TRANSFORMER-BASED STREAMING ASR WITH CUMULATIVE ATTENTION

Mohan Li, Shucong Zhang, Cătălin Zorilă and Rama Doddipatla

Cambridge Research Laboratory, Toshiba Europe Ltd, Cambridge, UK

ABSTRACT

In this paper, we propose an online attention mechanism, known as cumulative attention (CA), for streaming Transformer-based automatic speech recognition (ASR). Inspired by monotonic chunk-wise attention (MoChA) and head-synchronous decoder-end adaptive computation steps (HS-DACS) algorithms, CA triggers the ASR outputs based on the acoustic information accumulated at each encoding timestep, where the decisions are made using a trainable device, referred to as halting selector. In CA, all the attention heads of the same decoder layer are synchronised to have a unified halting position. This feature effectively alleviates the problem caused by the distinct behaviour of individual heads, which may otherwise give rise to severe latency issues as encountered by MoChA. The ASR experiments conducted on AIShell-1 and Librispeech datasets demonstrate that the proposed CA-based Transformer system can achieve on par or better performance with significant reduction in latency during inference, when compared to other streaming Transformer systems in literature.

Index Terms— End-to-end ASR, Transformer, online attention mechanism, cumulative attention

1. INTRODUCTION

In recent years, the ASR community has witnessed a surge in developing end-to-end (E2E) systems, which have exhibited less computational complexity and competitive performance when compared with the conventional hidden Markov model (HMM) based systems. For E2E ASR, the acoustic model (AM), lexicon and language model (LM) are integrated into a holistic network, and could be optimised without any prior knowledge. The E2E techniques can be generally categorised into three classes, known as connectionist temporal classification (CTC) [1, 2], recurrent neural network transducer (RNN-T) [3] and attention-based encoder-decoder framework [4, 5, 6]. As an epitome of the last class, Transformer [7] has been introduced to ASR following its overwhelming success in the field of natural language processing (NLP). Consequently, state-of-the-art performance is reported on a number of standard ASR tasks [8]. The self-attention mechanism that underpins the Transformer architecture has soon replaced the dominant role of RNN in terms of both acoustic and language modelling, leading to the advent of Speech Transformer [9], Transformer Transducer [10, 11] and Transformer-XL [12]. Unlike RNN structures that process the inputs in the temporal order, the self-attention module captures the correlation between the elements of any distance, which efficiently lifts the restriction imposed by long-term dependency.

Although Transformer has shown prominent advantage over the RNN based systems, it faces major difficulties in online ASR. Both the encoder and the decoder require access to the full speech utterance, leading to large latency and thus limiting the use for practical scenarios. In this paper, we focus on the streaming strategies in the decoder to address the latency in recognition. Previously, the following methods have been proposed and were shown to be effective for streaming Transformer architecture: (1) hard monotonic attention mechanisms, represented by the monotonic chunkwise attention (MoChA) [13, 14, 15] and its simplified variant, monotonic truncated attention (MTA) [16, 17]; (2) Motivated by the streaming characteristic of CTC technique, triggered attention (TA) [18] takes the spikes of CTC score as the boundaries for computing attention weights; (3) blockwise synchronous inference [19] performs standard attention independently on each input chunk, with the broken decoder states carried over to the succeeding chunks; (4) Continuous integrate-and-fire (CIF) [20], along with the decoder-end adaptive computation steps (DACS) [21] and its head-synchronous version (HS-DACS) [22] interpret the online decoding as an accumulation process of attending confidence, which is halted when the accumulator exceeds a predefined threshold.

Among the aforementioned streaming methods, MoChA has been widely acknowledged, where it considers the ASR decoding as a monotonic attending process. The output is emitted around its acoustic endpoint. However, the output decision in MoChA is merely dependent on a single frame, adding possibilities for accidental triggering in complex speech conditions. Apart from this, in the context of multi-head Transformer, some of the heads might not capture valid attentions, and fail to truncate the decoding before the end of speech is reached. HS-DACS circumvents the problems of MoChA by: (i) transforming the decisions based on single-frame-activation into an accumulation of attending confidence; (ii) merging the halting probabilities to produce a same halting position for all attention heads in a decoder layer. Nonetheless, it is still not guaranteed that there is enough confidence to trigger the output, and an additional stopping criterion in the form of maximum look-ahead steps was introduced.

To overcome the above issues encountered by MoChA and HS-DACS, the paper proposes the cumulative attention (CA) algorithm. The CA-based attention heads collectively accumulate the acoustic information along each encoding timestep, and the ASR output is triggered once the accumulated information is deemed sufficient by a trainable device. The proposed algorithm not only involves multiple frames into the acoustic-aware decision making, but also achieves a unified halting positions for all the heads in the decoder layer at each decoding step.

The rest of the paper is organised as follows: Section 2 presents the architecture of Transformer ASR and some online attention approaches. Section 3 elaborates the workflow of the proposed CA algorithm. Experimental results are demonstrated in Section 4. Finally, conclusions are drawn in Section 5.

2. STREAMING TRANSFORMER ASR SYSTEM

A typical Transformer-based ASR system consists of three components: front-end, encoder and decoder. The front-end is commonly

implemented as convolutional neural networks (CNNs), which aims to enhance the acoustic feature extraction and conduct frame-rate reduction. The output of the CNN front-end enters into a stack of encoder layers, with each comprising of two sub-layers as a multi-head self-attention module and a pointwise feed-forward network (FFN). The decoder also stacks several layers and has a similar architecture as that of the encoder, except for an additional cross-attention sub-layer that interacts with the encoder states.

The attention modules in the Transformer architecture adopts the dot-product attention mechanism to model inter-speech and speech-to-text dependencies:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\text{T}}}{\sqrt{d_k}})\mathbf{V}, \qquad (1)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T/L \times d_k}$ denote the encoder/decoder states in the self-attention sub-layers, or $\mathbf{Q} \in \mathbb{R}^{L \times d_k}$ denotes the decoder states, and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d_k}$ denote the encoder states in the cross-attention sub-layers, given d_k as the attention dimension and T, L as the length of the encoder and decoder states, respectively.

The power of the Transformer architecture also arises from the utilisation of multi-head attention at each sub-layer, where the heads project raw signals into various feature spaces, so as to capture the sequence dependencies from various aspects:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, ..., \text{head}_H)\mathbf{W}^O, \quad (2)$$

$$\text{where head}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^Q, \mathbf{K}\mathbf{W}_h^K, \mathbf{V}\mathbf{W}_h^V), \quad (3)$$

where $\mathbf{W}_h^{Q,K,V} \in \mathbb{R}^{d_k \times d_k}$ and $\mathbf{W}^O \in \mathbb{R}^{d_m \times d_m}$ represent the projection matrices, and H is the number of attention heads given $d_m = H \times d_k$.

With regard to streaming ASR systems, the main challenge for Transformer is that it requires the full speech to perform the attention computations. At the encoder side, a relatively straightforward approach to overcome this issue is to splice the input features into overlapping chunks, and feed them to the system sequentially [17]. However, such a strategy cannot be applied to the decoder side, because the attention boundaries of the ASR outputs might not be restricted within a single chunk. As a result, several online attention mechanisms have been proposed. Monotonic chunkwise attention (MoChA) was one of the first methods to address the problem. During decoding, the attending decision is made monotonically for each encoder state. Once a certain timestep is attended, a second-pass soft attention is conducted within a small chunk to serve as the layer outcome. Head-synchronous decoder-end adaptive computation steps (HS-DACS) was later proposed to take advantage of the history frames and better handle the distinct performance of the attention heads. The halting probabilities were produced and accumulated along the encoder states across all heads, and the output is triggered when the accumulation reaches a threshold. In the next section, we'll present the cumulative attention (CA) algorithm, which incorporates both the acoustic-aware decision of MoChA, and the head-synchronisation ability of HS-DACS.

3. PROPOSED CUMULATIVE ATTENTION ALGORITHM

According to the investigation presented in [15], the lower Transformer decoder layers tend to capture noisy and invalid attentions, which does harm to the general decoding especially for streaming scenarios. Similarly, we adopt the layer-drop strategy by applying the proposed CA algorithm just to the cross-attention module of the top decoder layer. Meanwhile, the rest of the layers are only

---

Algorithm 1. CA Inference for Transformer ASR

Input: encoder states $\boldsymbol{k}(\boldsymbol{v})$, decoder states $\boldsymbol{q}$, input length T, number of heads H.

1 **Initialization:** $i = 1$, $y_0 = \langle sos \rangle$, $t_0 = 0$
2 **while** $y_{i-1} \neq \langle eos \rangle$ **do**
3    $\boldsymbol{c}_{i,0}^h = \mathbf{0}$
4   **for** $j = 1$ **to** T **do**
5   **for** $h = 1$ **to** H **do**
6        $a_{i,j}^h = \text{Sigmoid}(\frac{\boldsymbol{q}_{i-1}^h(\boldsymbol{k}_j^h)^{\text{T}}}{\sqrt{d_k}})$
7        $\boldsymbol{c}_{i,j}^h = \boldsymbol{c}_{i,j-1}^h + a_{i,j}^h \cdot \boldsymbol{v}_j^h$
8   **end**
9      $\boldsymbol{c}_{i,j} = \text{Concat}(\boldsymbol{c}_{i,j}^1, ..., \boldsymbol{c}_{i,j}^H)$
10      $p_{i,j} = \text{HaltSelect}(\boldsymbol{c}_{i,j})$
11   **if** $p_{i,j} \geqslant 0.5$ **then**
12   break
13   **end**
14   **end**
15    $\boldsymbol{c}_i := \boldsymbol{c}_{i,j}$
16    $t_i = \max(t_{i-1}, j)$
17    $y_i = \text{OutputLayer}(\boldsymbol{c}_i)$
18    $i \mathrel{+}= 1$
19 **end**

---

equipped with the self-attention module and perform language modelling. The workflow of the algorithm for a certain decoding step i is described as follows.

Like all other online attention mechanisms, at head h, for each encoding timestep j, an attention energy $e_{i,j}^h$ is computed given the last decoder state $\boldsymbol{q}_{i-1}^h$ and the encoder state $\boldsymbol{k}_j^h$:

$$e_{i,j}^h = \frac{\boldsymbol{q}_{i-1}^h(\boldsymbol{k}_j^h)^T}{\sqrt{d_k}}. \qquad (4)$$

The energy is immediately fed to a sigmoid unit to produce a monotonic attention weight:

$$a_{i,j}^h = \text{Sigmoid}(e_{i,j}^h). \qquad (5)$$

The sigmoid unit is regarded as an effective alternative to the softmax function in the streaming case, which scales the energy to the range $(0, 1)$ without accessing the entire input sequence. As opposed to MoChA and HS-DACS where the outcome of eq. (5) is directly interpreted as the attending/halting probability that dictates the output triggering, CA interprets it as the relevance of the encoder state to the current decoding step. This is the same as the standard attention mechanism in the offline system or the second-pass attention performed in MoChA.

Next, an interim context vector is generated at j in an autoregressive manner:

$$\boldsymbol{c}_{i,j}^h = \boldsymbol{c}_{i,j-1}^h + a_{i,j}^h \cdot \boldsymbol{v}_{i,j}^h, \qquad (6)$$

which carries all the processed acoustic information accumulated at the current timestep (the term $\boldsymbol{c}_{i,j-1}^h$ is discarded when $j = 1$). Though HS-DACS performs accumulation at each decoding step in a similar way, it is with respect to the halting probabilities instead of acoustic information.

Inspired by HS-DACS, in order to force all the attention heads to halt at the same position, the interim context vectors produced by

different heads are concatenated into a comprehensive one:

$$\boldsymbol{c}_{i,j} = \text{Concat}(\boldsymbol{c}_{i,j}^1, ..., \boldsymbol{c}_{i,j}^H). \tag{7}$$

We introduce a trainable device, referred to as *Halting Selector*, to determine whether to trigger the ASR output at each timestep, which is implemented as a single/multi-layer deep neural network (DNN) with the output dimension of one in our system. Then $\boldsymbol{c}_{i,j}$ is input to the DNN to calculate a halting probability:

$$p_{i,j} = \text{Sigmoid}(\text{HaltSelect}(\boldsymbol{c}_{i,j}) + r + \epsilon), \tag{8}$$

which represents the likelihood of halting the ith decoding step at timestep j, provided the acoustic features accumulated so far by all the attention heads. The parameters r denotes a bias term that is initialised to -4, and ϵ is an additive Gaussian noise applied only to training in order to encourage the discreteness of $p_{i,j}$.

Here, one may notice the major difference between CA and other streaming methods. In CA algorithm, the interim context vectors are computed first, based on which the ASR outputs will be activated. Whereas for MoChA and HS-DACS, the attending decision is taken first and then the final context vector is produced. If the decision is made inappropriately, then the corresponding context vector can contain invalid acoustic information for the decoder.

Since the halting selector assigns hard decisions, making the system parameters non-differentiable, the training requires to take the expected value of c_i by marginalising all possible halting positions. Hence, a distribution of the halting hypotheses is given as:

$$\alpha_{i,j} = p_{i,j} \prod_{k=1}^{j-1}(1 - p_{i,k}), \tag{9}$$

which is a simplified version of the counterpart calculated in MoChA [13]. Finally, the expected context vector for the decoding step i is computed as:

$$\boldsymbol{c}_i = \sum_{j=1}^{T} \alpha_{i,j} \cdot \boldsymbol{c}_{i,j} \tag{10}$$

It is important to note that HS-DACS doesn't calculate such an expectation in training, as the accumulation is truncated by a preset threshold and only the one-best context vector is derived.

During CA inference, $p_{i,j}$ is monotonically computed at each timestep from $j = 1$, and the decoding step i would be halted at the earliest j where $p_{i,j} \geqslant 0.5$. The corresponding interim context vector $\boldsymbol{c}_{i,j}$ is selected and sent to the output layer to predict the ASR output. The pseudocodes of the above inference process are presented in algorithm 1.

One should be aware that although the CA algorithm adopts the Bernoulli sample process at single timesteps similar to what MoChA does, the halting decision is based on the whole encoding history. Unlike MoChA where individual heads detect separate halting positions, in CA all the attention heads simultaneously contribute to $\boldsymbol{c}_{i,j}$ and have a unified halting position. This makes it less vulnerable to the disturbance in single frames. Also, CA rules out the use of arbitrary accumulation threshold as does in HS-DACS, allowing the decoding to be halted flexibly in the weak attention scenarios.

4. EXPERIMENTS

4.1. Experimental setup

The proposed CA algorithm has been evaluated on the AIShell-1 Chinese task and the Librispeech English task. Following the standard recipes in ESPNet toolkit [23], speech perturbation is applied

Table 1. Character error rates (CERs %) on AIShell-1.

Model	dev	test
Offline		
Transformer [8]	-	6.7
Online		
MMA-MoChA Transformer [15]	-	7.5
BS-DEC Transformer [19]	6.4	7.3
MoChA Transformer	6.4	7.2
HS-DACS Transformer	6.3	7.0
CA Transformer	**6.3**	**7.0**

Table 2. Word error rates (WERs %) on Librispeech.

Model	dev		test	
	clean	other	clean	other
Offline				
Transformer (ours)	2.4	6.0	2.6	6.1
Online				
CIF [20]	-	-	3.3	9.6
Triggered attention [18]	-	-	2.8	7.2
BS-DEC Transformer [19]	2.5	6.8	2.7	7.1
MoChA Transformer	2.8	7.1	3.1	7.4
HS-DACS Transformer	2.4	6.6	2.6	6.6
CA Transformer	**2.5**	**6.7**	**2.7**	**6.8**

to AIShell-1, and SpecAugment [24] is conducted on Librispeech. The acoustic features are 80-dimensional filterbank coefficients along with 3-dimensional pitch information. The vocabulary of the above datasets is 4231 Chinese characters and 5000 BPE tokenised word-pieces [25], respectively.

Both tasks adopt a similar Transformer architecture. The front-end consists of 2 CNN layers, with each having 256 kernels with the width of 3×3 and a stride of 2×2 that reduces the frame rate by 2 folds. In order to deal with the online input, the 12-layer encoder takes the chunkwise streaming strategy as in [17] where the sizes of the left, central and right chunks are $\{64, 64, 32\}$ frames. At each encoder layer, the number of heads, attention dimension and the unit size of FFN are $\{4, 256, 2048\}$ for AIShell-1, and $\{8, 512, 2048\}$ for Librispeech. The decoder of the system stacks 6 layers with the same parameters as the encoder for each task, and there's only 1 DNN layer in the halting selector.

During training, the joint CTC/attention loss is utilised for multi-objective learning with the weight of $\lambda_{CTC} = 0.3$. The learning rate (lr) of both tasks follows the Noam weight decay scheme [7] with the initial lr, warmup step and number of epochs set to $\{1.0, 25000, 50\}$ for AIShell-1 and $\{5.0, 25000, 120\}$ for Librispeech. As for inference is concerned, CTC joint decoding is carried out with $\lambda_{CTC} = 0.3$ for AIShell-1 and $\lambda_{CTC} = 0.4$ for Librispeech. An external LM trained with the texts of the training set is incorporated to rescore the beam search (beam width=10) hypotheses decoded by the system, where the LM is a 650-unit 2-layer Long Short Term Memory (LSTM) network and a 2048-unit 4-layer LSTM for AIShell-1 and Librispeech, respectively.

4.2. Experimental results

Table 1 and 2 demonstrate the ASR performance of the proposed system on AIShell-1 and Librispeech datasets in terms of character-

Table 3. Latency (frames) on AIShell-1 and Librispeech.

Model	AIShell-1		Librispeech			
	dev	test	dev		test	
			clean	other	clean	other
offline	232.8	257.0	497.9	440.7	528.5	463.8
MoChA	90.1	92.5	295.2	256.4	303.1	271.2
HS-DACS	53.4	54.2	163.8	145.0	156.1	146.5
CA	**52.8**	**51.8**	**68.1**	**63.5**	**68.5**	**65.5**

error-rate (CER) and word-error-rate (WER) respectively. The reference systems are chosen to have similar Transformer architecture, input chunk sizes and external LM. Besides, for a fair comparison with CA, both the MoChA and HS-DACS based systems are trained with only one cross-attention layer ($D = 1$), except for the HS-DACS on Librispeech which has three ($D = 3$ as models with $D = 1$ or 2 failed to converge well).

We observe that on both tasks, the CA system achieves better accuracy than the other systems in literature. With regard to the reproduced MoChA and HS-DACS models, on AIShell-1, CA obtains a relative gain of 2.8% when compared with MoChA, and similar performance to HS-DACS. As for Librispeech is concerned, CA outperforms MoChA in both clean and noisy conditions with the relative gains of 16.1% and 10.8%, respectively. Moreover, CA still achieves comparable WERs to HS-DACS, given fewer cross-attention layers were used in the CA system.

4.3. Latency measurement

The latency of inference has also been measured for the proposed algorithm together with MoChA and HS-DACS. Here we adopt the corpus-level latency as defined in [26], which is computed as the difference between the boundary of the right input chunk $\hat{b}_i^k$ where the halting position is located, and the actual boundary of the output token b_i^k obtained from HMM forced alignment:

$$\triangle_{corpus} = \frac{1}{\sum_{k=1}^{N} |\boldsymbol{y}^k|} \sum_{k=1}^{N} \sum_{i=1}^{|\boldsymbol{y}^k|} (\hat{b}_i^k - b_i^k), \qquad (11)$$

where N denotes the total number of utterances in the dataset, and $|\boldsymbol{y}^k|$ is the number of output tokens in each utterance. Since there might be ASR errors in the hypothesis sequence that will result in faulty latency computation, we only include $\hat{b}_i^k$ of the correctly decoded tokens in the above equation. Though this might lead to different denominators in eq. (11), the comparison of latency is still reasonable given the similar ASR accuracy achieved by three systems. Also, as all the online attention mechanisms in our experiments are independently performed at each decoding step, it is not guaranteed that the halting positions are monotonic. Thus, when computing the latency, $\hat{b}_i^k$ is always synchronised to the furthest timestep ever seen in the decoding process (see Algorithm. 1 line 16).

Table 3 presents the latency levels of MoChA, HS-DACS and CA based systems evaluated on AIShell-1 and Librispeech datasets. In order to have a fair comparison to CA, in both MoChA and HS-DACS, the maximum look-ahead steps (M) is not applied during the decoding process. On AIShell-1, we observed that the latency levels of both systems seem to be reasonable when compared to the offline system, while on Librispeech, it was noticed that the latency levels were close to the offline system, due to the redundant heads that cannot capture valid attentions. In order to reduce the latency, a maximum look-ahead step of $M = 16$ is imposed during recognition

(a) MoChA

(b) HS-DACS   (c) CA
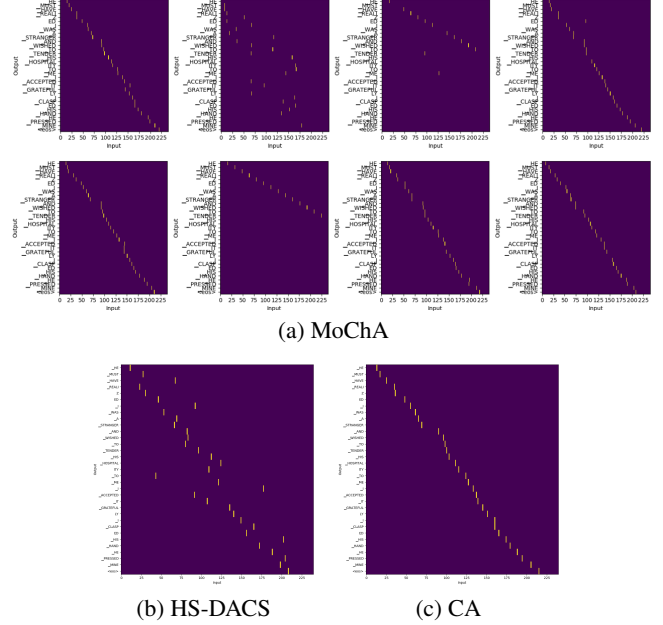
Fig. 1. Separate/joint halting decisions of the ASR outputs given by online attention heads in streaming Transformer systems.

only for the Librispeech task. One can observe that CA (without M) achieves better latency levels than MoChA and HS-DACS on both AIShell-1 (without M) and on Librispeech (with M).

The poor latency performance given by MoChA and HS-DACS might be explained by looking at the halting decisions of various heads to generate the ASR outputs, as shown in Fig. 1. One can observe from Fig. 1 (a), heads 2, 3 and 6 in MoChA are mostly unable to halt the decoding and have to rely on the truncation executed by the maximum look-ahead steps. Similarly, in Fig. 1 (b) the accumulation of halting probabilities in HS-DACS heads fails to exceed the joint-threshold (number of heads, 8) at certain decoding steps, making the inference process reach the end of speech at early stages. On the other hand, the halting decisions of CA are made monotonically and always in time as illustrated in Fig. 1 (c). Although CA might also have redundant heads, these heads can be backed up by the other functioning ones, since all of them are synchronised and the halting decision is based on the overall acoustic information.

5. CONCLUSION

The paper presented a novel online attention mechanism, known as cumulative attention (CA) for streaming Transformer ASR. Combining the advantages of the acoustic-aware method (MoChA) and the accumulation based approach (HS-DACS), the CA algorithm utilised an trainable device called halting selector to determine robust halting positions to trigger the ASR output. The attention heads in the CA layer were synchronised to produce a unified halting positing in the decoder layer. By doing so, all the heads simultaneously contribute to the potential ASR output, so that the issues caused by the distinct behaviours of the heads were effectively eased off. Experiments on AIShell-1 and Librispeech showed that the proposed CA approach achieved similar or better ASR performance compared to existing algorithms in literature with significantly noticeable gains in latency during inference.

6. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen 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] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International conference on machine learning*. PMLR, 2014, pp. 1764–1772.

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

[4] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," *arXiv preprint arXiv:1506.07503*, 2015.

[5] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 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 *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[8] 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 *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 449–456.

[9] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[10] Zhengkun Tian, Jiangyan Yi, Jianhua Tao, Ye Bai, and Zhengqi Wen, "Self-attention transducers for end-to-end speech recognition," *arXiv preprint arXiv:1909.13037*, 2019.

[11] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

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

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

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

[15] Hirofumi Inaguma, Masato Mimura, and Tatsuya Kawahara, "Enhancing monotonic multihead attention for streaming asr," *arXiv preprint arXiv:2005.09394*, 2020.

[16] Haoran Miao, Gaofeng Cheng, Pengyuan Zhang, and Yonghong Yan, "Online hybrid ctc/attention end-to-end automatic speech recognition architecture," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1452–1465, 2020.

[17] Haoran Miao, Gaofeng Cheng, Changfeng Gao, Pengyuan Zhang, and Yonghong 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.

[18] Niko Moritz, Takaaki Hori, and Jonathan 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.

[19] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe, "Streaming transformer asr with blockwise synchronous beam search," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 22–29.

[20] Linhao Dong and Bo Xu, "Cif: Continuous integrate-and-fire for end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6079–6083.

[21] Mohan Li, Cătălin Zorilă, and Rama Doddipatla, "Transformer-based online speech recognition with decoder-end adaptive computation steps," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 1–7.

[22] Mohan Li, Cătălin Zorilă, and Rama Doddipatla, "Head-synchronous decoding for transformer-based streaming asr," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5909–5913.

[23] 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," *arXiv preprint arXiv:1804.00015*, 2018.

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

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[26] Hirofumi Inaguma, Yashesh Gaur, Liang Lu, Jinyu Li, and Yifan Gong, "Minimum latency training strategies for streaming sequence-to-sequence asr," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6064–6068.