Interspeech 2022
18-22 September 2022, Incheon, Korea

Streaming Automatic Speech Recognition with Re-blocking Processing Based on Integrated Voice Activity Detection

Yui Sudo[1], *Muhummad Shakeel*[2], *Kazuhiro Nakadai*[1,2], *Jiatong Shi*[3], *Shinji Watanabe*[3]

[1]Honda Research Institute Japan Co., Ltd., Saitama, JAPAN
[2]Department of Systems and Control Engineering, School of Engineering, Tokyo Institute of Technology, Tokyo, JAPAN
[3]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA

{yui.sudo, nakadai}@jp.honda-ri.com, shakeel@ra.sc.e.titech.ac.jp, {jiatongs, shinjiw}@cs.cmu.edu

Abstract

This paper proposes streaming automatic speech recognition (ASR) with re-blocking processing based on integrated voice activity detection (VAD). End-to-end (E2E) ASR models are promising for practical ASR. One of the key issues in realizing such a system is the detection of voice segments to cope with streaming input. There are three challenges for speech segmentation in streaming applications: 1) the extra VAD module in addition to the ASR model increases the system complexity and the number of parameters, 2) inappropriate segmentation of speech for block-based streaming methods deteriorates the performance, 3) non-voice segments that are not discarded results in the increase of unnecessary computational costs. This paper proposes a model that integrates a VAD branch into a block processing-based streaming ASR system and a re-blocking technique to avoid inappropriate isolation of the utterances. Experiments show that the proposed method reduces the detection error rate (ER) by 25.8% on the AMI dataset with a less than 1% of increase in the number of parameters. Furthermore, the proposed method show 7.5% relative improvement in character error rate (CER) on the CSJ dataset with 27.3% reduction in real-time factor (RTF).

Index Terms: speech recognition, end-to-end, streaming, voice activity detection, blockwise processing

1. Introduction

End-to-end automatic speech recognition (E2E-ASR) is direct mapping from acoustic feature vector sequences to token sequences, obviating resource hungry elements such as morphological analysis and pronunciation dictionaries utilized in traditional ASR systems. Currently, there are four main types of network architectures for E2E-ASR: connectionist temporal classification (CTC) [1, 2], attention mechanism [3, 4], CTC/attention [5], and recurrent neural network transducer (RNN-T) [6, 7]. Although the performance of E2E-ASR systems has been improving, most research setups assume the recorded input speech is appropriately segmented into shorter utterances. Effective segmentation of speech utterances among streaming sound input is important in ASR applications. It is desirable to start recognizing without waiting for the end of the utterance and segment the audio input into short utterances of appropriate length to reduce response time.

Various VAD methods have been proposed, from simple detection of change in signal energy to statistical approaches such as hidden Markov model and Gaussian mixture model [8, 9, 10]. Other proposals include deep neural network-based (DNN) methods using multilayer perceptron (MLP) [11, 12], long-short-term memory (LSTM) [13, 14], convolutional neural network (CNN) [15], and Transformer [16, 17], all of which have performed better than traditional methods. Although VAD is computationally inexpensive, increasing the number of modules in ASR system increases its complexity. Several attempts have been made to integrate VAD into E2E-ASR [18, 19, 20, 21]. For example, an integrated VAD method using the blank output of a CTC-based E2E ASR model was proposed [20, 21], however, it lacked optimal performance because it was not trained for VAD.

Regarding streaming ASR systems, RNN-T was investigated with unidirectional LSTM and later extended with causal Transformer/Conformer networks [6, 7, 22, 23]. Another method of streaming ASR systems involves blockwise processing, which efficiently restricts the length of the attention window in Transformer. This method has been realized in hidden Markov model based systems [24, 25], RNN-T [26, 27], attention [28], and CTC/attention [29, 30, 31, 32]. In this paper, we focus on blockwise streaming processing based on CTC/attention leveraging its potential to handle VAD based on the CTC branch [20], as discussed above. Especially, we focus on CTC/attention Streaming Transformer [31, 32], which retains the context of long utterances across the block by using an additional context-embedding vector that inferits previous encoder states. However, if the speech is improperly segmented for each complete utterance, the inherited context is discarded, resulting in performance degradation.

This paper proposes a model that integrates VAD branch into a block processing-based streaming ASR. In our proposed method, to minimize the number of parameters, ASR and VAD share an encoder. The pretrained ASR parameters are frozen to avoid degrading the performance of the pretrained ASR. To segment each utterance at an appropriate length in a block processing-based approach, we also propose a re-blocking technique for encoder hidden states based on VAD branches. In summary, this paper has the following contributions,

- Integration of VAD with streaming ASR avoids increasing system complexity.
- Re-blocking technique for block-based streaming ASR segments utterances while retailing context.
- Discarding non-voice segments by VAD reduces computational costs.

2. Preliminary

This section describes the Streaming Transformer ASR [31, 32] which is the basis of the proposed blockwise method.

10.21437/Interspeech.2022-11216

2.1. Attention-based Transformer ASR

Attention-based Transformer ASR [33] has an encoder-decoder architecture. The encoder usually comprises of two convolutional layers for downsampling, a linear projection layer, and a positional encoding layer, followed by N_e Transformer blocks. The convolutional layers subsample a T-length audio feature sequence, $\boldsymbol{X} = [x_1, ..., x_T]$ into a subsampled feature sequence, $\boldsymbol{u} = [u_1, ..., u_L]$, where L represents the subsampled length ($L < T$) as:

$$\boldsymbol{u} = \mathrm{ConvSubsamp}(\boldsymbol{X}). \tag{1}$$

Then, the Transformer blocks transform the subsampled feature sequence $\boldsymbol{u}$ to an L length hidden states $\boldsymbol{h} = [h_1, ..., h_L]$as:

$$\boldsymbol{h} = \mathrm{TrEncoder}(\boldsymbol{u}). \tag{2}$$

Each Transformer block has a multiheaded self-attention layer, a linear layer, and a layer-normalization layer, with residual connections.

Given $\boldsymbol{h}$ and previously estimated token sequence $\boldsymbol{y}_{s-1} = [y_0, ..., y_{s-1}]$, the decoder estimates the next token y_s. This process is recursively performed as:

$$y_s = \mathrm{TrDecoder}(\boldsymbol{h}, \boldsymbol{y}_{s-1}). \tag{3}$$

Token history $\boldsymbol{y}_{s-1}$ is first converted to token embeddings. These are then fed into N_d decoder layers with hidden states $\boldsymbol{h}$, followed by a linear projection. Given outputs of the linear projection, the predicted probability of y_s is given by a softmax function. The decoder layer comprises a self-attention network and an source-target attention, followed by a position-wise feed-forward network.

2.2. Contextual Block Processing of the Encoder

Real-time ASR applications must perform recognition online-as continuous speech is streamed along with noise. To achieve such online processing in the Transformer framework, the encoder may be computed blockwise as described in [31]. Let L_{block} and L_{hop} denote the block size and the hop length, respectively. The b-th block is defined as input features u_t from $t = (b-1)L_{\text{hop}} + 1$ to $t = (b-1)L_{\text{hop}} + L_{\text{block}}$, which is denoted as an L_{block}-length subsequence $\boldsymbol{u_b}$,

$$\boldsymbol{u}_b = (u_{(b-1)L_{\text{hop}}+1}, ..., u_{(b-1)L_{\text{hop}}+L_{\text{block}}}) \tag{4}$$

Corresponding b-th hidden states, $\boldsymbol{h}_b$ are encoded for each block containing L_{block}-length hidden states, based on the predetermined Transformer encoder TrEncoder, described as,

$$\boldsymbol{h}_b = \mathrm{BlockTrEncoder}(\boldsymbol{u}_b). \tag{5}$$

2.3. Blockwise Synchronous Beam Search of the Decoder

In the blockwise streaming method, the decoder must perform real-time processing with block processing as well to achieve real-time recognition [32]. Online beam search with label-synchronous decoding for attention-based ASR can be defined as: the problem of finding the most probable output sequence $\hat{\boldsymbol{y}}$ given the hidden states block encoded, $\boldsymbol{h}_{1:b} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_b)$, at that time. The procedure can be formulated as:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in \mathcal{V}^*}{\mathrm{argmax}}(\log(p(\boldsymbol{y}|\boldsymbol{h}_{1:b}))), \tag{6}$$

where $p(\boldsymbol{y}|\boldsymbol{h_{1:b}})$ is computed by the decoder, $\mathcal{V}^*$ represents all possible output sequences, and, practically, $\hat{\boldsymbol{y}}$ is found by beam search technique. Note that all hidden states from the start of the utterance to the current block are required during decoding. In other words, the longer the utterance, the longer the decoding time. Therefore, the hidden states sequence should be divided into appropriately short utterances prior to decoding.

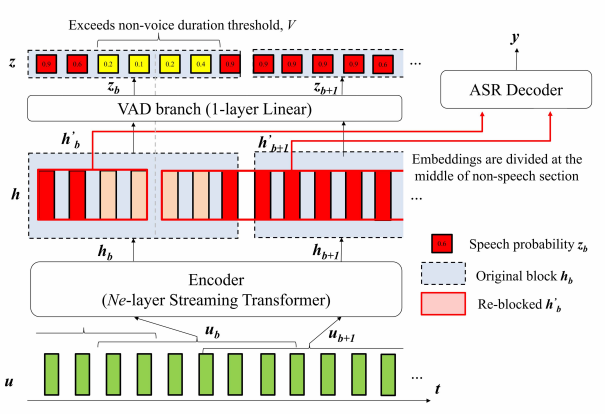

Figure 1: *Architecture of the proposed VAD model integrated with blockwise processing-based ASR model. The ASR decoder and VAD branch share the encoder of a pretrained ASR system. During encoding, the hidden states sequence h_b is output based on the predetermined block size, L_{block}. During decoding, the hidden states blocks, $\boldsymbol{h_b}$, are re-blocked to $\boldsymbol{h'_b}$ at the center index of the non-voice segments detected by the VAD branch.*

3. Proposed method

Figure 1 shows the overall architecture of the proposed model comprising of a shared encoder, an ASR decoder, and a VAD branch. The model uses no external VAD module. Its VAD and ASR system share an encoder to minimize the number of parameters. The model was trained in two stages. First, a conventional E2E-ASR system was trained, then the VAD branch was trained while freezing the pretrained ASR system. We used CTC/Attention-based streaming Transformer described in Section 2 [32]. Although this method could optimize both VAD and ASR branches simultaneously by multitask learning, it is generally time-consuming to search for optimal weights used in the loss function. In addition, multitask learning may impair performance in existing ASR model. The proposed strategy has the advantage of not impairing the pretrained ASR model. During inference, the block size of the hidden states sequence, $\boldsymbol{h}_b$, is adjusted by the re-blocking process such that an unsegmented speech is divided appropriately into short utterances. The re-blocking technique is described in Section 3.2.

3.1. VAD branch

VAD is formulated as a single-label classification problem. We used a linear layer for the VAD branch. The VAD branch outputs the probability of the presence of a voice segment using the output $\boldsymbol{h}$ in Eq. (2) of the shared ASR encoder. Then, framewise posterior probabilities of the speech activities, $\boldsymbol{z} = [z_1, ..., z_L]$ are estimated using a fully connected layer with a Sigmoid function. The length of speech probability array $\boldsymbol{z}$ equals that of hidden states, $\boldsymbol{h}$. The system estimates a sequence of speech activities depending on this speech probability. If a posterior probability z_t at time index t, we determine presence of speech, if it is above a certain threshold, $p = 0.5$, otherwise absence. Additionally, a threshold V which represents the number of con-

secutive non-voice segments is introduced to prevent detection of short non-voice segments within an utterance. If the number of consecutive non-voice segments exceeds V, those segments are considered non-voice and the encoder/decoder history is reset.

The loss function was binary CrossEntropy between the target output $\boldsymbol{z}_t$ and the estimated output $\hat{\boldsymbol{z}}_t$. Target output $\boldsymbol{z}_t$ was obtained based on the temporal information provided by the dataset.

3.2. Re-blocking hidden states sequence based on VAD

As described in Section 2, in the Streaming Transformer, hidden states sequences are processed blockwisely. Blockbased methods, it may not be possible to properly segment an utterance if the hidden states sequence is simply divided by blocks. Figure 1 shows an example for the consecutive non-voice duration threshold, V = 4. VAD is performed for each block with L_{block} frames of hidden states and outputs L_{block} frames of speech probabilities. The b-th block contains four consecutive non-voice segments, shown in yellow. However, since the b-th block contains part of the beginning of another utterance, a simple segmentation by pre determined block would result in a split in the middle that would break the context of the next utterance. During decoding, therefore, the hidden states block is resized at the center of the detected non-voice segments, C-th hidden state, before transmission to the ASR decoder. We call this operation, Re-blocking. The re-blocked current hidden states sequence is described as $\boldsymbol{h}_b^{'} = [h_{b_1}, ..., h_{b_C}]$, where h_{b_1} denotes the first hidden state and h_{b_C} denotes the C-th hidden state of b-th block ($C < L_{block}$). Any unused hidden states sequence of the current b-th block is stacked into the next hidden states sequence, $\boldsymbol{h}_{b+1}$, described as,

$$\boldsymbol{h}_{b+1}^{'} = \text{concat}([h_{b_{C+1}}, ..., h_{b_E}], \boldsymbol{h}_{b+1}), \tag{7}$$

where $h_{b_{C+1}}$ denotes the next hidden state of the center of the non-voice segments and h_{b_E} denotes the last hidden state of b-th block.

4. Experiments

To validate the three contributions described in Section 1, we evaluate the proposed method with respect to its VAD performance, the impact on ASR, and its computation time.

4.1. Experimental setup

The input feature consisted of 80-dimensional mel-scale filterbank features using a window size of 512 samples and a hop length of 128 samples. Then, SpecAugment [34] was applied. The encoder consists of two convolutional layers with stride 2 and 3 for downsampling, a 512-dim linear projection layer, and a positional encoding layer, followed by $N_e = 12$ Transformer layers with 2048 linear units and layer normalization. The decoder had $N_d = 6$ layers with 2048 units. We set the attention dimension size as 256 with 4-multihead attentions. The L_{block} and L_{hop} described in Section 2.2 were 40 and 16, respectively. A linear layer was applied to the VAD branch. The number of parameters of the pretrained ASR model was 30.3 M.

In the first stage, the ASR model training, we used multitask learning with a CTC loss as in [5] with a weight of 0.3. A linear layer was added to the end of the encoder to project $\boldsymbol{h}$ into the token probability for CTC. The Transformer models were trained using the Adam optimizer. 40 epochs were trained at a learning rate of 0.005 with warmup steps of 25000 on the CSJ corpus and 80 epochs were trained at a learning rate of 0.005 with warmup steps of 25000 for the TED-LIUMv2 corpus.

In the second stage, the VAD branch was trained using the Adam optimizer, and 30 epochs were trained at a learning rate of 0.00001 with warmup steps of 10000 for all datasets.

Table 1: *ER results of VAD task*

Model	Params	AMI	CSJ	TED-LIUMv2
External VAD	4.45 M	5.7	4.7	7.1
CTC-based	0	9.3	19.2	5.7
Internal VAD	0.8 K	6.9	5.7	5.5

4.2. Evaluation of VAD task

The proposed method was compared with a conventional external VAD model and CTC-based VAD [20] in a VAD task to confirm it could detect voice segments. An external VAD was a neural VAD model that was not integrated with ASR while CTC-based VAD is an integration method of VAD and ASR. The purpose of this experiment was to confirm that Internal VAD works before evaluating it on ASR task. Although, CTC-based VAD was originally proposed for LSTM-based models, we applied it to Streaming Transformer. For evaluation, we employed AMI, CSJ, and TED-LIUMv2 [35, 36, 37] corpuses, which were suitable for the VAD task as they contain recordings of over 20 min duration. The result of CSJ in Table 1 shows the average of three sets of eval1, eval2, and eval3.

4.2.1. Metrics for VAD

VAD system performance was measured using detection error rate (ER) where ER is the sum of two different error types: false alarm of speech and missed detection of speech. False alarm, *F*(*t*), are the number of reference events that are incorrectly identified, and missed detection of speech, *M*(*t*), are the number of events in system output that are missed. Total ER is calculated by integrating framewise counts over the total number of time-frames *T*, with *N*(*t*) being the number of voice segments marked as active in the reference in time-frame *t*:

$$ER = \frac{\sum_{t=1}^{T} M(t) + \sum_{t=1}^{T} F(t)}{\sum_{t=1}^{T} N(t)}, \tag{8}$$

4.2.2. Results

Table 1 shows the number of parameters and ERs (%) required for the VAD branch. While the External VAD required 4.45M parameters, the proposed method integrated Internal VAD, which shares the ASR encoder, required much fewer. CTC-based VAD did not increase the number of parameters at all, however, the ERs on the AMI and CSJ dataset were much worse than the proposed method, whereas the proposed method showed better ERs for all datasets than CTC-based method. Although the proposed method had an average 27% worse ER compared to External VAD, VAD and ASR can be performed simultaneously with only a 1% increase in the number of parameters.

4.3. Evaluation of ASR task

We evaluated the proposed method in ASR task. Two datasets, CSJ and TED-LIUMv2, [36, 37] were used to test the proposed method. In our experiments, we used ESPnet as the E2E-ASR toolkit [38]. The following three methods were compared,

Table 2: *Performance comparison in CER/WER*

Corpus (Percentage of speech frames)	CSJ eval1 (80%)	CSJ eval2 (81%)	CSJ eval3 (77%)	TED-LIUMv2 (95%)
Oracle	5.9	4.2	4.6	9.8
Baseline (L_{th}=0)	15.1	12.9	14.1	22.44
Baseline (L_{th}=300)	10.1	7.4	7.9	**14.9**
Baseline (L_{th}=500)	27.1	22.8	23.0	21.18
CTC-based (uni-LSTM) [20]		18.9 (ave.)		18.9
Proposed	**9.1**	**6.7**	**7.7**	15.0

- Oracle: The speech input was segmented according to the temporal information provided by the dataset.
- Baseline: The speech input was automatically divided into each utterance when the length of the hidden states exceeds the length threshold of hidden states sequence L_{th}. In other words, the speech input was segmented at equal time intervals regardless of voice activity.
- Proposed: The speech input was automatically segmented based on the proposed method. Then hidden states blocks were re-blocked based on the VAD results described in Section 3.2. The E2E model was trained on segmented data provided by the dataset. Due to rare long utterances, hidden states were also automatically segmented even when the length of the hidden states exceeded L_{th}=300.

Decoding was performed using the blockwise beam search algorithm. Beam width 20 was used during inference. Character error rate (CER) and word error rate (WER) were calculated for CSJ and TED-LIUMv2, respectively.

4.3.1. Performance comparison

Table. 2 shows CER/WER results by dataset. At baseline, CER was smallest when L_{th}=300. When L_{th}=0, it means that the utterance was segmented every single block regardless of the voice activity. When L_{th} is small, the length of one segment becomes shorter which causes performance degradation because the context of the utterance is broken. In contrast, when L_{th} is large, the length of one segment becomes longer. As reported in [5], long utterance causes performance degradation in Attention-based ASR methods.

The proposed method showed a CER reduction of about 10% compared to the baseline for CSJ. While the baseline divides the speech input at equal intervals regardless of voice activity, the proposed method reduces CER by dividing it at appropriate non-voice segments. The proposed method reduced CER by about 50% compared to the results of the CTC-based integrated VAD method (reported in [20]).

In contrast, the proposed method did not improve the WER on TED-LIUMv2 evaluation set. According to the temporary information provided by the datasets, the CSJ evaluation set had an average speech interval of 79%, while that of the TED-LIUMv2 evaluation set was 95%. Since 95% of the speech signal in TED-LIUM2 was voice segments, the proposed method could not detect consecutive non-voice segments.

4.3.2. Effect of Re-blocking

The effect of re-blocking of the proposed method in Section 3.2 was then verified. Table. 3 shows the comparison between with or without re-blocking. The speech input was automatically segmented using the proposed method for both with and without re-blocking, but the hidden state blocks were not re-blocked for without re-blocking. With re-blocking resulted in a smaller CER than the baseline, however, the CER was worse than the baseline without re-blocking. The reason is that the context of speech across multiple blocks is divided for without re-blocking. Therefore, we found that the proposed re-blocking technique is effective for block-based streaming ASR models by preventing the context of an utterance from being inappropriately divided.

Table 3: *Effect of re-blocking*

Method	eval1	eval2	eval3
Baseline (L_{th}=300)	10.1	7.4	7.9
w/o re-blocking	10.4	7.7	11.7
w/ re-blocking	**9.1**	**6.7**	**7.7**

Table 4: *RTF comparison for different thresholds*

Threshold	eval1	eval2	eval3	RTF
Baseline (L_{th}=300)	10.1	7.4	7.9	0.55
V = 5	9.4	7.1	8.3	**0.35**
V = 10	**9.1**	**6.7**	**7.7**	0.40
V = 16	9.4	7.2	**7.7**	0.48

4.3.3. Inference speed

We measured real time factors (RTF) using a GPU (NVIDIA A100-SXM4-40GB). Table 4 shows the CERs and RTFs for different consecutive non-voice segments thresholds, V, in the proposed method. As V increases, the RTF increases because the segmented utterances become longer, which increases the computational complexity during decoding. Conversely, decreasing V resulted in shorter decoding times due to the shorter length of each utterance, but slightly worse CER due to the disconnection of the context. Both conditions showed comparable or better CERs with smaller RTFs than the baseline (L_{th}=300). There are two reasons for the large baseline RTF: one is that the non-voice segments are always decoded, and the other is that the decoding time is longer because the L_{th}=300 history is always maintained even for short utterances. Additionally, CER was worse than the proposed method because the context was also disconnected by resetting the encoder and decoder history in the middle of the utterance.

5. Conclusions

This work presents a model that integrates VAD branch into a block processing-based streaming ASR system and a re-blocking technique to avoid inappropriate isolation of the speech context. Experiments showed that the proposed method reduced the ER by 25.8% with AMI dataset with a small increase in the number of parameters compared to a conventional CTC-based integrated VAD method. Furthermore, the proposed method showed 7.5% relative improvement in CER with CSJ dataset with 27.3% reduction in RTF compared to the baseline.

6. References

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

[2] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Proc. NIPS*, vol. 28, pp. 577–585, 2015.

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

[7] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

[8] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances," *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, 1975.

[9] D. Haws, D. Dimitriadis, G. Saon, S. Thomas, and M. Picheny, "On the importance of event detection for ASR," in *Proc. ICASSP*, 2016, pp. 5705–5709.

[10] A. Lee, K. Nakamura, R. Nisimura, H. Saruwatari, and K. Shikano, "Noice robust real world spoken dialogue system using GMM based rejection of unintended inputs," in *Proc. ICSLP*, 2004.

[11] X.-L. Zhang and J. Wu, "Deep belief networks based voice activity detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 697–710, 2012.

[12] N. Ryant, M. Liberman, and J. Yuan, "Speech activity detection on YouTube using deep neural networks," in *Proc. Interspeech*, 2013, pp. 728–731.

[13] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *Proc. ICASSP*, 2013, pp. 7378–7382.

[14] G. Gelly and J.-L. Gauvain, "Optimization of RNN-based speech activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, 2017.

[15] S. Thomas, S. Ganapathy, G. Saon, and H. Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions," in *Proc. ICASSP*, 2014, pp. 2519–2523.

[16] W. Wang, X. Qin, and M. Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for M2MeT challenge," *arXiv preprint arXiv:2202.02687*, 2022.

[17] M. Gaido, M. Negri, M. Cettolo, and M. Turchi, "Beyond voice activity detection: Hybrid audio segmentation for direct speech translation," *arXiv preprint arXiv:2104.11710*, 2021.

[18] S.-Y. Chang, B. Li, and G. Simko, "A unified endpointer using multitask and multidomain training," in *Proc. ASRU*, 2019, pp. 100–106.

[19] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *Proc. ICASSP*. IEEE, 2020, pp. 6069–6073.

[20] T. Yoshimura, T. Hayashi, K. Takeda, and S. Watanabe, "End-to-end automatic speech recognition integrated with ctc-based voice activity detection," in *Proc. ICASSP*, 2020, pp. 6999–7003.

[21] Y. Fujita, T. Wang, S. Watanabe, and M. Omachi, "Toward streaming ASR with non-autoregressive insertion-based model," in *Proc. Interspeech*, 2021, pp. 3740–3744.

[22] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[23] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming asr," in *Proc. ICASSP*, 2021.

[24] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for ASR," in *Proc. ICASSP*, 2018, pp. 5874–5878.

[25] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. ICASSP*. IEEE, 2021.

[26] L. Lu, C. Liu, J. Li, and Y. Gong, "Exploring transformers for large-scale speech recognition," *Proc. Interspeech*, pp. 5041–5045, 2020.

[27] Y. Shi, V. Nagaraja, C. Wu, J. Mahadeokar, D. Le, R. Prabhavalkar, A. Xiao, C.-F. Yeh, J. Chan, C. Fuegen *et al.*, "Dynamic encoder transducer: A flexible solution for trading off accuracy for latency," *arXiv preprint arXiv:2104.02176*, 2021.

[28] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech*, pp. 4390–4394, 2019.

[29] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, 2020, pp. 6074–6078.

[30] M. Li, C. Zorilă, and R. Doddipatla, "Head-synchronous decoding for transformer-based streaming asr," in *Proc. ICASSP*, 2021.

[31] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer ASR with contextual block processing," in *Proc. ASRU*, 2019, pp. 427–433.

[32] E. Tsunoo, Y. Kashiwagi, and S. Watanabe, "Streaming transformer ASR with blockwise synchronous beam search," in *Proc. SLT*, 2021, pp. 22–29.

[33] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on Transformer vs RNN in speech applications," in *Proc. ASRU*, 2019, pp. 449–456.

[34] 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.

[35] J. Carletta, "Unleashing the killer corpus: experiences in creating the multi-everything AMI meeting corpus," *Language Resources and Evaluation*, vol. 41, no. 2, pp. 181–190, 2007.

[36] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[37] A. Rousseau, P. Deléglise, Y. Esteve *et al.*, "Enhancing the TED-LIUM corpus with selected data for language modeling and more ted talks," in *Proc. LREC*, 2014, pp. 3935–3939.

[38] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta 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.