INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

End-to-End ASR with Adaptive Span Self-Attention

Xuankai Chang[1], *Aswin Shanmugam Subramanian*[1], *Pengcheng Guo*[1,2]
Shinji Watanabe[1], *Yuya Fujita*[3], *Motoi Omachi*[3]

[1]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA
[2]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[3]Yahoo Japan Corporation, Tokyo, Japan

{xchang14,aswin,shinjiw}@jhu.edu, pcguo@nwpu-aslp.org, {yuyfujit,momachi}@yahoo-corp.jp

Abstract

Transformers have demonstrated state-of-the-art performance on many tasks in natural language processing and speech processing. One of the key components in Transformers is self-attention, which attends to the whole input sequence at every layer. However, the computational and memory cost of self-attention is square of the input sequence length, which is a major concern in automatic speech recognition (ASR) where the input sequence can be very long. In this paper, we propose to use a technique called adaptive span self-attention for ASR tasks, which is originally proposed for language modeling. Our method enables the network to learn an appropriate size and position of the window for each layer and head, and our newly introduced scheme can further control the window size depending on the future and past contexts. Thus, it can save both computational complexity and memory size from the square order of the input length to the adaptive linear order. We show the effectiveness of the proposed method by using several ASR tasks, and the proposed adaptive span methods consistently improved the performance from the conventional fixed span methods.
Index Terms: Self-attention, adaptive, Transformer, end-to-end, speech recognition,

1. Introduction

Deep neural networks (DNN) have significantly improved the performance of automatic speech recognition (ASR) in the last decade [1] bridging the performance gap between humans and machines [2–4]. Furthermore, the end-to-end (E2E) ASR paradigm has become quite popular because of its simplicity [5–7] in recent years. One type of successful E2E ASR model is based on the encoder-decoder framework [7], where the encoder maps the acoustic features to some hidden representation and the attention-based decoder generates the output tokens one at a time. The encoders and decoders typically use the long short-term memory recurrent neural networks (LSTM-RNNs) [7–9] because LSTM can learn long term sequential information.

Transformer is a new sequence modeling architecture proposed for neural machine translation [10] and it achieved significant performance improvement over LSTM-RNNs based E2E models. It has been explored in a lot of other natural language processing tasks, such as language modeling [11, 12]. Subsequently, Transformer is also proven to be effective for E2E ASR [13,14] as well. One reason for the success of Transformer models is that it can capture the long term sequence dependency better than LSTMs. Because at each time step, the self-attention layer computes the weighted sum over the whole input sequence given the attention weights. In addition, unlike the LSTMs which use recurrent connections for capturing sequential information, Transformers can process the input sequence in parallel via self-attention, which enables efficient GPU training.

Even though Transformer has a lot of advantages, it has serious computational and memory cost issues. In the computation of self-attention, the input sequence of length T is first mapped to the multiple key (K), query (Q) and value (V) sequences of the same length T. Then, the similarity scores are computed between every key and query state. And the scores are normalized with Softmax to get the attention weights. Afterwards, the weighted summation computation is performed with the attention weights and the V sequence. The corresponding computational and memory complexities are both $\mathcal{O}(T^2)$ in the above self-attention process. This cost becomes a barrier to apply Transformer in many tasks where the input sequence length can be inevitably long as was mentioned in [15–24]. The majority of the research direction to avoid the cost is to compute the self-attention at each time step only on a *fixed span* of the subset from the whole input sequence [16–19], e.g., [19] used a local self-attention of fixed size for masking-based speech enhancement. In [21], a more flexible method called the *adaptive span* was proposed where the sizes of attention span at every layer are parameters learnt during training. In [22], a method called the Hashing attention was proposed to compute the self-attention only on the similar key and query states.

Among these techniques, in this paper, we propose to use the adaptive span technique [21] to E2E ASR Transformer. The adaptive span was originally proposed for language modeling to determine the proper span size of each head and layer. However, we found that the adaptive span is instinctively appropriate for ASR applications because the features that are temporally close to each other usually have a higher correlation in ASR, unlike language modeling or other common natural language tasks. One of the problems when applying it to ASR Transformer is that the original adaptive span used for language modeling only considers the history context, while in ASR, it is important to flexibly control the history and future context spans separately. Thus, we newly introduce another learning parameter, span ratio, to decide the relative position of the span window. Finally, we examine the proposed method on several E2E ASR tasks, including the common single-speaker and the multi-speaker corpora. The proposed method was superior/comparable to the conventional methods including whole sequence and fixed span methods, while reducing both computational and memory complexity theoretically. We also experimentally validated the reduction in the time complexity.

2. Adaptive Self-attention

This section first describes a standard self-attention in the Transformer layers, which is computed over the whole input sequence. Then, the proposed adaptive self-attention is described,

http://dx.doi.org/10.21437/Interspeech.2020-2816

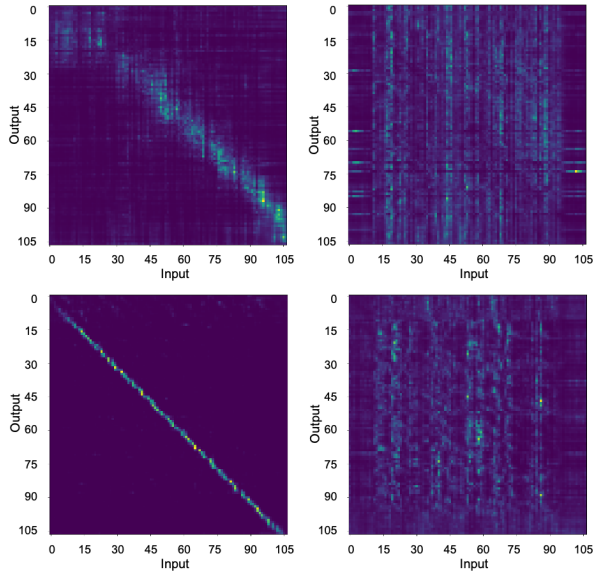

Figure 1: *An example of the attention patterns at four heads of the first encoder layer in the whole sequence self-attention model. The horizontal axis and vertical axis represent the input and output, respectively. It shows that the two attention heads shown on the right tend to attend to the context frames over long spans, while the two heads shown on the left tend to attend to the context frames locally with a short span.*

which adaptively learns the span size and span ratio.

2.1. Whole Sequence Self-Attention in Transformer

The self-attention is composed of scaled dot product attention which receives three state sequences, namely the key ($\mathbf{K}$), query ($\mathbf{Q}$), and value ($\mathbf{V}$). $\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$ usually have the same dimension as $T \times d^{\text{att}}$, where T is the sequence length. To compute the attention weights $\mathbf{A}$, we first compute the normalized dot product between key $\mathbf{K}$ and query $\mathbf{Q}$, and apply a Softmax function, as follows:

$$\text{score}(\mathbf{Q}, \mathbf{K}) = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d^{\text{att}}}} \in \mathbb{R}^{T \times T}, \tag{1}$$

$$\mathbf{A} = (a_{t,i}) = \text{Softmax}(\text{score}(\mathbf{Q}, \mathbf{K})),$$

$$a_{t,i} = \frac{\exp(\text{score}_{t,i})}{\sum_j \exp(\text{score}_{t,j})}, \tag{2}$$

where the term $\sqrt{d^{\text{att}}}$ is used for scaling the dot product to avoid a very large magnitude because of large dimension d^{att}. The term $\text{score}_{t,i}$ in Eq. (2) represents the similarity score between the query vector at t-th frame of $\mathbf{Q}$ and the key vector at i-th frame of $\mathbf{K}$. The output of the scaled dot product attention block is computed by multiplying the attention weights A in Eq. (2) with the value $\mathbf{V}$ as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\text{score}(\mathbf{Q}, \mathbf{K})\right)\mathbf{V}. \tag{3}$$

This is the basic calculation of the attention network with the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ representation.

Multi-head attention (MHA) is often used in the self-attention of Transformer to capture the information from different representation subspaces. Linear projections are used to produce the variants of $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ depending on head h. After the attention computation in Eq. (3) for each h, we obtain the context vector $\mathbf{H}_h$, and the final output is computed by using the concatenated context vectors across all heads as:

$$\mathbf{H}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^{\text{q}}, \mathbf{K}\mathbf{W}_h^{\text{k}}, \mathbf{V}\mathbf{W}_h^{\text{v}}), \tag{4}$$

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}([\mathbf{H}_h]_{h=1}^{d^{\text{head}}})\mathbf{W}^{\text{head}}, \tag{5}$$

where d^{head} is the number of attention heads, and $\mathbf{W}^{\text{head}} \in \mathbb{R}^{(d^{\text{head}} d^{\text{att}}) \times d^{\text{att}}}$, and $\mathbf{W}_h^{\text{q}}, \mathbf{W}_h^{\text{k}}, \mathbf{W}_h^{\text{v}} \in \mathbb{R}^{d^{\text{att}} \times d^{\text{att}}}$ are projection weight matrices. Self-attention is the above multi-head attention layer with the scaled dot product attention whose $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are the same, i.e., $\mathbf{K} = \mathbf{Q} = \mathbf{V} \triangleq \mathbf{X}$, and $\text{MHA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$.

2.2. Fixed Span Self-Attention

The computational complexity of the Attention$(\cdot)$ in Eq. (3) is the order of the square of the sequence length, i.e., $\mathcal{O}(T^2)$ and the memory used to store the attention weights is also the square of the sequence length, T^2. This limits the use of Transformer when the input length is very large, which normally happens in the speech feature sequence. One straightforward approach to alleviate this problem is to limit the span size as a relatively small value W, so that the self-attention is performed on a local segment [19]. Then the computation in Eq. (3) is adjusted as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\text{score}(\mathbf{Q}, \mathbf{K}')\right)\mathbf{V}', \tag{6}$$

where the corresponding keys and values at each time step t are $\mathbf{K}' = \mathbf{K}_{t-W_l:t+W_r}$ and $\mathbf{V}' = \mathbf{V}_{t-W_l:t+W_r}$, respectively. W_l and W_r denotes the left and right context window, respectively. With $W = W_l + W_r$, the computational and memory costs potentially become a linear order, i.e., $\mathcal{O}(TW)$. The costs can be considerably reduced if we set $W \ll T$.

However, the behavior of each head at every layer is not necessarily the same, and using a single span size hyperparameter W (or W_l and W_r) for all the self-attention computations is not appropriate. As shown in Figure 1, the spans of the dominant attention values at each head are quite different. Some heads focus on the long sequence information, while others use only local information. Such trends can be also observed at each layer. Therefore, this observation shows that we can reduce the span size at certain heads and layers.

2.3. Adaptive Self-attention

Adaptive span. In this part, we first introduce the method called the adaptive span. Inspired from [21], the motivation is to learn the appropriate span size at each self-attention head and layer during training. For each attention head, an adaptive span $W_\theta \in [0, W]$ is employed to control the proper span size that allows for the information propagation. The maximum span size W is set in advance. To apply the adaptive span, a soft mask $m(t, i) \in [0, 1]$ is used in computing the attention weights. Then, Eq. (2) is reformulated as:

$$a_{t,i} = \frac{m(t,i)\exp(\text{score}_{t,i})}{\sum_j m(t,j)\exp(\text{score}_{t,j})}, \tag{7}$$

$$m(t,i) = \min\left\{\max\left\{\frac{1}{R}(R + W_\theta - \text{abs}(t-i)), 0\right\}, 1\right\}, \tag{8}$$

where $\text{abs}(\cdot)$ is the absolute value and R is hyperparameter for buffer. To regularize the adaptive span size, we use a penalty function, a ℓ_1 loss by aggregating all the span sizes

Table 1: *Time complexity of each attention methods. We denote B as batch size, H as number of heads, T as sequence length, W/$W_\theta(h,s)$ as fixed and adaptive span sizes for each head h and layer s, respectively.*

Model	Time
whole-seq	$\mathcal{O}(BHT^2)$
fixed-span	$\mathcal{O}(BHTW)$
adaptive-span	$\mathcal{O}(BHT\max(W_\theta(h,s)))$

$\mathcal{L}_{\text{span}} = \sum_{h,s} W_\theta(h,s)$ where h and s are head and layer indexes, respectively. Note that the same span sizes on the left and right context are used for all the heads at every layer based on this formulation. In Table 1, we summarize the time complexity for three different attention methods mentioned above, which shows that span attention avoids the square of sequence length with a maximum span size term instead.

Adaptive ratio. This paper proposes to extend the previous adaptive span by introducing additional learning parameter, called the adaptive ratio γ, to adaptively control the relative size between the left and right context frames in the encoder. This corresponds to adaptively determine W_l and W_r introduced in Section 2.2 for each attention head. This extension is quite suitable for ASR since the importance of the context information is asymmetric in the ASR case, and the previous (history) contexts have more important information in general [25]. If we denote the proportion of the left side span as γ, then the corresponding size $W_\theta^l = W_\theta \times \gamma$. Accordingly, the right side span in the future is $W_\theta^r = W_\theta \times (1-\gamma)$.

Similar to the regularization of the span size, in order to regularize the span ratio term, we use another penalty function in terms of the span ratio to control the trade-off between the left and right span. In our initial experiments, we observed that the left span is more important, as we expected. In order to encourage the network to use more history information, we limit the γ by setting the loss as $\mathcal{L}_{\text{ratio}} = 1 - \text{Mean}(\{\gamma(h,s)\}_{h,s})$

During training, we minimize the total loss function comprised of three terms:

$$\mathcal{L} = \mathcal{L}_{\text{ASR}} + \lambda(\mathcal{L}_{\text{span}} + \mathcal{L}_{\text{ratio}}), \tag{9}$$

where $\lambda > 0$ is a regularization hyperparameter. In this experiment, we used the value $1e-7$.

3. Experiments

3.1. Setup

To evaluate the effectiveness of the adaptive span and ratio, we conducted experiments on several ASR corpus, including AIShell [26], Switchboard [27] and TED-LIUM2 [28] for single-speaker ASR tasks and WSJ-2mix [29] for multi-speaker ASR task. Our implementation was based on ESPnet [30] [1].

As a benchmark, we first trained the normal Transformer with whole-sequence self-attention model on each of these datasets. We followed the ESPnet Transformer recipe to set the hyper-parameters of the model. The encoder of the speech recognition model contains a two-layer CNN and twelve layers self-attention network (SAN) blocks. The decoder contains six layers of self-attention network (SAN) blocks. The parameters of the SAN are: $d^{\text{head}} = 4, d^{\text{att}} = 256, d^{\text{ff}} = 2048$ for the number of heads, dimension of attention (att) and dimension of feed forward (ff) layer, as introduced in Section 2.1. We temporarily set the hyperparameter $R = 2$ that is mentioned in Section 2.3

[1] We plan to make our implementation open source for public use.

3.2. Results on AIShell

In this part, we present the performance on the AIShell corpus, an open-source Chinese Mandarin speech corpus, which is shown in Table 2. The SAN with whole-sequence model achieves state-of-the-art character error rates (CERs), 6.0% and 6.6% on the development and evaluation set, respectively. Then, we explore the performance of the shorter context. In the SAN with the fixed span, if the span size in the encoder is set to be 50 and that in the decoder to be 25, the CERs are 6.2% and 6.9%, which is a slight degradation. Here, the span sizes are the same for all the heads at every layer. If we reduce the span sizes to be $[\text{enc} = 29, \text{dec} = 14]$[2], we could maintain the CERs but can reduce the memory and computational cost.

Next we trained the SAN with adaptive span size. We set the maximum span size to be $[50, 25]$ as in the experiment of fixed span and the span ratio $\gamma = 0.7$ [3]. The character error rates are 6.2% and 6.9% on the development and evaluation sets respectively, which is almost similar to the fixed span one. When we checked the estimated average span size over all the heads and layers in the encoder, it is less than 50, around 29. If we further apply the adaptive span ratio, the CERs became better, achieving 6.0% and 6.7%, which are very close to the performance of whole sequence SAN, but we could significantly save the computational cost.

Note that the following experiments on the other corpora used the same hyperparameters in the adaptive/fixed span SANs.

Table 2: *Performance comparison between the whole sequence SAN, fixed-span SAN and the adaptive-span SAN. Character error rates (CERs) [%] on the AIShell corpus.*

Model	Max-span	Span-ratio	dev	eval
Kaldi chain TDNN [25, 31]	[7,3]	n/a	n/a	7.4
ESPnet Transformer [14]	inf	n/a	6.0	6.7
Whole-seq (ours)	inf	n/a	6.0	6.6
Fixed-span	[29, 14]	[0.5:0.5]	6.2	7.0
Fixed-span	[50, 25]	[0.5:0.5]	6.2	6.9
Adaptive-span	[50, 25]	[0.7:0.3]	6.2	6.9
Adaptive-span	[50, 25]	adaptive	**6.0**	**6.7**

3.3. Results on Switchboard and TED-LIUM2

The performance on both Switchboard and TED-LIUM2 in Table 3 show quite similar trends to the AIShell result in the previous section. First, the whole sequence SAN achieves reasonable performance. The fixed-span with span sizes $[29, 14]$ results in degradation. If we use the adaptive span SAN, the performance was significantly improved from that of the fixed span SAN, and became close to that of the whole sequence SAN. Notably, half of the adaptive span results with the fixed span-ratio even outperformed the whole sequence SAN results, although we need further hyperparameter tuning for the Switchboard experiment to show the effectiveness of the adaptive span ratio.

3.4. Results on WSJ-2mix

In this part, we investigate the effectiveness of the windowed SAN with the standard multi-speaker speech recognition dataset, WSJ-2mix [32]. We used an end-to-end multi-speaker

[2] These numbers come from the average span size obtained by the adaptive span experiments to make the fair comparisons between adaptive and fixed span methods.

[3] For fixed span SAN, $\gamma = 0.5$ works better than $\gamma = 0.7$ but for adaptive span SAN, it is the opposite.

Table 3: *Performance comparison between the whole sequence SAN, fixed-span SAN and the adaptive-span SAN. Word error rates (WERs) [%] on the Switchboard and TED-LIUM2 corpus.*

Model	Max-span	Span-ratio	SWBD							TED-LIUM2	
			train_dev	eval2000			rt03			dev	test
				callhm	ctm	swbd	ctm	fsh	swbd		
Whole-seq	inf	n/a	11.9	17.2	12.8	8.3	14.8	11.2	18.1	11.6	10.3
Fixed-span	[29, 14]	[0.5:0.5]	11.7	17.3	13.0	8.7	15.3	11.8	18.5	12.1	11.1
Adaptive-span	[50, 25]	[0.7:0.3]	**11.5**	**16.5**	**12.5**	**8.4**	**14.8**	**11.3**	**18.0**	11.7	**11.0**
Adaptive-span	[50, 25]	adaptive	11.7	16.9	12.7	**8.4**	14.9	11.6	18.1	**11.6**	**11.0**

Table 4: *An example of average span sizes in the encoder of Adaptive-span SAN trained on WSJ-2mix dataset. SD_i^s represents the i-th layer for s-th speaker differentiating encoder. R_i represents the i-th layer for speech recognition encoder.*

Layer	SD_1^1	SD_2^1	SD_3^1	SD_4^1	SD_1^2	SD_2^2	SD_3^2	SD_4^2	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
Average	53.1	72.3	76.0	76.0	57.9	72.1	67.5	67.5	58.2	58.2	46.8	39.5	45.4	48.8	41.0	58.0

Table 5: *Performance comparison between the whole sequence SAN, fixed-span SAN and the adaptive-span SAN. Word error rates (WERs) [%] on the WSJ-2mix corpus.*

Model	Max-span	Span-ratio	dev	eval
Whole-seq	inf	n/a	15.4	11.4
Fixed-span	[29, 14]	[0.5:0.5]	24.6	21.7
Fixed-span	[75, 25]	[0.5:0.5]	18.8	14.1
Adaptive-span	[75, 25]	adaptive	**18.5**	**13.7**

ASR system with Transformer [19], which has special speaker-differentiating SAN layers to perform speech separation inside the network. Note that multi-speaker speech recognition is more challenging because the implicit speech separation in some hidden representation requires more global context information related to speaker characteristics in general. Thus, this task is more sensitive for the use of the context information, and quite suitable to test our windowed SAN techniques,

Table 5 shows that the whole sequence SAN is distinctly better than the other models, as we expected. This shows the importance of global context for speech separation. The importance of the global context is also partly supported by the fixed span model results with the small and large span sizes, as increasing the span sizes to be $[75, 25]$ could recover the performance degradation to some extent. If we use the adaptive span SAN with the adaptive ratio, it can further reduce the WERs, while also reducing the average span size. In Table 4, we show the average of trained span size at every layer in the encoder. Interestingly, the average span size learned from the adaptive span SAN was relatively large in the speaker-differentiating (SD) encoder layers, which perform speech separation, than that of the normal SAN layers (R). This result indicates that the proposed adaptive span SAN can adaptively change the span size depending on the network roles (i.e., separation and recognition roles in this experiment).

3.5. Computational efficiency

One of the strong benefits of the windowed attention is to reduce the computational cost during inference, thanks to the adaptive span size and ratio, especially for the long input sequences. We present the average computation time of the self attention unit for the whole sequence self-attention and the self-attention with maximum span size of 50 in Figure 2 by using the TED-LIUM2

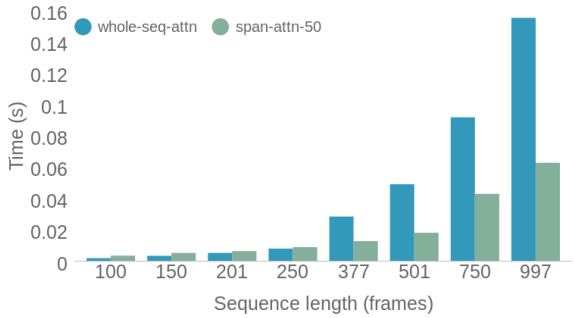

Figure 2: *Average computation time of the self attention unit with the whole sequence and the adaptive span methods over different lengths of speech sequences.*

test examples. The proposed adaptive span self-attention can significantly save the time compared to the whole sequence self-attention for long sequences as in this case "$W \ll T$" like explained in Section 2.2. For example, given a sequence of length 997 after subsampling (almost 40 seconds), self-attention with span saves more than 50% of the computation time.

4. Conclusions

In this paper, we applied the windowed self-attention to the E2E ASR tasks, as an alternative to the whole sequence self-attention. From the observation of attention patterns in normal SAN based ASR system, it is not necessary to attend to the whole sequence in the self-attention computation, which wastes the computation. In order to choose an appropriate span size, we introduced the adaptive span attention technique. We also proposed to adjust the span ratio using another learning parameter. The results show that span attention generally has small impact on the performance in single-speaker speech recognition tasks, which indicates that it does not require the whole sequence in the self-attention computation. For application, it is one possible direction to use the span size and ratio parameters after training as a guide in designing other model architectures, such as the kernel size of a convolutional neural network. In the future, we will focus on online/streaming ASR based on this adaptive span technique. To do that, we will apply the adaptive span technique in the source-target attention to make an entire network adaptive [33–36].

5. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012.

[2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The Microsoft 2016 conversational speech recognition system," in *Proc. IEEE ICASSP*, Mar. 2017, pp. 5255–5259.

[3] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[4] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim *et al.*, "English conversational telephone speech recognition by humans and machines," *Proc. Interspeech 2017*, pp. 132–136, 2017.

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

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

[7] 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.

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

[9] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. IEEE ICASSP*, Mar. 2017, pp. 4835–4839.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI preprint*, Jun. 2018.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.

[13] L. Dong, S. Xu, and B. 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.

[14] 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," *arXiv preprint arXiv:1909.06317*, 2019.

[15] B. Yang, Z. Tu, D. F. Wong, F. Meng, L. S. Chao, and T. Zhang, "Modeling localness for self-attention networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4449–4458.

[16] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," in *Proc. of Interspeech*, 2018, pp. 3723–3727.

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

[18] N. Moritz, T. Hori, and J. L. Roux, "Streaming automatic speech recognition with the transformer model," in *Proc. of ICASSP*, 2018, pp. 6074–6078.

[19] X. Chang, W. Zhang, Y. Qian, J. Le Roux, and S. Watanabe, "End-to-end multi-speaker speech recognition with transformer," in *Proc. IEEE ICASSP*. IEEE, 2020, pp. 6134–6138.

[20] J. Kim, M. El-Khamy, and J. Lee, "Transformer with gaussian weighted self-attention for speech enhancement," *arXiv preprint arXiv:1910.06762*, 2019.

[21] S. Sukhbaatar, É. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers," in *Proc. ACL*, 2019, pp. 331–335.

[22] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[23] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. 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.

[24] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[25] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. of Interspeech*, 2015, pp. 3214–3218.

[26] 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.

[27] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 1992, pp. 517–520.

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

[29] H. Seki, T. Hori, S. Watanabe, J. Le Roux, and J. R. Hershey, "A purely end-to-end system for multi-speaker speech recognition," in *Proc. ACL*, Jul. 2018.

[30] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "ESPnet: End-to-End Speech Processing Toolkit," in *Proc. ISCA Interspeech*, 2018, pp. 2207–2211.

[31] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding (ASRU*, 2011, accessed at May 17, 2020. [Online]. Available: https://github.com/kaldi-asr/kaldi/blob/master/egs/aishell/s5/RESULTS

[32] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *IEEE ICASSP*, 2016, pp. 31–35.

[33] A. Tjandra, S. Sakti, and S. Nakamura, "Local monotonic attention mechanism for end-to-end speech and language processing," *arXiv preprint arXiv:1705.08091*, 2017.

[34] S. Zhang, E. Loweimi, P. Bell, and S. Renals, "Windowed attention mechanisms for speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7100–7104.

[35] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Towards online end-to-end transformer automatic speech recognition," *arXiv preprint arXiv:1910.11871*, 2019.

[36] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online CTC/attention end-to-end speech recognition architecture," in *IEEE ICASSP*, 2020, pp. 6084–6088.