with 512 channels and a linear layer. The total number of parameters is about 90M. For the streaming model, we adopt the causal convolution [18] and chunk attention based conformer. The convolution kernel size is 15 and the attention chunk size is 16. Other configurations are the same as the non-streaming model. We train the model from scratch for 100 epochs and average 10 checkpoints which perform best on the development set to obtain the final model. At the inference stage, we use beam search with a beam size 10, and no extra language model is used.

On the 30000-hour in-house dataset, we stack the consecutive frames within a context window of 7 (3+1+3) to produce the 560-dimensional features and then perform 6× down-sampling on the input frames. The modeling units are 3445 Chinese characters. The model configurations are the same as the AISHELL-1. We train the model for 20 epochs and average 5 checkpoints which perform best on the development set.

## 4.2. Latency metrics

We use two metrics to characterize the latency for streaming RNN-T and streaming BAT.

**Average last token Emitting Time, avg ET.** The average time when the last token is emitted for all utterances in the test set.

**Partial Recognition (PR) Latency.** The difference of the time (1) when the last token is emitted and (2) when a user finishes speaking, which is estimated by the alignment from the conventional model. Following [9], we report both 50-th (medium, PR50) and 90-th (PR90) percentile values of PR.

## 4.3. Results

Table 1: *The non-streaming RNN-T and boundary-aware transducer (BAT) results on AISHELL-1.*

| model | $R_d$ | $R_u$ | dev CER | test CER |
|---|---|---|---|---|
| RNN-T | - | - | 4.86 | **5.22** |
| BAT | 1 | 1 | 5.05 | 5.45 |
| BAT | 2 | 2 | 4.86 | 5.32 |
| BAT | 3 | 3 | **4.82** | 5.28 |

For the non-streaming model (Table 1), BAT and RNN-T perform comparably in accuracy, especially when $R_d$ and $R_u$

Table 2: *The streaming RNN-T and boundary-aware transducer (BAT) results on AISHELL-1. ET is the last token emitting time. PR is Partial Recognition Latency. FE is short for FastEmit. $\lambda$ is the FastEmit hyperparameter.*

| Exp ID | model | $R_d$ | $R_u$ | dev CER | test CER | avg ET (ms) | PR50 (ms) | PR90 (ms) |
|---|---|---|---|---|---|---|---|---|
| 1 | RNN-T | - | - | 5.80 | 6.96 | 4633 | 100 | 230 |
| 2 | RNN-T + FE, $\lambda = 0.002$ | - | - | 6.11 | 7.44 | 4517 (-116) | -20 (-120) | 120 (-110) |
| 3 | RNN-T + FE, $\lambda = 0.004$ | - | - | 6.04 | 7.67 | 4416 (-217) | -110 (-210) | 30 (-200) |
| 4 | BAT | 1 | 1 | 5.89 | 7.63 | 4483 (-150) | -10 (-110) | 120 (-110) |
| 5 | BAT | 2 | 2 | 5.88 | 7.35 | 4523 (-110) | 10 (-90) | 140 (-90) |

are relatively large ($R_d$=3 and $R_u$=3). A memory-CER trade-off is observed as smaller $R_d$ and $R_u$ lead to less memory usage but higher CERs.

For the streaming model (Table 2), we report results for BAT and RNN-T with and without FastEmit. It is shown that 1) similar to FastEmit [14, 13], boundary-aware training lead to accuracy degradation compared to the baseline RNN-T, 2) BAT achieves comparable CER-latency trade-offs to FastEmit, while BAT is more memory-efficient in training.

To better visualize the effectiveness of boundary-aware training in improving emission latency, we show the alignment for an utterance in the AISHELL-1 test set given by RNN-T and BAT in Fig. 4. It is shown that boundary-aware training encourages a faster emission of tokens at inference.

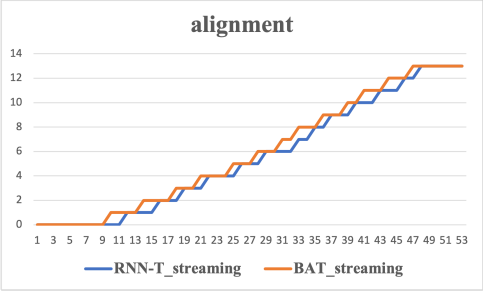


Figure 4: *Alignment of BAT ($R_d$=2, $R_u$=2, colored in orange) and RNN-T (blue) for the streaming model. The Y-axis is the non-blank token emitted by the model, and the X-axis is the time step.*

The results of the in-house 30000-hour task (table 3) prove that the proposed method can be successfully applied to RNN-T training on large-scale data.

Table 3: *The results of RNN-T and BAT ($R_d$=2, $R_u$=2) on the industrial 30000-hour task.*

| model | streaming | CER |
|---|---|---|
| RNN-T | N | 8.08 |
| BAT | N | 8.12 |
| RNN-T | Y | 8.75 |
| BAT | Y | 8.87 |

### 4.4. Time and memory usage

We benchmark the time and memory usage on the AISHELL-1 training data. The experiments are conducted on 1 Tesla V100 GPU. The model configuration is the same as the AISHELL-1 non-streaming model in Sec. 4.1. We sort utterances by the input feature lengths and set the number of frames (after padding) in a batch per GPU to 50k. Table 4 compares time and peak memory usages [1] for RNN-T loss calculation with warp-rnnt[2], pruned RNN-T[3] and BAT. We report stats for joint network and loss calculation as the encoder and prediction network calculations for the three implementations are the same. For pruned RNN-T, the time and memory statistics includes the trial joint network computation, and the number of indexes that will be evaluated for any time step (S in the original paper) is set to 5 by default. For BAT, the time and memory statistics for CIF computation are included, and $R_d = R_u$=2. It is shown that both pruned RNN-T and BAT reduce time and memory consumption drastically. BAT outperforms pruned RNN-T, which indicates that pruning bounds generated by CIF are more efficient than that in pruned RNN-T.

Table 4: *Time per batch (ms) and peak memory usage (GB) for RNN-T, pruned RNN-T and BAT ($R_d$=2, $R_u$=2).*

| model | time (ms) | peak mem usage (GB) |
|---|---|---|
| warp-rnnt | 230 | 16.9 |
| pruned RNN-T | 94 | 7.4 |
| BAT | 85 | 6.4 |

## 5. Limitations and future work

In BAT, the token boundary information is only used in the training stage, and the inference of BAT is exactly the same as the standard RNN-T. In fact, the CIF alignment could also be used to guide and speed up the RNN-T inference (e.g. perform frame skipping [19] based on the CIF weights), which would be our future work.

## 6. Conclusions

In this paper we propose boundary-aware transducer (BAT) for memory-efficient and low-latency ASR. Different from previous works which utilize alignment references generated from an external pre-trained model, BAT can be end-to-end optimized, as the alignment for BAT is generated on the fly efficiently. Extensive experiments demonstrate that compared to RNN-T, BAT reduces time and memory consumption significantly in training, and achieves good CER-latency trade-offs in streaming inference.

[1] The memory allocating information is collected using the pytorch max_memory_allocated API.

[2] https://github.com/1ytic/warp-rnnt/tree/master/pytorch_binding

[3] https://github.com/danpovey/fast_rnnt

# 7. References


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

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, p. 369–376.

[3] Y. Wang, Z. Chen, C. yong Zheng, Y. Zhang, W. Han, and P. Haghani, "Accelerating rnn-t training and inference using ctc guidance," *ArXiv preprint arXiv:2210.16481*, 2022.

[4] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned rnn-t for fast, memory-efficient asr training," *arXiv preprint arXiv:2206.13236*, 2022.

[5] J. Mahadeokar, Y. Shangguan, D. Le, G. Keren, H. Su, T. Le, C. feng Yeh, C. Fuegen, and M. L. Seltzer, "Alignment restricted streaming recurrent neural network transducer," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 52–59.

[6] J. Lee, L. Lee, and S. Watanabe, "Memory-Efficient Training of RNN-Transducer with Sampled Softmax," in *Proc. Interspeech*, 2022, pp. 4441–4445.

[7] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving rnn transducer modeling for end-to-end speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 114–121.

[8] W. Zhou, W. Michel, R. Schlüter, and H. Ney, "Efficient Training of Neural Transducer for Speech Recognition," in *Proc. Interspeech*, 2022, pp. 2058–2062.

[9] J. Yu, C.-C. Chiu, B. Li, S. yiin Chang, T. N. Sainath, Y. R. He, A. Narayanan, W. Han, A. Gulati, Y. Wu, and R. Pang, "Fastemit: Low-latency streaming asr with sequence-level emission regularization," in *Proc. ICASSP*, 2021, pp. 6004–6008.

[10] T. N. Sainath, R. Pang, D. Rybach, B. García, and T. Strohman, "Emitting word timings with end-to-end models," in *Proc. Interspeech*, 2020, pp. 3615–3619.

[11] L. Dong and B. Xu, "CIF: Continuous integrate-and-fire for end-to-end speech recognition," in *Proc. ICASSP*, 2020, pp. 6079–6083.

[12] H. Inaguma, Y. Gaur, L. Lu, J. Li, and Y. Gong, "Minimum latency training strategies for streaming sequence-to-sequence asr," in *Proc. ICASSP*, 2020, pp. 6064–6068.

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

[14] J. Kim, H. Lu, A. Tripathi, Q. Zhang, and H. Sak, "Reducing streaming asr model delay with self alignment," *arXiv preprint arXiv:2105.05005*, 2021.

[15] 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 *Proc. O-COCOSDA*, 2017, pp. 1–5.

[16] 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.

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

[18] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[19] Z. Tian, J. Yi, Y. Bai, J. Tao, S. Zhang, and Z. Wen, "Fsr: Accelerating the inference process of transducer-based models by applying fast-skip regularization," in *Proc. Interspeech*, 2021.