STREAMING TRANSFORMER ASR WITH BLOCKWISE SYNCHRONOUS BEAM SEARCH

Emiru Tsunoo[1], *Yosuke Kashiwagi*[1], *Shinji Watanabe*[2]

[1]Sony Corporation, Japan
[2]Johns Hopkins University, USA

ABSTRACT

The Transformer self-attention network has shown promising performance as an alternative to recurrent neural networks in end-to-end (E2E) automatic speech recognition (ASR) systems. However, Transformer has a drawback in that the entire input sequence is required to compute both self-attention and source–target attention. In this paper, we propose a novel blockwise synchronous beam search algorithm based on blockwise processing of encoder to perform streaming E2E Transformer ASR. In the beam search, encoded feature blocks are synchronously aligned using a block boundary detection technique, where a reliability score of each predicted hypothesis is evaluated based on the end-of-sequence and repeated tokens in the hypothesis. Evaluations of the HKUST and AISHELL-1 Mandarin, LibriSpeech English, and CSJ Japanese tasks show that the proposed streaming Transformer algorithm outperforms conventional online approaches, including monotonic chunkwise attention (MoChA), especially when using the knowledge distillation technique. An ablation study indicates that our streaming approach contributes to reducing the response time, and the repetition criterion contributes significantly in certain tasks. Our streaming ASR models achieve comparable or superior performance to batch models and other streaming-based Transformer methods in all tasks considered.

Index Terms: speech recognition, end-to-end, Transformer, self-attention network, knowledge distillation

1. INTRODUCTION

End-to-end (E2E) automatic speech recognition (ASR) has been attracting attention as a method for directly integrating acoustic models and language models (LMs) because of its simple training and efficient decoding procedures. In recent years, various models have been studied, such as connectionist temporal classification (CTC) [1–3], attention-based encoder–decoder models [4–6], their hybrid models [7], and the RNN-transducer [8, 9]. Transformer [10] has been successfully introduced into E2E ASR by replacing RNNs [11–13], and it outperforms bidirectional RNN models in most tasks [14]. Transformer has multihead self-attention network (SAN) layers and source–target attention (STA) layers, which can leverage a combination of information from completely different positions of the input.

However, similarly to bidirectional RNN models [15], Transformer has a drawback in that the entire utterance is required to compute the attentions, making its use in streaming ASR systems difficult. In addition, the memory and computational requirements of Transformer grow quadratically with input sequence length, which makes it difficult to apply to long speech utterances. These problems generally appear when we use SAN, and various works have been recently carried out to tackle these problems for SAN-based acoustic modeling, CTC, and transformer toward streaming ASR [11, 16–18]. These approaches simply introduce blockwise process-

ing for the SAN layers. Miao *et al.* [19] proposed using the previous chunk, inspired by Transformer XL [20]. Furthermore, context-aware inheritance mechanism is also proposed [21]. In that approach, a context embedding vector handed over from the previously processed block helps encode not only local acoustic information, but also global linguistic, channel, and speaker attributes.

In addition to the aforementioned blockwise SAN processing, to realize entire streaming ASR for attention-based models, blockwise processing for STA networks is also required. In [16], a triggered attention mechanism was introduced to realize this. However, it requires a complicated training procedure using CTC forced alignment. Monotonic chunkwise attention (MoChA) [22] is a popular approach to achieve online processing [19, 23–26]. However, MoChA degrades accuracy [24, 26], and it is also difficult to control the latency within an acceptable range.

In this paper, we propose a novel blockwise synchronous beam search algorithm for streaming Transformer to provide an alternative to the MoChA or triggered attention based approaches. The main idea of this algorithm is based on our newly introduced block boundary detection (BBD) technique for decoding after the contextual block encoder used in [21]. The decoder receives encoded blocks one by one from the contextual block encoder. Then, each block is decoded synchronously until an unreliable prediction occurs. Predictions are evaluated on the fly using BBD, where a reliability score of each prediction is computed based on the end-of-sequence token, "⟨eos⟩," and a repetition of a token. Once an unreliable prediction occurs, the decoder waits for the encoder to finish the next block. The main contributions of this paper are summarized as follows. 1) A blockwise synchronous beam search algorithm using BBD is proposed, which is incorporated with the contextual block processing of the encoder in CTC/attention hybrid decoding scheme. 2) Knowledge distillation [27–29] is performed on the streaming Transformer, guided by the original batch Transformer. 3) The proposed streaming Transformer algorithm is compared with conventional approaches including MoChA. The results indicate our approach outperforms them in the HKUST [30] and AISHELL-1 [31] Mandarin, LibriSpeech [32] English, and CSJ [33] Japanese tasks. 4) The impact of each factor in the proposed blockwise synchronous beam search on latency is evaluated through an ablation study.

2. RELATION WITH PRIOR WORK

Among the various available approaches for streaming processing in Transformer, such as time-restricted Transformer [16, 17], Miao *et al.* [19] adopted chunkwise self-attention encoder (Chunk SAE), which was inspired by transformer XL [20], where not only the current chunk but also the previous chunk are used for streaming encoding. Although this encoder is similar to that in [21, 25], in our case, not only the previous chunk but also a long history of chunks is efficiently referred to by introducing context embeddings.

Tian *et al.* [34] applied a neural transducer [35] to the synchronous Transformer decoder, which decodes sequences in a similar manner to the approach proposed in this paper. However, the synchronous Transformer has to be trained using a special forward–backward algorithm similarly to the training of a neural transducer using dynamic programming alignment. In this paper, the proposed beam search algorithm does not require any additional training constraints. Our general decoding algorithm is applied to the parameters as they are. Whereas in [34] the authors only use a ⟨eos⟩ token to synchronously shift the processing blocks, we also take into account a repetition of a token, which significantly improves performance in the LibriSpeech and CSJ tasks.

3. STREAMING TRANSFORMER ASR

3.1. Transformer ASR

Our baseline Transformer ASR follows that described in [14], which is based on an encoder–decoder architecture. An encoder transforms a T-length speech feature sequence $\mathbf{x} = (x_1, \ldots, x_T)$ to an L-length intermediate representation $\mathbf{h} = (h_1, \ldots, h_L)$, where $L \leq T$ owing to downsampling. Given $\mathbf{h}$ and previously emitted character outputs $\mathbf{y}_{0:i-1} = (y_0, \ldots, y_{i-1})$, a decoder estimates the next character y_i.

The encoder consists of two convolutional layers with stride 2 for downsampling, a linear projection layer, and a positional encoding layer, followed by N_e encoder layers and layer normalization. Each encoder layer has a multihead SAN followed by a position-wise feedforward network, both of which have residual connections. In each SAN, attention weights are formed from queries ($\mathbf{Q} \in \mathbb{R}^{t_q \times d}$) and keys ($\mathbf{K} \in \mathbb{R}^{t_k \times d}$) and are applied to values ($\mathbf{V} \in \mathbb{R}^{t_v \times d}$) as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (1)$$

where typically $d = d_{\text{model}}/M$ for the number of heads M. We use multihead attention, denoted as the $\text{MHD}(\cdot)$ function, as follows:

$$\text{MHD}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_M)\mathbf{W}_O^n, \quad (2)$$

$$\text{head}_m = \text{Attention}(\mathbf{Q}\mathbf{W}_{Q,m}^n, \mathbf{K}\mathbf{W}_{K,m}^n, \mathbf{V}\mathbf{W}_{V,m}^n). \quad (3)$$

In (2) and (3), the nth layer is computed with projection matrices $\mathbf{W}_{Q,m}^n \in \mathbb{R}^{d_{\text{model}} \times d}$, $\mathbf{W}_{K,m}^n \in \mathbb{R}^{d_{\text{model}} \times d}$, $\mathbf{W}_{V,m}^n \in \mathbb{R}^{d_{\text{model}} \times d}$, and $\mathbf{W}_O^n \in \mathbb{R}^{Md \times d_{\text{model}}}$. For all the SANs in the encoder, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the same matrices, which are the inputs of each SAN. The position-wise feedforward network is a stack of linear layers.

The decoder predicts the probability of the following character from the previous output characters $\mathbf{y}_{0:i-1}$ and the encoder output $\mathbf{h}$, i.e., $p(y_i|\mathbf{y}_{0:i-1}, \mathbf{h})$. The character history sequence is converted to character embeddings. Then, N_d decoder layers are applied, followed by linear projection and the Softmax function. The decoder layer consists of an SAN and an STA, followed by a position-wise feedforward network. The first SAN in each decoder layer applies attention weights to the input character sequence, where the input sequence of the SAN is set as $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. Then, the subsequent STA attends to the entire encoder output sequence by setting $\mathbf{K}$ and $\mathbf{V}$ to be $\mathbf{h}$.

Transformer can leverage a combination of information from completely different positions of the input. It requires the entire speech utterance for both the encoder and the decoder; thus, they are processed only after the end of the utterance, which causes a huge delay. To realize a streaming ASR system, both the encoder and decoder have to be processed online synchronously.
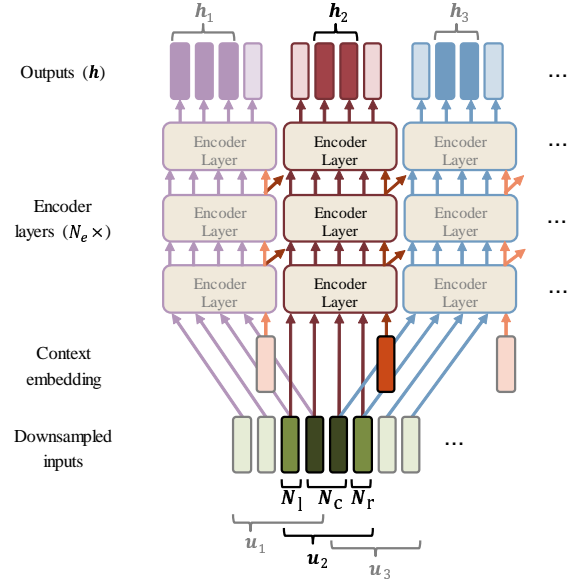

Fig. 1. Context inheritance mechanism of the encoder

3.2. Contextual Block Processing of the Encoder

A simple way to process the encoder online is through blockwise computation, as in [11, 16–19, 35]. However, the global channel, speaker, and linguistic context are also important for local phoneme classification. A context inheritance mechanism for block processing was proposed in [21] by introducing an additional context embedding vector. As shown by the tilted arrows in Fig. 1, the context embedding vector is computed in each layer of each block and handed over to the upper layer of the following block. Thus, the SAN in each layer is applied to the block input sequence using the context embedding vector. A similar idea was also proposed in image and natural language processing around the same time in [36].

Note that the blocks can overlap. In [21], the authors originally proposed a half-overlapping approach, where the central frames of block b, $\mathbf{h}_b$, are computed using the blocked input $\mathbf{u}_b$, which includes past frames as well as looking ahead for future frames. Typically the numbers of frames used for left/center/right in [21] are $\{N_l, N_c, N_r\} = \{4, 8, 4\}$, where the frames are already downsampled by a factor of 4. This can be easily extended to use more frames, such as $\{N_l, N_c, N_r\} = \{16, 16, 8\}$, which are equivalent to the parameters in [19].

3.3. Blockwise Synchronous Beam Search of the Decoder

The original Transformer decoder requires the entire output of the encoder $\mathbf{h}$. Thus, it is not suitable for streaming processing as is. In [25], the authors proposed using MoChA [22], which was tailored for STA. However, accuracy significantly drops when MoChA is applied to decoder layers; this was also observed in other studies [24, 26]. In addition, there is no guarantee that latency stays within established bounds. To avoid these problems, we propose a novel blockwise synchronous beam search algorithm.

3.3.1. Conventional Beam Search of Attention-based ASR

The ordinary beam search with label synchronous decoding of attention-based ASR can be formulated as a problem to find the most probable output sequence $\hat{\mathbf{y}}$ given all the encoded features

$\mathbf{h}_{1:B} = (\mathbf{h}_1, \ldots, \mathbf{h}_B) = \mathbf{h}$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{V}^*} \log p(\mathbf{y}|\mathbf{h}_{1:B}), \qquad (4)$$

where $p(\mathbf{y}|\mathbf{h}_{1:B})$ is computed by the decoder, $\mathcal{V}^*$ represents all possible output sequences, and $\hat{\mathbf{y}}$ is found via a beam search technique.

Let Ω_i be a set of partial hypotheses of length i, and Ω_0 be initialized with one hypothesis with the start-of-sequence token, $y_0 = \langle\text{sos}\rangle$, at the beginning of the beam search. Until $i = I_{\max}$, each partial hypothesis in Ω_{i-1} is expanded by appending possible tokens, i.e., $\mathbf{y}_{0:i} = (\mathbf{y}_{0:i-1}, y_i)$ where $\mathbf{y}_{0:i-1}$ is a partial hypothesis in Ω_{i-1}. Then, new hypotheses are stored in Ω_i and pruned with beam width K, so that only the top-K scored hypotheses survive ($|\Omega_i| = K$).

$$\Omega_i = \text{Search}_K(\Omega_{i-1}, \mathbf{h}_{1:B}) \qquad (5)$$

The score of partial hypothesis $\mathbf{y}_{0:i} \in \Omega_i$ is accumulated in the log domain as

$$\alpha(\mathbf{y}_{0:i}, \mathbf{h}_{1:B}) = \sum_{j=1}^{i} \log p(y_j|\mathbf{y}_{0:j-1}, \mathbf{h}_{1:B}). \qquad (6)$$

In a conventional beam search in attention-based ASR, if y_i is $\langle\text{eos}\rangle$, the hypothesis $\mathbf{y}_{0:i}$ is added to $\hat{\Omega}$, which denotes a set of completed hypotheses. Finally, $\hat{\mathbf{y}}$ is obtained by

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \hat{\Omega}} \alpha(\mathbf{y}, \mathbf{h}_{1:B}). \qquad (7)$$

3.3.2. Blockwise Synchronous Beam Search

Since the decoding problem for ASR does not depend on far-future context information, with a sufficiently high number of blocks $b(< B)$, we assume it can ignore future encoded blocks $\mathbf{h}_{b+1:B}$ and the following approximation is satisfied.

$$\log p(y_i|\mathbf{y}_{0:i-1}, \mathbf{h}_{1:B}) \approx \log p(y_i|\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) \qquad (8)$$

The approximation is more valid when the decoder states attend to the encoded features more locally within b blocks. Thus, the beam search is approximately carried out with the limited features encoded so far ($\mathbf{h}_{1:b}$). While (6) is synchronous to output index i, it can be rewritten to be also synchronous to encoded block b, as
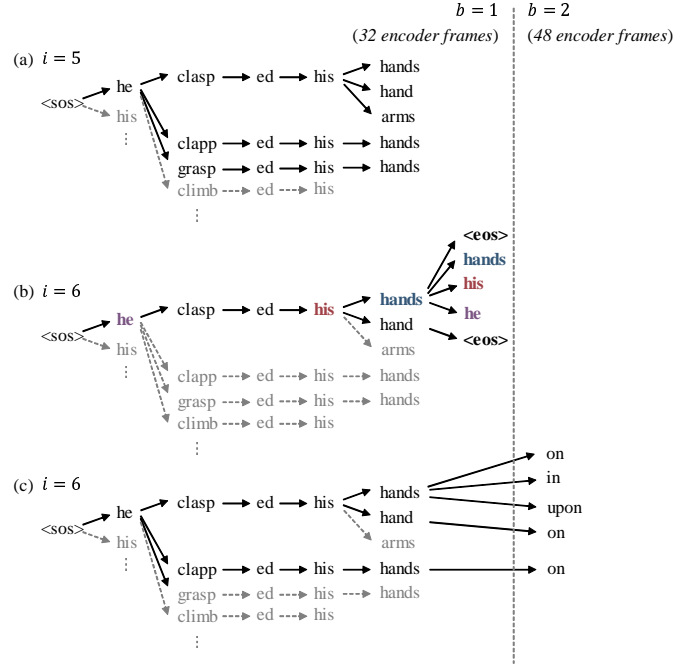
$$\alpha(\mathbf{y}_{0:i}, \mathbf{h}_{1:B}) \approx \sum_{b=1}^{B} \sum_{j=I_{b-1}+1}^{I_b} \log p(y_j|\mathbf{y}_{0:j-1}, \mathbf{h}_{1:b}), \qquad (9)$$

where I_b is an index boundary, which is the last output index for the number of blocks b that satisfies the approximation (8), and $I_0 = 0$. The main idea of this paper is to find appropriate index boundary I_b during beam search.

3.3.3. Block Boundary Detection

When a hypothesis is longer than can be supported by the current encoded data, such a hypothesis is unreliable, because the approximation (8) no longer holds. In such cases, the decoder tends to struggle with two common errors in attention-based ASR described in [7]:

1. It prematurely predicts $\langle\text{eos}\rangle$ as the attentions reach the end of insufficient encoder blocks.

2. It predicts a repeated token because it attends to a position that has already been attended.

Fig. 2. Example of the blockwise synchronous beam search of "He clasped his hands on the desk and said" with a beam width of 5

Therefore, we consider a hypothesis that contains $\langle\text{eos}\rangle$ or a repetition as unreliable with insufficient b encoded blocks. Further, a hypothesis that has lower score than that of the unreliable hypothesis can also be considered as unreliable. We propose a detection technique called BBD, where the index boundaries I_b is found by comparing those scores on the fly.

For convenience, we share the $\langle\text{sos}\rangle$ token with $\langle\text{eos}\rangle$ ($\langle\text{eos}\rangle = \langle\text{sos}\rangle$), so that $\langle\text{eos}\rangle$ is also regarded as a repeat of the $\langle\text{sos}\rangle(= y_0)$ token. When token $y_j \in \mathbf{y}_{0:i-1}$ is repeatedly predicted from $\mathbf{y}_{0:i-1}$, the score is accumulated as $\log p(y_j|\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) + \alpha(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b})$. Thus, the highest score among unreliable hypotheses with a repetition is described as

$$r(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) = \max_{0 \le j \le i-1} \log p(y_j|\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) + \alpha(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}).$$
$$(10)$$

As mentioned above, all the hypothesis with a lower score than $r(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b})$ is considered to be unreliable. We define a reliability score for hypothesis $\mathbf{y}_{0:i}$ as follows.

$$s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) = \alpha(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) - r(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) \qquad (11)$$

Only when $s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) > 0$, the hypothesis is considered to be reliable.

As long as the encoder does not reach the end of the input utterance ($b < B$), each predicted hypothesis is evaluated using the reliability score (11). If it finds an unreliable hypothesis with $s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) < 0$, we assume that this unreliable hypothesis $\mathbf{y}_{0:i}$ is already longer than current index boundary I_b. Our preliminary experiments showed that, when one hypothesis $\mathbf{y}_{0:i}$ contains such $\langle\text{eos}\rangle$ or a repetition, most of the other hypotheses within the same output index i also have the same tendency. Therefore, we can empirically regard that all the hypotheses in i are not considered to satisfy (8) if at least one of the top-K hypotheses is unreliable, i.e., $s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) < 0$. In this way, the index boundary I_b is assigned as the previous output index, i.e., $I_b = i - 1$, and the decoder

waits for the next block, $\mathbf{h}_{b+1}$, to be encoded. The beam search for output index i resumes using hypothesis set Ω_{i-1}, given encoded features $\mathbf{h}_{1:b+1}$. BBD is general so that it is applicable not only to Transformer but also other architectures such as RNNs.

3.3.4. Example of Blockwise Synchronous Beam Search

Figure 2 is an example of a blockwise synchronous beam search of the decoder with beam width $K = 5$. First, the decoder starts with the first encoded block $\mathbf{h}_1$ (length is 32 when $\{N_l, N_c, N_r\} = \{16, 16, 8\}$). As in Fig. 2-(a), hypotheses are predicted from Ω_4 with the limited encoded block and appended. They are then stored in Ω_5 after being pruned.

In Fig. 2-(b), $\langle eos \rangle$ appears in the hypotheses, as well as repetitions ("hands," "his," and "he"). In all cases, the reliability scores (11) are not greater than 0, because all the hypotheses in top-5 score contain repetition and the highest one is $r(\mathbf{y}_{0:4}, \mathbf{h}_1)$. Therefore, the decoder does not store those hypotheses in Ω_6. Instead, the decoder waits for the encoder to output the next block $\mathbf{h}_2$ and resumes decoding from Ω_5 using 48 encoded features $\mathbf{h}_{1:2}$ (Fig. 2-(c)). In this example, the index boundary is assigned as $I_1 = 5$.

3.3.5. Additional Heuristics

Note that the same repetition will not be evaluated again with $b + 1$ blocks because the repetition of a token is most likely correct when it still occurs when sufficient encoder blocks are given. For instance, "$\langle sos \rangle$ - he - clasp - ed - his - hands - he" might be correct if it still occurs with $\mathbf{h}_{1:2}$. Therefore, the hypothesis already evaluated is stored in a set, Ω_R, to prevent it from being reevaluated. In the example in Fig. 2, all hypotheses in (b) are stored in Ω_R. Then, (10) is rewritten by excluding hypotheses in Ω_R as

$$r_{\overline{\Omega}_R}(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) = \max_{\substack{0 \leq j \leq i-1 \\ (\mathbf{y}_{0:i-1}, y_j) \in \overline{\Omega}_R}} \log p(y_j | \mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) + \alpha(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}). \quad (12)$$

The proposed beam search is carried out synchronously as the encoder finishes each block, and thus streaming decoding in Transformer is realized. After the encoder finishes the last block $\mathbf{h}_B$, the beam search continues with all the encoded features $\mathbf{h}_{1:B}$ as usual until the ending criterion is met as described in [7]. The proposed beam search algorithm is summarized in Algorithm 1.

More conservatively, not only Ω_i but also Ω_{i-1} might be considered to contain unreliable hypotheses when $s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) < 0$. In this conservative case, two steps before the output index is assigned to the index boundary, i.e., $I_b = i - 2$ instead of $I_b = i - 1$, as in line 14 of Algorithm 1. In the example shown in Figure 2, the algorithm resumes from hypothesis set Ω_4 instead of Ω_5. This can reduce errors caused by the insufficient encoded features. However, it leads to more overlaps in the decoding process, which reduces computationally efficiency. The effectiveness of conservative decoding is evaluated in our ablation study in Sec. 4.3.

3.3.6. On-the-fly CTC Prefix Scoring

Decoding is carried out jointly with CTC as in [7]. Originally, for each hypothesis, the CTC prefix score is computed as

$$p_{ctc}(\mathbf{y}_{0:i}|\mathbf{h}) = \gamma_T^{(\mathfrak{N})}(\mathbf{y}_{0:i-1}) + \gamma_T^{(\mathfrak{B})}(\mathbf{y}_{0:i-1}), \quad (13)$$

where the superscripts $(\mathfrak{N})$ and $(\mathfrak{B})$ denote CTC paths ending with a nonblank or blank symbol, respectively. Thus, the entire encoded

---

Algorithm 1 Blockwise synchronous beam search of the decoder

Input: encoder feature blocks $\mathbf{h}_b$, total block number B, beam width K
Output: $\hat{\Omega}$: complete hypotheses
1: **Initialize:** $y_0 \leftarrow \langle sos \rangle$, $\Omega_0 \leftarrow \{y_0\}$, $\Omega_R \leftarrow \{\}$, $b \leftarrow 1$, $I_* \leftarrow I_{max}$, $I_0 \leftarrow 0$
2: **while** $b < B$ **do**
3:     NextBlock $\leftarrow false$
4:     **for** $i \leftarrow I_{b-1} + 1$ to I_b unless NextBlock **do**
5:         $\Omega_i \leftarrow \text{Search}_K(\Omega_{i-1}, \mathbf{h}_{1:b})$
6:         **for** $\mathbf{y}_{0:i} \in \Omega_i$ **do**
7:             **if** $s(\mathbf{y}_{0:i}, \mathbf{h}_{1:b}) \leq 0$ **then**
8:                 NextBlock $\leftarrow true$
9:                 $\Omega_R \leftarrow \Omega_R \cup \mathbf{y}_{0:i}$ // store the hypothesis already evaluated
10:            **end if**
11:         **end for**
12:         **if** NextBlock **then**
13:            **if** $i \geq 2$ **then**
14:                $I_b \leftarrow i - 2$ // for conservative decoding
15:            **else**
16:                $I_b \leftarrow i - 1$
17:            **end if**
18:            $b \leftarrow b + 1$ // wait for the next block
19:         **end if**
20:     **end for**
21: **end while**
22: // ordinary decoding follows to obtain $\hat{\Omega}$ after $b = B$
23: **for** $i \leftarrow I_{B-1} + 1$ to I_{max} unless EndingCriterion(Ω_{i-1}) **do**
24:     $\Omega_i \leftarrow \text{Search}_K(\Omega_{i-1}, \mathbf{h}_{1:B})$
25:     **for** $\mathbf{y}_{0:i} \in \Omega_i$ **do**
26:         **if** $y_i = \langle eos \rangle$ **then**
27:            $\hat{\Omega} \leftarrow \hat{\Omega} \cup \mathbf{y}_{0:i}$
28:         **end if**
29:     **end for**
30: **end for**
31: **return** $\hat{\Omega}$

---

features $\mathbf{h}$ is required for accurate computation. However, in the case of a blockwise synchronous beam search, computations are carried out with a limited input length. Therefore, the CTC prefix score is computed from the blocks that are already encoded as follows:

$$p_{ctc}(\mathbf{y}_{0:i}|\mathbf{h}_{1:b}) = \gamma_{T_b}^{(\mathfrak{N})}(\mathbf{y}_{0:i-1}) + \gamma_{T_b}^{(\mathfrak{B})}(\mathbf{y}_{0:i-1}), \quad (14)$$

where T_b is the last frame of the currently processed block b. When a new block output $\mathbf{h}_{b+1}$ is emitted by the encoder, the decoder resumes the CTC prefix score computation according to Algorithm 2 in [7]. Equation (14) incurs a higher computational cost as the input sequence becomes long. However, it can be efficiently computed using a technique described in [37].

3.4. Knowledge Distillation Training

Our preliminary experiments show that parameters trained for the ordinary batch decoder perform well without significant degradation when they are directly used in the blockwise synchronous beam search of the decoder. Therefore, instead of using special dynamic programming or a forward–backward training method as in [34, 35], we propose applying knowledge distillation [27–29] to the streaming Transformer, guided by the ordinary batch Transformer model for further improvement.

Table 1. CERs in the HKUST task

	Dev	Test
Batch processing		
Transformer [14] (reprod.)	24.0	23.5
+ *SpecAugment*	21.2	21.4
Chunk SAE + Batch Dec. [19] (reprod.)	25.8	25.0
CBP-ENC + Batch Dec. [21]	25.3	24.6
+ *SpecAugment*	22.3	22.1
Streaming processing		
CIF + Chunk-hopping [38]	–	23.6
CBP-ENC + MoChA Dec. [25]		
+ *SpecAugment*	28.1	26.1
CBP-ENC + BBD (proposed)		
+ *SpecAugment*	22.6	22.6
+ *Knowledge Distillation*	**22.2**	**22.4**

Table 2. CERs in the AISHELL-1 task

	Dev	Test
Batch processing		
Transformer ($N_e = 6$) [14] (reprod.)	7.4	8.1
CBP-ENC + Batch Dec. ($N_e = 6$) [21]	7.6	8.4
CBP-ENC + Batch Dec. ($N_e = 12$) [21]	6.4	7.2
Streaming processing		
RNN-T [39]	10.1	11.8
Sync-Transformer ($N_e = 6$) [34]	7.9	8.9
CBP-ENC + MoChA Dec. [25]	9.7	9.7
CBP-ENC + BBD ($N_e = 6$, proposed)	7.6	8.5
+ *Knowledge Distillation*	7.6	8.4
CBP-ENC + BBD ($N_e = 12$, proposed)	**6.4**	**7.3**

Table 3. WERs in the LibriSpeech task (Beam width is 30)

	Dev		Test	
	clean	other	clean	other
Batch processing				
ContextNet [42] (SOTA)	2.1	4.6	1.9	4.1
Transformer [14]	2.2	5.6	2.6	5.7
Transformer [14] (reprod.)	2.5	6.3	2.8	6.4
w/ Transforemr LM	2.4	5.9	2.7	6.1
CBP-ENC + Batch Dec. [21]	2.7	7.2	2.9	7.3
Streaming processing				
CBP-ENC + CTC [21]	3.2	9.0	3.3	9.1
CIF + Chunk-hopping [38]	–	–	3.3	9.6
Triggered Attention [16] (large, SOTA)	2.6	7.2	2.8	7.3
CBP-ENC + BBD (proposed)	2.5	6.8	2.7	7.1
w/ Transformer LM	**2.3**	**6.5**	**2.6**	**6.7**

Table 4. CERs in the CSJ task

	eval 1	eval 2	eval 3
Batch processing			
Transformer [14] (reprod.)	5.0	3.7	4.1
CBP-ENC + Batch Dec [21]	5.3	4.0	4.5
Streaming processing			
CBP-ENC + CTC [21]	6.2	4.5	5.2
CBP-ENC + BBD (proposed)	**5.3**	**4.1**	**4.5**

Let $q^{\text{tchr}}(y_i|\mathbf{y}_{0:i-1}, \mathbf{h})$ be a probability distribution computed by a teacher batch model trained with the same dataset, and $p(y_i|\mathbf{y}_{0:i-1}, \mathbf{h})$ be a distribution predicted by the student streaming Transformer model. The latter is forced to mimic the former distribution by minimizing the cross-entropy, which can be written as

$$\mathcal{L}_{\text{KD}} = -\sum_{y_i \in \mathcal{V}} q^{\text{tchr}}(y_i|\mathbf{y}_{0:i-1}, \mathbf{h}) \log p(y_i|\mathbf{y}_{0:i-1}, \mathbf{h}), \quad (15)$$

where $\mathcal{V}$ is a set of vocabulary. The aggregated loss function for the attention encoder and decoder is calculated as

$$\mathcal{L}_{\text{att,KD}} = (1 - \lambda_{\text{KD}})\mathcal{L}_{\text{att}} + \lambda_{\text{KD}}\mathcal{L}_{\text{KD}}, \quad (16)$$

where λ_{KD} is a controllable parameter; typically $\lambda_{\text{KD}} = 0.5$. Then, this loss is combined with CTC loss as in [7].

Note that the knowledge distillation is only applied to the encoder–decoder, i.e., the CTC part for the student is trained by its own. Incorporating with training of the CTC student model would requires a complicated dynamic-programming-like matching algorithm, which is beyond the scope of this paper and left for future work.

4. EXPERIMENTS

4.1. Experimental Setup

We carried out experiments using the HKUST [30] and AISHELL-1 [31] Mandarin tasks, the English LibriSpeech dataset [32], and the Japanese CSJ dataset [33]. The input acoustic features were 80-dimensional filter bank features and the pitch.

For the training process, we used multitask learning with CTC loss as in [7, 14] with a weight of 0.3. A linear layer was added to the encoder to project $\mathbf{h}$ onto the character probability for CTC. The Transformer models were trained using the Adam optimizer

and Noam learning rate decay as in [10]. Decoding was performed alongside CTC, using the proposed beam search algorithm under the conservative condition described in Sec. 3.3.5.

The encoder had $N_e = 12$ layers with 2048 units and the decoder had $N_d = 6$ layers with 2048 units. We set $d_{model} = 256$ and $M = 4$ for the multihead attentions. The input block was overlapped with parameters $\{N_l, N_c, N_r\} = \{16, 16, 8\}$ to enable a comparison with [19], as explained in Sec. 3.2. We trained the contextual block processing encoder (CBP-ENC) with the batch decoder. The parameters for the batch decoder were directly used in the proposed blockwise synchronous beam search algorithm of the decoder using BBD for inference.

Training was carried out using ESPNet [1] [40] with the PyTorch backend.

4.2. ASR Results

4.2.1. HKUST

We used 3655 character classes with a CTC weight of 0.3 and a beam width of 10. Shallow fusion of a two-layer LSTM LM with 650 units was applied with a weight of 0.3. For comparison, we implemented Chunk SAE [19], which is similar to our CBP-ENC approach except that it does not use the contextual embedding procedure introduced in Section 3.2. Though we were unable to reproduce the original score in [19], the implemented model performed reasonably well.

The results are listed in Table 1. By comparing CBP-ENC with Chunk SAE, we can confirm that our contextual embedding approach performed better, in both cases where the batch decoder was used. SpecAugment [41] resulted in further improvement. For streaming processing, we obtained better performance by combining CBP-ENC and BBD rather than CBP-ENC and the MoChA decoder [25]. The knowledge distillation training in Sec. 3.4 further improved its performance. The proposed method achieved state-of-the-art performance as a streaming E2E approach.

---

[1] The training and inference implementations are publicly available at https://github.com/espnet/espnet.

Table 5. Ablation study and computational speed comparison with C++ CPU implementation (Beam width is 10)

	HKUST			CSJ			Librispeech (large LM)		
	CER	RTF	Response	CER (eval1/eval2/eval3)	RTF	Response	WER (clean/other)	RTF	Response
Average utterance length		4.9s			4.9s			9.1s	
Batch Transformer [14] (reprod.)	21.4	0.07	0.31s	5.0% / 3.7% / 4.1%	0.16	0.74s	2.9% / 6.7%	0.36	3.49s
CBP-ENC + Batch Dec [21]	22.1	0.08	0.31s	5.3% / 4.0% / 4.5%	0.17	0.71s	2.8% / 7.4%	0.33	2.81s
CBP-ENC + BBD (proposed)	22.4%	0.09	0.23s	5.3% / 4.1% / 4.5%	0.17	0.52s	3.0% / 7.8%	0.35	1.19s
- conservative decoding	22.8%	0.09	0.19s	5.5% / 4.2% / 4.8%	0.17	0.50s	5.3% / 10.6%	0.35	1.08s
- repetition	25.4%	0.08	0.15s	28.6% / 29.5% / 24.8%	0.17	0.32s	32.3% / 39.8%	0.35	0.71s

4.2.2. AISHELL-1

For this task, 4231 character classes were used with parameters {CTC weight, beam width, LM weight} = {0.5, 10, 0.7}. To make a comparison with Sync-Transformer [34] possible, we trained a smaller Transformer with $N_e = 6$. The results are shown in Table 2. Additionally, the results for RNN-T evaluated in [39] are listed. As can be seen in the results, our approach outperformed both the MoChA decoder and Sync-Transformer [34], especially when we applied the knowledge distillation.

4.2.3. LibriSpeech

For LibriSpeech, we adopted byte-pair encoding (BPE) subword tokenization [43], which had 5000 token classes. In addition to a large LM (four-layer LSTM with 2048 units), we evaluated the use of a Transformer LM (16-layer transformer LM with 2048 units and 8 heads); both were fused with a weight of 0.6. CTC weight and beam width were set as 0.4 and 30. SpecAugment [41] was also applied when it was trained.

The results are shown in Table 3. Though we did not use a large model as in [14, 16], we obtained similar results. The proposed method achieved better performance than CTC decoding [21] and continuous integer-and-fire (CIF) online E2E ASR [38], which indicats that our blockwise synchronous beam search also works with BPE tokenization. Even with the LSTM LM, we also achieved comparable performance to state-of-the-art streaming E2E ASR using triggered attention [16], which was a model twice as large as ours. Note that there is still room to improve accuracy, since our reproduction of [14] was not as well tuned as the original paper.

4.2.4. CSJ

CSJ data had 3260 character classes. The parameters were set as {CTC weight, beam width, LM weight} = {0.3, 10, 0.3}, and a two-layer LSTM LM with 650 units was fused. SpecAugment [41] was used for data augmentation. The results are shown in Table 4. The proposed method outperformed a CTC-based streaming approach [21], and also did not degrade significantly from the batch Transformer.

4.3. Ablation Study and Computational Speed Comparison

We carried out an ablation study to evaluate how each factor contribute to both accuracy and computational efficiency. HKUST, LibriSpeech, and CSJ were used. To evaluate error rates, beam widths were fixed at 10. To evaluate computational speed, we implemented the proposed beam search algorithm in C++, and subset of each task was used. We used Intel Math Kernel Library to perform matrix operations with CPUs. To avoid redundant computation in the decoder, we applied caching techniques similarly to [44, 45], which reduce the real-time factor (RTF) of RNN-T computations from 0.89 to 0.61 in [45]. For LibriSpeech, a large LM (four-layer LSTM with 2048

units) was used as described in Sec. 4.2.3. The latency during the utterance was not evaluated in this study because the alignment between the input and the output sequence was not provided. The theoretical delay was 0.64 seconds because the encoder block shifted every 16 frames with 4-factor downsampling. Instead, we measured the response time, which was the time required to finish decoding after the end of each utterance. RTF and response time were measured with an 8 core 3.60 GHz Intel i9-9900K processor.

The results are shown in Table 5. The RTFs of the batch Transformer were smaller than those of the streaming Transformer for HKUST and CSJ, because the proposed streaming Transformer processed with overlaps. As for LibriSpeech, the RTF of the batch Transformer was greater than streaming because utterance length were longer (9.1 s on average), which had a quadratic-order effect. In addition, only LibriSpeech was used with a larger LM. Therefore, its response time was greater than that for other tasks. The response times of the streaming Transformer were shorter for all task owing to its efficient blockwise beam search. Whereas the differences in performance were small for the HKUST and CSJ tasks, in which the utterances were generally short, the relative improvement observed for the LibriSpeech task was significant due to the longer utterances. When the decoding process was carried out without the conservative approach described in Sec. 3.3.5, the error rates slightly increased for the LibriSpeech because BBD failed to detect the block boundary, while the response times improved. We also performed an ablation study to evaluate the repetition criterion by modifying (10) as

$$r'(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}) = \log p(\langle \text{eos} \rangle | \mathbf{y}_{i-1}, \mathbf{h}_{1:b}) + \alpha(\mathbf{y}_{0:i-1}, \mathbf{h}_{1:b}), \tag{17}$$

which only evaluated $\langle \text{eos} \rangle$ as in [34, 35]. The results indicate that the repetition of tokens is an important criterion for the blockwise synchronous beam search, because the error rates significantly increased without them, dramatically in the CSJ and LibriSpeech tasks.

5. CONCLUSIONS

We proposed a new blockwise synchronous beam search algorithm based on a blockwise processing of encoder to achieve streaming E2E Transformer ASR. A block boundary detection technique was proposed, where a reliability score is computed based on $\langle \text{eos} \rangle$ and repeated tokens in the hypotheses. Using this technique, each prediction is judged as either reliable or unreliable using the current limited number of blocks from the encoder. If a prediction is deemed unreliable, the decoder waits for the encoder to finish the next block. Evaluations of the HKUST and AISHELL-1 Mandarin, LibriSpeech English, and CSJ Japanese tasks showed that the proposed streaming Transformer outperforms conventional online approaches including MoChA, especially when using the knowledge distillation technique. The algorithm is general so that future work is to apply it also to the latest architecture such as Conformer [46].

6. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. of 23rd International Conference on Machine Learning*, 2006, pp. 369–376.

[2] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. of ASRU Workshop*, 2015, pp. 167–174.

[3] Dario Amodei et al., "Deep Speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. of 33rd International Conference on Machine Learning*, 2016, vol. 48, pp. 173–182.

[4] Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Proc. of NIPS*, 2015, pp. 577–585.

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

[6] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. of ICASSP*, 2018, pp. 4774–4778.

[7] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

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

[9] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. of ASRU Workshop*, 2017, pp. 193–199.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. of NeurIPS*, 2017, pp. 5998–6008.

[11] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel, "Self-attentional acoustic models," in *Proc. of Interspeech*, 2018, pp. 3723–3727.

[12] Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *Proc. of ICASSP*, 2019, pp. 7115–7119.

[13] Yuanyuan Zhao, Jie Li, Xiaorui Wang, and Yan Li, "The SpeechTransformer for large-scale Mandarin Chinese speech recognition," in *Proc. of ICASSP*, 2019, pp. 7095–7099.

[14] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al., "A comparative study on transformer vs RNN in speech applications," in *Proc. of ASRU Workshop*, 2019, pp. 449–456.

[15] Mike Schuster and Kuldip K. Paliwal, "Bidirectional recurrent neural networks," *Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[16] Niko Moritz, Takaaki Hori, and Jonathan Le Roux, "Streaming automatic speech recognition with the transformer model," in *Proc. of ICASSP*, 2020, pp. 6074–6078.

[17] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur, "A time-restricted self-attention layer for ASR," in *Proc. of ICASSP*, 2018, pp. 5874–5878.

[18] Linhao Dong, Feng Wang, and Bo Xu, "Self-attention aligner: A latency-control end-to-end model for ASR using self-attention network and chunk-hopping," in *Proc. of ICASSP*, 2019, pp. 5656–5660.

[19] Haoran Miao, Gaofeng Cheng, Zhang Pengyuan, and Yonghong Yan, "Transformer online CTC/attention end-to-end speech recognition architecture," in *Proc. of ICASSP*, 2020, pp. 6084–6088.

[20] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[21] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe, "Transformer ASR with contextual block processing," in *Proc. of ASRU Workshop*, 2019, pp. 427–433.

[22] Chung-Cheng Chiu and Colin Raffel, "Monotonic chunkwise attention," *arXiv preprint arXiv:1712.05382*, 2017.

[23] Ruchao Fan, Pan Zhou, Wei Chen, Jia Jia, and Gang Liu, "An online attention-based model for speech recognition," *Proc. of Interspeech*, pp. 4390–4394, 2019.

[24] Kwangyoun Kim, Kyungmin Lee, Dhananjaya Gowda, Junmo Park, Sungsoo Kim, Sichen Jin, Young-Yoon Lee, Jinsu Yeo, Daehyun Kim, Seokyeong Jung, et al., "Attention based on-device streaming speech recognition with large speech corpus," in *Proc. of ASRU Workshop*, 2019, pp. 956–963.

[25] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe, "Towards online end-to-end transformer automatic speech recognition," *arXiv preprint arXiv:1910.11871*, 2019.

[26] Hirofumi Inaguma, Yashesh Gaur, Liang Lu, Jinyu Li, and Yifan Gong, "Minimum latency training strategies for streaming sequence-to-sequence ASR," in *Proc. of ICASSP*, 2020, pp. 6064–6068.

[27] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong, "Learning small-size DNN with output-distribution-based criteria," in *Proc of 15th Annual Conference of the International Speech Communication Association*, 2014.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[29] Liang Lu, Michelle Guo, and Steve Renals, "Knowledge distillation for small-footprint highway networks," in *Proc. of ICASSP*, 2017, pp. 4820–4824.

[30] Yi Liu, Pascale Fung, Yongsheng Yang, Christopher Cieri, Shudong Huang, and David Graff, "HKUST/MTS: A very large scale Mandarin telephone speech corpus," in *International Symposium on Chinese Spoken Language Processing*. Springer, 2006, pp. 724–735.

[31] Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng, "AIShell-1: An open-source Mandarin speech corpus and a speech recognition baseline," in *Oriental COCOSDA*, 2017, pp. 1–5.

[32] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *Proc. of ICASSP*, 2015, pp. 5206–5210.

[33] Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara, "Spontaneous speech corpus of Japanese," in *Proc. of the International Conference on Language Resources and Evaluation (LREC)*, 2000, pp. 947–9520.

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

[35] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio, "An online sequence-to-sequence model using partial conditioning," in *Proc. of NIPS*, 2016, pp. 5067–5075.

[36] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[37] Hiroshi Seki, Takaaki Hori, Shinji Watanabe, Niko Moritz, and Jonathan Le Roux, "Vectorized beam search for ctc-attention-based speech recognition," in *Proc. of Interspeech*, 2019, pp. 3825–3829.

[38] Linhao Dong and Bo Xu, "CIF: Continuous integrate-and-fire fore end-to-end speech recognition," in *Proc. of ICASSP*, 2020, pp. 6079–6083.

[39] Zhengkun Tian, Jiangyan Yi, Jianhua Tao, Ye Bai, and Zhengqi Wen, "Self-attention transducers for end-to-end speech recognition," in *Proc. of Interspeech*, 2019, pp. 4395–4399.

[40] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., "ESPnet: End-to-end speech processing toolkit," in *Proc. of Interspeech*, 2019, pp. 2207–2211.

[41] 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," in *Proc. of Interspeech*, 2019.

[42] Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," *arXiv preprint arXiv:2005.03191*, 2020.

[43] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *Proc. of the Association for Computational Linguistics*, 2016, vol. 1, pp. 1715–1725.

[44] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *Proc. of ICASSP*, 2019, pp. 6381–6385.

[45] George Saon, Zoltán Tüske, and Kartik Audhkhasi, "Alignment-length synchronous decoding for RNN transducer," in *Proc. of ICASSP*, 2020, pp. 7804–7808.

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