INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Streaming End-to-End ASR based on Blockwise Non-Autoregressive Models

Tianzi Wang[1], *Yuya Fujita*[2], *Xuankai Chang*[1,3], *Shinji Watanabe*[1,3]

[1]Johns Hopkins University, USA
[2]Yahoo Japan Corporation, Japan
[3]Carnegie Mellon University, USA

wtianzi1@jhu.edu, yuyfujit@yahoo-corp.jp, {xuankaic,swatanab}@andrew.cmu.edu

Abstract

Non-autoregressive (NAR) modeling has gained more and more attention in speech processing. With recent state-of-the-art attention-based automatic speech recognition (ASR) structure, NAR can realize promising real-time factor (RTF) improvement with only small degradation of accuracy compared to the autoregressive (AR) models. However, the recognition inference needs to wait for the completion of a full speech utterance, which limits their applications on low latency scenarios. To address this issue, we propose a novel end-to-end streaming NAR speech recognition system by combining blockwise-attention and connectionist temporal classification with mask-predict (Mask-CTC) NAR. During inference, the input audio is separated into small blocks and then processed in a blockwise streaming way. To address the insertion and deletion error at the edge of the output of each block, we apply an overlapping decoding strategy with a dynamic mapping trick that can produce more coherent sentences. Experimental results show that the proposed method improves online ASR recognition in low latency conditions compared to vanilla Mask-CTC. Moreover, it can achieve a much faster inference speed compared to the AR attention-based models. All of our codes will be publicly available at https://github.com/espnet/espnet.

Index Terms: Non-autoregressive speech recognition, streaming ASR, blockwise-attention, Mask-CTC

1. Introduction

Over the past years, the advances in deep learning have dramatically boosted the performance of end-to-end (E2E) automatic speech recognition (ASR) [1–3]. Most of these E2E-ASR studies were based on autoregressive (AR) models and they achieved state-of-the-art performance [4]. However, there is a disadvantage of the AR models in that the inference time linearly increases with the output length. Recently, non-autoregressive (NAR) models has gained more and more attention in sequence-to-sequence tasks, including machine translation [5–7], speech recognition (ASR) [1, 8–11], and speech translation [12]. In contrast to the AR modeling, NAR modeling can predict the output tokens concurrently, the inference speed of which is dramatically faster than AR modeling. Especially, connectionist temporal classification (CTC) is a popular and simple NAR modeling [1,6]. However, CTC makes a strong conditional independent assumptions between the predicted tokens, leading to an inferior performance compared to the AR attention-based models [13].

To overcome this issue, several NAR studies have been proposed in the ASR field. A-FMLM [14] is designed to predict the masked tokens conditioning on the unmasked ones and the input speech. However, it needs to first predict the output length, which is difficult and easily leads to a long output sequence. Imputer [8] directly used the length of input feature sequence to address the issue and achieves comparable performance with AR models, but the computational cost can be very large. ST-NAR [15] used CTC to predict the target length and to guide the decoding. It is fast but suffers from large accuracy degradation compared with AR models. Different from previous methods, Mask-CTC [16] first generates the output tokens with greedy decode of a CTC, and then refines the tokens which have low confidence by a mask-predict decoder [7]. Mask-CTC usually predicts sequences with reasonable length and can achieve a fast inference speed, 7x faster than the AR attention-based model.

In addition to fast inference, latency is an important factor to be considered for the ASR system used in a real-time speech interface. There have been a lot of prior studies for low-latency E2E ASR based on Recurrent Neural Networks Transducer (RNN-T) [2, 13, 17, 18] and online attention-based encoder decoder (AED) with AR models, such as monotonic chunkwise attention (MoChA) [19,20], triggered attention [21], and blockwise-attention [22, 23].

Motivated by such online AR AED studies and emergent NAR research trends, we propose a novel end-to-end streaming NAR speech recognition system, by combining the blockwise-attention and Mask-CTC models. During inference, the input audio is first separated into small blocks with 50% overlap between consecutive blocks. CTC firstly predicts the preliminary tokens per block with an efficient greedy forward pass based on the output of a blockwise-attention encoder. To address the insertion and deletion error of CTC outputs frequently appeared at the boundary of each block, we apply a dynamic overlapping strategy [24] to produce coherent sentences. Then, low-confidence tokens are masked and re-predicted by a mask-predict NAR decoder [25] conditioning on the rest tokens. The greedy CTC, dynamic overlapping decoding, and mask-prediction all perform very fast, thus can achieve quite low RTF. We evaluated our approach on TEDLIUM2 [26] and AISHELL1 [27]. Compared to vanilla full-attention Mask-CTC, our proposed method decreases the online ASR recognition error rate in a low latency condition, with very fast inference speed. To the best of our knowledge, this is the first work that extending the NAR mechanism into streaming ASR.

1.1. Relationship with other streaming ASR studies

The most important success of streaming ASR recently is RNN-T and its variants. RNN-T based systems achieve state-of-the-art ASR performance for streaming applications and are successfully deployed in production systems [17, 18, 28, 29]. However, the recurrent mechanism predicts the token of the current input frame based on all previous tokens using recurrent layers, to which NAR cannot be easily applied. Besides, several ideas in this paper are inspired from online AR AED architec-

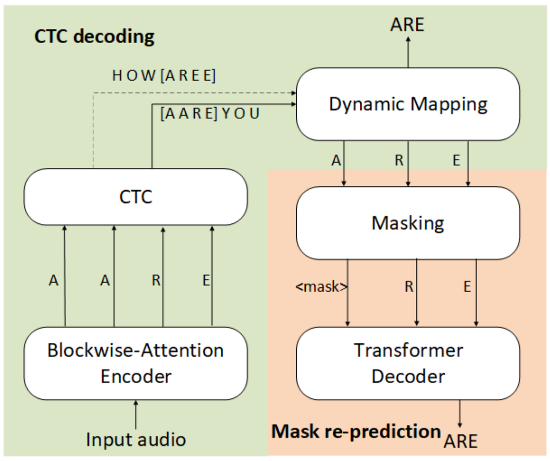

Figure 1: *Architecture of proposed streaming NAR*

tures [22–24], since they have more technical connections with NAR models based on the similar encoder-decoder framework.

2. Mask-CTC

Mask-CTC is a non-autoregressive model trained with both CTC objective and mask-prediction objective [16], where the mask-predict decoder predicts the masked tokens based on CTC output tokens and encoder output. CTC predicts a frame-level input-output alignment based on conditional independence assumption between frames. It models the probability $P(Y|X)$ of output sequence by summing up all possible alignments, where Y denotes the sequence of output and X denotes the input audio. However, due to the conditional independence assumption, CTC loses the ability of modeling correlations between output tokens and consequently loses performance.

Mask-CTC was designed to mitigate this issue by adopting an attention-based decoder as a masked language model (MLM) [7, 30], and iterative refining the output of CTC greedy decoding. During training, the tokens in the ground-truth are randomly selected and replaced by a special $\langle$mask$\rangle$ token. Then the decoder is trained to predict the actual tokens at the masked positions, Y_{mask}, conditioning on the rest unmasked tokens, Y_{obs}, and attention-based encoder output. $\mathbf{H}_{\text{enc}}$.

$$\mathbf{H}_{\text{enc}} = \text{MHSAEncoder}(X), \tag{1}$$

$$P_{\text{MLM}}(Y_{\text{mask}}|Y_{\text{obs}}, \mathbf{H}_{\text{enc}}) = \prod_{y_{\text{mask}} \in Y_{\text{mask}}} P(y_{\text{mask}}|Y_{\text{obs}}, \mathbf{H}_{\text{enc}}). \tag{2}$$

where the MHSAEncoder$(\cdot)$ denotes a multi-headed self-attention based encoder. The Mask-CTC model is optimized by a weighted sum of the CTC and MLM objectives:

$$\mathcal{L} = \gamma \log P_{\text{ctc}}(Y|\mathbf{X}) + (1-\gamma) \log P_{\text{MLM}}(Y_{\text{mask}}|Y_{\text{obs}}, \mathbf{X}), \tag{3}$$

where γ is a tunable hyper-parameter.

3. Streaming NAR

The overall architecture of the proposed E2E streaming NAR speech recognition system is shown in Figure 1. The main difference compared with Mask-CTC is that the normal MHSA-based encoder (e.g. Transformer/Conformer) as shown in Eq. (1) is replaced by a blockwise-attention encoder to make the model streamable.

3.1. blockwise-attention Encoder

To build a streaming AED-based ASR system, the encoder is only allowed to access limited future context. We use a blockwise-attention (BA) based encoder [22, 23] instead of normal multi-headed self-attention (MHSA). In a BA based encoder, the input sequence X is divided into fix-length blocks $X = [X_b]_{b=0}^{B}$, where $X_b = [x_{bl}, \dots, x_{(b+1)l-1}]$. Here b is the block index, B is the index of the last block in the whole input and l is the block length. In the computation of blockwise-attention, each block only attends to the former layer's output within the current block and the previous block. Blockwise-attention at the b-th block is defined as:

$$Z_b^i = \text{BA}(Z_b^{i-1}) = \text{MHSA}(Z_b^{i-1}, [Z_{b-1}^{i-1}, Z_b^{i-1}], [Z_{b-1}^{i-1}, Z_b^{i-1}]), \tag{4}$$

where Z_b^i is the output of encoder layer i at b-th block, $Z^0 = X$. The three arguments of MHSA$(\cdot)$ are query, key, and value matrix variables, respectively. Likewise, the blockwise-depthwise-convolution (BDC) in Conformer encoder is defined as:

$$\text{BDC}(Z_b^{i-1}) = \text{CONV}(\text{BPAD}(Z_b^{i-1}, [Z_{b-1}^{i-1}, Z_b^{i-1}])), \tag{5}$$

where CONV$(\cdot)$ is 1D depth-wise convolution and BPAD$(\cdot)$ refers to blockwise-padding, that pads zeros at right edges and pads Z_{b-1}^i at left edges of Z_b^{i-1} to keep the input/output dimension identical. The rest operations, such as point-wise convolution and activation function, is the same as in Conformer [31].

3.2. Blockwise Mask-CTC

As shown in Figure 1, the proposed E2E streaming NAR speech recognition system consists of a blockwise-attention based encoder, a CTC, a dynamic mapping process, and an MLM decoder. During training, we use full audio as the input for convenience. But the CTC output within a block only depends on the input in the current and previous block, since the computation of the encoder is blockwise. In this paper, following the NAR manner, we applied greedy decoding for CTC, which selects the token with the highest probability at each time step. The output of each block from greedy decoding CTC is:

$$\pi_b = \text{Concat}_{y_t}[\arg\max_{y_{t_{i,b}}} P(y_{t_{i,b}}|\text{BAEncoder}(X_b))] \tag{6}$$

where $b \in [0, .., B]$, $t_{i,b}$ denotes the i-th time step belonging to the b-th block, $t_{i,b} \in [t_{0,b}, \dots, t_{l-1,b}]$. BAEncoder refers to the blockwise-attention encoder as mentioned in Sec 3.1, X_{-1} is set to be a zero-matrix with the same size as X_b during computation. At the same time, the ground-truth target tokens are randomly masked and re-predicted by the MLM decoder.

During inference, the input is online segmented into fixed-length blocks with 50% overlap and fed into the encoder in a streaming way. The encoder forward pass and the CTC decoding follow the same way as in the training. As shown in Figure 1, the dotted line denotes CTC output y_{b-1} corresponds to input block X_{b-1}, and solid line denotes CTC output y_b of X_b. A dynamic mapping trick is applied on the overlap tokens between y_{b-1} and y_b to make a coherent output. More details will be shown in Sec. 3.3. Then the tokens with low-confidence scores from CTC decoding outputs, $\pi_{0..B}$, are masked and re-predicted by MLM decoder. The predicting process is done in several iterations. In each iteration, tokens with higher predicting probability are filled into masked positions and the re-filled sequences are used as input for the next iteration.

$$\hat{Y}_0 = \text{mask}(\pi_b) \tag{7}$$

$$y_{k,m} = \arg\max \log P_{\text{MLM}}(y_{mask}|\hat{Y}_{m-1}, X) \tag{8}$$

where $\hat{Y}_0$ denotes the masked token sequence from CTC output, m denotes the iteration number of re-prediction, $y_{k,m}$ denotes k re-predicted tokens in m_{th} iteration, $y_{k,m}$ are then infilled into the predicted sequences $\hat{Y}_{m-1}$ at corresponding masked positions to form the m_{th} predition $\hat{Y}_m$. $\hat{Y}_N$ would be the final output as N is total number of iterations.

3.3. Dynamic mapping for overlapping inference

Although splitting the input audio into small blocks with fixed-length is a straightforward approach to form streaming ASR, it will result in horrible performance degradation at the block boundaries. A segment boundary may appear in the middle of a token, leading to that one token may have repetitive recognition or non-recognition in two consecutive blocks. The VAD-based segmentation method is a way to solve the issue, but it is sensitive to the threshold and may lead to large latency.

In this paper, we applied overlapping inference with dynamic mapping tricks as in [24] to recover the erroneous output at the boundary, as shown in Algorithm 1. During inference, we use 50% overlap when segmenting the input audio, which ensures any frame of input audio is predicted twice by Encoder and CTC. By locating the token index in y_b that is closest to the center point of C_b, we dynamically search for the best alignment between ($y_{b,:idx_b}$ and $y_{b-1,idx_{b-1}:}$). Normalize refers to removing repeated and blank token from CTC output C_b. Following the scoring function: $\text{Score}(t^b_j) = -|j - (l-1)/2|$, we select one of the token in token pairs on the best alignment path as the output of the overlapped segment. Here t^b_j refers to the j-th token in b-th block, and l is block length.

Algorithm 1 CTC overlap decode and dynamic map

```
1:  $y_{out}$ = [∅];
2:  $\mathbf{X}$ = Audio blocks iterator with $50\%$ overlap
3:  for $b$ = 0 to B do
4:      $C_b$ = CTC_Predict(BAEncoder($X_b$))
5:      if $b > 0$ then
6:          $y_b$ = remove repeated tokens in $C_b$
7:          $idx_b$ = token index in $y_b$ that is closest to $C_{b,\frac{1}{2}l}$
8:          path$_b$ = Alignment($y_{b,:idx_b}$, $y_{b-1,idx_{b-1}:}$)
9:          $y_{out,b} = [\varnothing]$
10:         for ($p$, $q$) in path$_b$ do
11:             token = $p$ if Score($p$)>Score($q$) else $q$
12:             APPEND token to $y_{out,b}$
13:         end for
14:         $y_{out,b}$ = Normalize($y_{out,b}$)
15:     else
16:         $y_{out,b}$ = Normalize($C_{b,\frac{1}{2}l}$)
17:     end if
18:     APPEND $y_b$ to $y_{out}$
19: end for
```

4. Experiment

4.1. Experimental setup and Dataset

We evaluate our proposed model on both Chinese and English Speech corpora: TEDLIUM2 [26] and AISHELL1 [27]. For all experiments, the input features are 80-dimensional log-mel filter-banks with pitch computed with frame length of 25ms and frame shift of 10ms. We use Kaldi toolkit [32] for feature extraction. We also apply speed perturbation(speed rate=0.9, 1.0, 1.1) and spectrum augmentation [33] for data augmentation. Models are evaluated with both full-context decoding and streaming decoding.

All experiments are conducted using the open-source, E2E speech processing toolkit ESPnet [31, 34, 35]. The encoder first contains 2 CNN blocks to downsample the input to $1/4$, followed by 12 MHSA-based layers. In streaming NAR, BA-based layers(Eq. 4, 5) is used in encoder to replace MHSA. Decoders have a similar stacked-block structure with 6 layers and only full-attention transformer blocks. For any self-attention block in this paper, we use $h = 4$ parallel attention heads, with dimension $d^{\text{att}} = 256$. For feed-forward layer, we use dimensionality $d^{\text{ffn}} = 2048$ and apply swish as activation functions. In self-attention, relative position embedding is augmented in the input [36]. We use 15 as the kernel size for Conformer convolution. Models on TEDLIUM2 are trained for 200 epochs and on AISHELL1 are trained for 150 epochs. The evaluation is done on the averaged model over the best 10 checkpoints. We do not integrate Language Model during decoding.

4.2. Results

Table 1 shows TEDLIUM2 results, including word error rates (WERs) on dev/test sets, averaged latency and real-time factor (RTF). Latency and RTF were measured on the CPU platform (Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.3GHz) with 8 parallel jobs. We set block length $l = 16$, corresponding to 640ms before subsampling. The full-context evaluation is done on the utterance level, while the streaming evaluation is done on the unsegmented streaming audio for each speaker. Since it is hard to define the latency with long unsegmented audio, we calculated the averaged latency per utterance with dev:

$$\text{Latency} = \frac{\sum_{utt}(\text{last token emitted} - \text{end of speech})}{\#\text{total number of utterance}}. \tag{9}$$

The last token emitted time is the time stamp that model predicts last token, and the end of speech is determined by forced alignment with external model. This latency considered both look-ahead latency and computation time. For comparison, we also report the latency for full-context decoding by measuring the average decoding time per utterance, since in full-context case decoding can not start before entire audio is fed.

Compared to the vanilla Mask-CTC, the proposed streaming NAR model performs much better in streaming mode. With the Conformer encoder, the WER on the dev of Mask-CTC is 23.5 with 310ms averaged latency and that of streaming NAR is 12.1 with 140ms averaged latency. The RTF is about 2x/10x faster than the AR attention-based model with beam size=1/10 respectively. Mask-CTC models are trained with full context input and work better with full future information. We can observe a significant WER degradation in streaming decoding mode, which is due to the input mismatch during training and decoding. On the other hand, the proposed streaming NAR can recognize the current frame with only a small future context, thus fit well with the streaming inference. However, the WERs increased from full context to streaming mode in the proposed model (from 10.4/9.4 to 12.1/11.7). The reason would be the error raised at the segment boundaries. From our observation, these errors can be relieved but not totally solved by dynamic mapping and still exist even with large block lengths.

Table 1: *TEDLIUM2: WERs on dev/test, averaged Latency and RTF are reported. 640ms input segments is used for all streaming decode mode. The attention-based AR and Mask-CTC models trained with conventional attention, while the proposed streaming NAR use blockwise attention with 640ms block length.*

	Encoder type	Decode Mode	WER on dev	WER on test	Latency$_{\text{utt}}$(ms)	RTF
Attention-based AR	Transformer	full-context	11.7	9.9	5220	0.46
	+ beamsize=10	full-context	10.9	9.1	34160	3.01
	Conformer	full-context	11.0	8.4	6130	0.54
	+ beamsize=10	full-context	11.1	8.1	37780	3.33
Mask-CTC	Transformer	full-context	11.0	10.7	790	0.07
	Conformer	full-context	10.0	8.8	1070	0.09
	Transformer	streaming	18.2	16.4	300	0.20
	Conformer	streaming	23.5	21.3	310	0.26
Streaming NAR (Proposed)	Transformer	full-context	12.2	11.2	910	0.08
	Conformer	full-context	**10.4**	**9.4**	1030	0.09
	Transformer	streaming	14.2	14.0	**120**	0.22
	Conformer	streaming	**12.1**	**11.7**	**140**	0.32

Table 2: *AISHELL1: WERs, averaged latency and RTF are reported. 1280ms input segments is used for all streaming inference. The attention-based AR and Mask-CTC models work with conventional attention, while the proposed streaming NAR use blockwise attention with 1280ms block length*

	Encoder type	Decode Mode	WER on dev	WER on test	Latency$_{\text{utt}}$(ms)	RTF
Attention-based AR	Transformer	full-context	6.7	7.6	2040	0.45
	+ beamsize=10	full-context	6.6	7.4	11640	2.56
Mask-CTC	Transformer	full-context	6.9	7.8	220	0.05
	Transformer	streaming	9.0	10.4	280	0.18
Streaming NAR (Proposed)	Transformer	full-context	8.6	9.9	230	0.05
	Transformer	streaming	**8.6**	**9.9**	320	0.20

Table 3: *WERs and RTF on TEDLIUM2 conduct on proposed Streaming NAR model with different block length(BL), Experiments are all conducted under streaming mode.*

Encoder	Dev	Test	BL(ms)	Latency$_{\text{utt}}$(ms)	RTF
BA-TF	12.8	12.0	5120	1530	0.17
	12.8	11.9	2560	700	0.18
	13.0	12.2	1280	290	0.18
	14.2	14.0	640	120	0.22
BA-CF	10.5	10.3	5120	1700	0.29
	10.6	10.4	2560	790	0.29
	11.2	10.7	1280	380	0.30
	12.1	**11.7**	640	140	0.32

Table 2 shows the results on AISHELL1. Since the hyper-parameters for Conformer and Mask-CTC need careful tuning and the model is easy to be overfitted in our preliminary experiments, we only report transformer results on the AISHELL1 task. The vanilla Mask-CTC works better when full-context is provided during decoding while streaming NAR is better on streaming decode, but the benefits are smaller than those in TEDLIUM2. A possible reason is that the training utterance in AISHELL1 is much shorter than TEDLIUM2. Full-context attention can also gain the ability to recognize with only short future context.

To understand the performance of streaming NAR under different latency, in Table 3 we compare the WERs with different block lengths for blockwise-attention Transformer (BA-TF) and blockwise-attention Conformer (BA-CF) on TEDLIUM2. We observe that as the block length get shorter, the latency becomes smaller while RTF rises since the shorter blocks require more iterations to forward the whole input audio. Besides, when block length decreases from 5120ms to 1280ms, the result rarely changes. It indicates that in streaming ASR, the closer future context is a much more active player than distant ones.

5. Conclusion

In this paper, we proposed a novel E2E streaming NAR speech recognition system. Specifically, we combined the blockwise-attention based Encoder and Mask-CTC. Beside, we applied the dynamic overlapping inference to mitigate the errors at the boundary. Compared to vanilla Mask-CTC, the proposed streaming NAR model achieves competitive performance in full-context decoding and outperforms the vanilla Mask-CTC streaming decoding with very low utterance latency. Moreover, the decoding speed of the proposed model is about 2x/10x faster than the AR attention-based model with beam size=1/10 respectively. Our future plan is developing better boundaries localization method to replace the overlapping inference, and integrating external language model during decoding.

6. Acknowledgements

We would like to thank Mr. Yusuke Higuch of Waseda University for his valuable information about the Mask-CTC.

7. References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

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

[3] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[4] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[5] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," *arXiv preprint arXiv:1711.02281*, 2017.

[6] J. Libovickỳ and J. Helcl, "End-to-end non-autoregressive neural machine translation with connectionist temporal classification," *arXiv preprint arXiv:1811.04719*, 2018.

[7] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *in Proc. EMNLP-IJCNLP*, 2019, pp. 6114–6123.

[8] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1403–1413.

[9] Y. Fujita, S. Watanabe, M. Omachi, and X. Chang, "Insertion-based modeling for end-to-end automatic speech recognition," *Proc. Interspeech 2020*, pp. 3660–3664, 2020.

[10] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," *arXiv preprint arXiv:2010.14233*, 2020.

[11] R. Fan, W. Chu, P. Chang, and J. Xiao, "Cass-nat: Ctc alignment-based single step non-autoregressive transformer for speech recognition," *arXiv preprint arXiv:2010.14725*, 2020.

[12] H. Inaguma, Y. Higuchi, K. Duh, T. Kawahara, and S. Watanabe, "Orthros: Non-autoregressive end-to-end speech translation with dual-decoder," *arXiv preprint arXiv:2010.13047*, 2020.

[13] E. Battenberg, J. Chen, R. Child, A. Coates, Y. G. Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

[14] N. Chen, S. Watanabe, J. Villalba, and N. Dehak, "Listen and fill in the missing letters: Non-autoregressive transformer for speech recognition," *arXiv preprint arXiv:1911.04908*, 2019.

[15] Z. Tian, J. Yi, J. Tao, Y. Bai, S. Zhang, and Z. Wen, "Spike-triggered non-autoregressive transformer for end-to-end speech recognition," *arXiv preprint arXiv:2005.07903*, 2020.

[16] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *arXiv preprint arXiv:2005.08700*, 2020.

[17] A. Tripathi, J. Kim, Q. Zhang, H. Lu, and H. Sak, "Transformer transducer: One model unifying streaming and non-streaming speech recognition," *arXiv preprint arXiv:2010.03192*, 2020.

[18] M. Jain, K. Schubert, J. Mahadeokar, C.-F. Yeh, K. Kalgaonkar, A. Sriram, C. Fuegen, and M. L. Seltzer, "Rnn-t for latency controlled asr with improved beam search," *arXiv preprint arXiv:1911.01629*, 2019.

[19] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," *arXiv preprint arXiv:1712.05382*, 2017.

[20] H. Inaguma, M. Mimura, and T. Kawahara, "Enhancing monotonic multihead attention for streaming asr," *arXiv preprint arXiv:2005.09394*, 2020.

[21] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6074–6078.

[22] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online ctc/attention end-to-end speech recognition architecture," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6084–6088.

[23] E. Tsunoo, Y. Kashiwagi1, and S. Watanabe, "Streaming transformer asr with blockwise synchronous beam search," in *Proc. SLT*, 2021, pp. 22–29.

[24] C.-C. Chiu, W. Han, Y. Zhang, R. Pang, S. Kishchenko, P. Nguyen, A. Narayanan, H. Liao, S. Zhang, A. Kannan *et al.*, "A comparison of end-to-end models for long-form speech recognition," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 889–896.

[25] Y. Higuchi, H. Inaguma, S. Watanabe, T. Ogawa, and T. Kobayashi, "Improved mask-ctc for non-autoregressive end-to-end asr," *arXiv preprint arXiv:2010.13270*, 2020.

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

[27] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.

[28] 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 *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6069–6073.

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

[30] N. Chen, S. Watanabe, J. Villalba, and N. Dehak, "Nonautoregressive transformer automatic speech recognition," *arXiv preprint arXiv:1911.04908*, 2019.

[31] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on espnet toolkit boosted by conformer," *arXiv preprint arXiv:2010.13956*, 2020.

[32] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[33] 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," *arXiv preprint arXiv:1904.08779*, 2019.

[34] 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 *Proceedings of Interspeech*, 2018, pp. 2207–2211. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1456

[35] 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 *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 449–456.

[36] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.