ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413535

DEVELOPING REAL-TIME STREAMING TRANSFORMER TRANSDUCER FOR SPEECH RECOGNITION ON LARGE-SCALE DATASET

Xie Chen[*1], *Yu Wu*[*2], *Zhenghao Wang*[1], *Shujie Liu*[2], *Jinyu Li*[1]

[1]Microsoft Speech and Language Group
[2]Microsoft Research Asia

ABSTRACT

Recently, Transformer based end-to-end models have achieved great success in many areas including speech recognition. However, compared to LSTM models, the heavy computational cost of the Transformer during inference is a key issue to prevent their applications. In this work, we explored the potential of Transformer Transducer (T-T) models for the fist pass decoding with low latency and fast speed on a large-scale dataset. We combine the idea of Transformer-XL and chunk-wise streaming processing to design a streamable Transformer Transducer model. We demonstrate that T-T outperforms the hybrid model, RNN Transducer (RNN-T), and streamable Transformer attention-based encoder-decoder model in the streaming scenario. Furthermore, the runtime cost and latency can be optimized with a relatively small look-ahead.

Index Terms— Transformer, Transducer, Real-time decoding, Speech Recognition

1. INTRODUCTION

Great progress has been made to automatic speech recognition (ASR) with end-to-end (E2E) models [1, 2, 3, 4, 5, 6, 7, 8, 9]. Currently, Transducer (e.g., recurrent neural network Transducer (RNN-T) [10]) and Attention-based Encoder-Decoder (AED) [1, 11, 12] are two most popular types of E2E methods. AED models achieve very good performance thanks to the attention mechanism, however they are not streaming in nature by default and there are several studies towards that direction, such as monotonic chunk-wise attention [13] and triggered attention [14, 15]. In contrast, because of the streaming nature, transducer models especially RNN-T received a lot of attention for industrial applications and have also managed to replace traditional hybrid models for some cases [7, 8, 16].

While RNN with long short-term memory (LSTM) [17] units was widely used in the E2E works, the transformer architecture with self attention [18] has recently become the fundamental building block for E2E models [19, 20, 21]. In addition to accuracy, streaming recognizer and runtime computational cost are two crucial factors for deploying high quality automatic speech recognition (ASR) system in industry. In this work, we focus on developing streaming Transformer transducer (T-T) [22] and its variation with high accuracy and low computation cost. The computational cost of Transformer Transducer grows significantly with respect to the input sequence length, which obstacles the practical use of T-T. Recently conformer Transducer (C-T) [23] was proposed to further improve T-T, but it is not streamable because its encoder has attention on full sequence.

Existing methods may partially address these issues, but have their own drawbacks. 1) Time-restricted method [22, 24, 25, 26] simply masks left and right context in Transformer to control time cost. As the reception field grows linearly with the number of transformer layers, a large latency is introduced with the strategy. 2) chunk-wise method [27, 15] segments the input into small chunks and operates speech recognition on each chunk. However, the accuracy drops as the relationship between different chunks are ignored. 3) Memory based method [28, 29] employs a contextual vector to encode history information while reducing runtime cost by combining with chunk-wise method. However, the method breaks the parallel nature of Transformer in training, requiring a longer training time.

In this paper, our goal is to develop streaming Transformer and Conformer Transducer models that can be operated in real time. We wish to reach a balance between training cost, runtime cost, and accuracy. We combine Transformer-XL and chunk-wise processing to handle streaming scenario, and there is no overlap between chunks in training to guarantee the training efficiency. We can finish training 65 thousand hours of anonymized training data in 2 days with mixed precision on 32 V100 GPUs. T-T and C-T outperform hybrid model, RNN-T, and streaming Transformer attention-based encoder-decoder model over 10% relative word error rate (WERR) on the streaming speech recognition evaluation. In term of runtime cost, our proposed method uses limited history while maintaining the same performance (1% WERR degradation). If a small look-ahead is allowed, such as 360ms, T-T reaches 0.25 real-time factor rate on CPU that satisfies the industry requirement of a real application.

2. MODEL STRUCTURE

2.1. Transducer Architecture

In this paper, we investigate the use of transducer model [30] for real-time and streaming speech recognition. A transducer has three components, an acoustic encoder network (encoder), a label predictor network (predictor), and a joint network. The acoustic feature sequence $\mathbf{x}_1^t$ is fed into the encoder to get encoder output $\mathbf{f}_t$, correspondingly the previous label sequence $\mathbf{y}_1^{u-1}$ are sent to the predictor to compute the predictor output $\mathbf{g}_{u-1}$. The outputs of encoder and predictor are then added by the joint network. A non-linear function, such as relu function, is applied before sending to softmax function to compute the probability distribution over the sentence piece vocabulary. The computation formulas in transducer could be written as below,

$$\begin{aligned} \mathbf{f}_t &= \text{encoder}(\mathbf{x}_1^t) \\ \mathbf{g}_{u-1} &= \text{predictor}(\mathbf{y}_1^{u-1}) \\ \mathbf{h}_{t,u-1} &= \text{relu}(\mathbf{f}_t + \mathbf{g}_{u-1}) \\ P(y_u|\mathbf{x}_1^t, \mathbf{y}_1^{u-1}) &= \text{softmax}(W_o * \mathbf{h}_{t,u-1}) \end{aligned} \quad (1)$$

In practice, we can employ different architectures for encoder

*Equal Contribution

and predictor. For example, in [30], LSTM is used for both encoder and predictor and is widely known as RNN-T model structure. In this paper, we use transformer as the encoder, and LSTM as the predictor due to the consideration of speed and memory cost.

2.2. Transformer and Conformer

In the past several years, the transformer model [18] has proven to present significant performance improvement over LSTM in a range of tasks [31, 32]. Very recently, the transducer using transformer were proposed and reported to outperform LSTM based transducer models [32, 22, 33]. In literature, to distinguish RNN-T which adopted RNN as encoder and decoder, the transformer based transducer is normally called transformer transducer (T-T).

The transformer model adopts the attention mechanism to capture the sequence information. Self-attention is used to compute the attention distribution over the input sequences with a dot-product similarity function, which could be written as,

$$\begin{aligned} \alpha_{t,\tau} &= \frac{\exp(\beta(W_q\mathbf{x}_t)^T(W_k\mathbf{x}_\tau))}{\sum_{\tau'}\exp(\beta(W_q\mathbf{x}_t)^T(W_k\mathbf{x}_{\tau'}))} \\ &= \text{Softmax}(\beta\mathbf{q}_t^T\mathbf{k}_\tau) \\ \mathbf{z}_t &= \sum_\tau \alpha_{t\tau}W_v\mathbf{x}_\tau \\ &= \sum_\tau \alpha_{t\tau}\mathbf{v}_\tau \end{aligned} \tag{2}$$

where $\beta = \frac{1}{\sqrt{d}}$ is a scaling factor. The input vector $\mathbf{x}_t$ is sent to three different matrices and the outputs are used as query $\mathbf{q}_t$, key $\mathbf{k}_t$ and value $\mathbf{v}_t$ respectively in the attention module. In the transformer model, multi-head attention (MHA) is applied to further improve the sequence model capacity, where multiple parallel self-attentions are applied on the input sequence and the outputs of each attention module are then concatenated. The range of input sequence for the softmax function in Equation 2 can be controlled by applying a mask. If we only want to use the current and previous frames $\mathbf{x}_1^t$ to compute $\mathbf{z}_t$, the attention weights $\alpha_{t,\tau}$, where $\tau > t$, could be masked to be 0. Hence, the use of mask provides a flexible approach to decide the scope of input sequence for computation. The details of the mask design used in this paper can be found in Section 3.2. In each transformer layer, it also contains two fully-connected feed-forward networks (FFN), a nonlinear activation, layer normalization, and residual connections. A transformer-based audio encoder usually stacks multiple transformer layers, e.g. 18 layers.

In transformer model, the position embedding is normally used to explicitly model the ordering information of input sequence. Relative position embedding was found to yield better performance compared to absolute position embedding [34, 31, 32]. The motivation is the offset between two frames should be considered in the attention weight calculation, and the offset is modeled by the relative position embedding. For efficiency, we use a simple but effective relative position embedding, which is formulated as

$$\mathbf{z}_t = \text{Softmax}(\beta\mathbf{q_t^T}(\mathbf{k}_\tau + \mathbf{p_{t,\tau}}))\mathbf{v}_\tau \tag{3}$$

where $\mathbf{p}_{t,\tau}$ is the relative position embedding obtained from a lookup table. The implementation is more efficient and memory friendly than the relative position embedding used in [31].

The Transformer model captures global context, but the local information does not model very well. A few recent work [23] shows that the marriage of CNN and Transformer improves the SR performance. Among them, Convolution augmented Transformer (a.k.a Conformer) [23] is a typical one, which inserts a special CNN based structure into each Transformer block, which achieves state-of-the-art performance on Librispeech. We adopt the original structure of [23] in our implementation but changing the depth-wise CNN to causal depth-wise CNN to avoid extra latency.

3. STREAMING TRANSFORMER-TRANSDUCER IN REAL-TIME

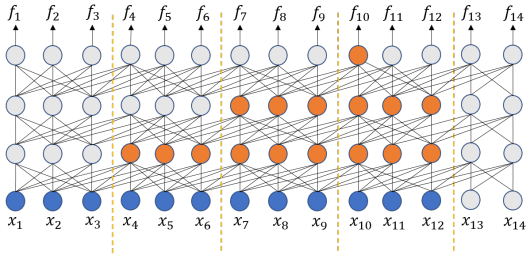
3.1. Challenges in Streaming Scenario

During inference, the computational cost is a potential issue if the full history is used for the computation of each frame. The computation of $\mathbf{z}_t$ in Equation 2 will increase linearly with t as it needs to compute the attention weights from the first to the current frames. As a result, it makes the overall computation complexity quadratic, which is not affordable for long utterances. Furthermore, there is a trade-off between the model accuracy and latency, if we could allow several look-ahead frames for the computation of the current frame, performance improvement could be achieved. The latency controlled technique has been applied widely in Hybrid systems [35] and also introduced into transducer models recently [33, 22]. In order to make computation feasible and achieve better performance, **truncated history** and **limited future information** can be used for transformer transducer. In [33, 22], the authors set the attention mask to allow a specific number of contextual frames in each transformer layer for both history and the future. This is able to reduce the computation in each layer efficiently. However, there is a potential drawback for the fixed number of future context. With the increase of transformer layers, the number of future context increases linearly. If the transformer has 18 layers and the future context is 5 frames per layer, a latency of 90 frames will be introduced as a result.

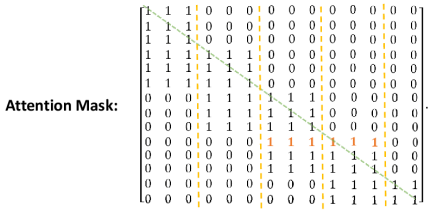
3.2. Attention Mask Design for Training Streaming Models

We design a simple but effective mask strategy to truncate history and allow limited future information. Transformer uses attention mechanism for sequence modeling and the attention mask can be applied on the attention weight matrix $\{\alpha_{t,\tau}\}$, to determine the range of input sequence involved for computation. When α_t, τ is set to 0 by the attention mask, then the input at time τ, $\mathbf{x}_\tau$, won't be used for the computation of output $\mathbf{z}_t$ at time t. The mask matrix is created with the following rules. First, the input acoustic feature sequence is segmented into chunks with a specified chunk size.[1] Then we construct the matrix according to the following rules: 1) Frames in the same chunk can see each other. For example, when computing the encoder output for frame $\mathbf{x}_{10}$, all frames belonging to the same chunk, including $\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}$, are considered in the attention calculation. x_{10} can see two future frames, while x_{12} has zero frame lookahead. The average number of lookahead frames for each frame is the half of the chunk size. 2) If two frames are in different chunks, the left one cannot see the right one in attention calculation, even if the offset is one. By this means, the chunk boundary will strictly restrict reception field, avoiding the linear reception field growing as the model becomes deeper. 3) If two frames are in different chunks, the right one can see the left one if their distance is less than the history window size. This method makes the left reception field grow linearly (history window size) with the model becoming deeper. Figure 1b gives an example of the streaming mask matrix. This attention mask is shared across all transformer layers.

[1] Chunks do not have overlap for efficient training.



(a) The reception field of position x_{10}. The left reception field grows with the number of Transformer layers, whereas the right reception field does not.



(b) Attention mask matrix M per layer. If $M(i,j)$ is 1, then the jth input will be used for computation in ith frame.

Fig. 1: An example of our mask strategy. Chunk size and history window size are both 3 in the example.

Figure 1a gives the reception field of the position x_{10}. We can find that the advantage of the masking strategy is that it allows the left context linearly increase while forbids the right reception field grows, so the model can use long history information while restrict future look-ahead. In the example, the left reception field grows 3 frames per layer, while the right context is restricted to x_{12} for all layers. Moreover, the masking strategy is very flexible, it can simulate most of the possible scenarios we will use in practice. When chunk size is one and history window is infinity, it simulates the naive zero look-ahead scenario.

3.3. Inference Optimizations

We use the following engineering optimizations for inference.

Caching: given the input acoustic frame at time t, $\mathbf{x}_t$, in order to compute the transformer encoder output $\mathbf{f}_t$, in each transformer layer, in addition to the linear matrix multiplications and non-linear functions, we also need to compute the attention weight over the input sequence and then sum over the weighted value vectors according to Equation 2. In order to avoid repeated computation, some intermediate variables are cached. Specifically, in Equation 2, the key $\mathbf{k}_\tau = W_k\mathbf{x}_\tau$ and value $\mathbf{v}_\tau = W_v\mathbf{x}_\tau$ in each layer are cached once computed. As a result, we only need to calculate the query $\mathbf{q}_t = W_q\mathbf{x}_t$, key $\mathbf{k}_t$ and value $\mathbf{v}_t$, and then applied softmax function over the input sequences using the cached key and value, for the acoustic frame at time t in each layer. In addition, as we used truncated history for transformer in each frame, the memory consumption for caching won't increase with the increase of t.

Chunk-wise compute: if the latency of several frames are allowed, as shown in Figure 1, we could group the input frames $[\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}]$ as a small minibatch and feed to the transformer encoder to compute $\mathbf{f}_{10}, \mathbf{f}_{11}$ and $\mathbf{f}_{12}$ simultaneously. The key and value in each layer will be cached once computed, which will be used in the computation of the future frames. In this way, the efficient matrix operation could be applied, instead of applying the matrix-vector operation multiple times. It is worth mentioning that, for the transformer encoder with zero lookahead, we can manually introduce a latency of several frames and significant speedup could be achieved by applying this chunk-wise computation.

4. EXPERIMENT

4.1. Experiment Setup

We used 65 thousand (K) hours of transcribed Microsoft data as the training data. The test set covers 13 different application scenarios such as Cortana, far-field speech and call center, consisting of a total of 1.8 million (M) words. The word error rate (WER) averaged over all test scenarios are reported. All the training and test data are anonymized data with personally identifiable information removed. 4000 sentence pieces trained on the training data transcription was used as vocabulary. We applied a context window of 8 for the input frames to form a 640-dim feature as the input of transducer encoder and the frame shift is set to 30ms. Utterances longer than 30 second were discarded from the training data.

For the RNN-T, the encoder contains 6 LSTM layers and the predictor consists of one embedding layer and 2 LSTM layers. The dimension of embedding and LSTM layers are set to 1024. In terms of T-T, 18 transformer layers with 720 hidden nodes and 1024 feed-forward nodes were used as encoder; Same as RNN-T, 2 LSTM layers with 720 hidden nodes were used as predictor. C-T chose 640 as the hidden layer size to get similar model size as RNN-T and T-T. The kernel size in C-T is 3. Relative position encoding is used for T-T. All the models are trained from scratch and with mixed precision for efficient training. The number of model parameter for various transducer models is around 80M.

The runtime speed is evaluated on a single CPU machine containing 16 cores, with Intel Xeon CPU e5-2620, 2.10 GHz, and 64GB memory. We randomly sample 500 utterances from the test set to measure the run-time factors. The average length of 12.7s for these utterances. Float precision is used for evaluating LSTM and transformer transducer models in the following experiment without an explicit statement. For the beam search of transducer decoder, n-best is set to 5 for all experiments. In order to conduct efficient decoding for transducer models, an efficient transducer decoder based on beam search was implemented with C++. The details on the decoding algorithm can be found in [30]. The transducer models used in this paper, including LSTM, transformer based transducers, are trained with Pytorch, then exported using Just-In-Time (JIT) compilation with Libtorch. These JIT exported models can be evaluated in the decoder implemented with C++ on CPU conveniently and efficiently. Real time factor (RTF) is used to evaluate efficiency.

4.2. Evaluation Result

4.2.1. Evaluation with the zero look-ahead setting

The zero look-ahead model is important for a real system, since many applications require the system give a quick response to a user's query. Thus, the first experiment compares the performance of RNN-T and T-T with zero look-ahead on the 1.8M testset. All models do not see any frame in the future and decode frame by frame. Table 1 presents the accuracy and runtime cost. We can observe that T-T and C-T significantly outperform RNN-T in terms of accuracy, and C-T is slightly better than T-T, which is consistent with previous literature. However, the RTF of T-T and C-T is much higher than 1 if full context is attended. When we truncate left history using the method proposed in section 3, we find that the model keeps almost the same performance, while reduces RTF significantly. It is also

	#hist #frames	WER (%)	RTF (#thread) 1	4
RNN-T	$+\infty$	9.86	1.56	0.46
T-T	$+\infty$	8.79	3.44	2.57
T-T	60	8.88	2.38	1.75
C-T	$+\infty$	8.78	4.02	2.56
C-T	60	8.80	2.41	1.83

Table 1: Model comparison for the zero look-ahead setting.

worth noting the truncated history reduced the memory consumption notably. The full context requires caching all keys and values in each frame and the memory grows linearly with the increase of audio frames, while truncated history keeps a fixed length of key and value vectors, such as 60 in this experiment. It demonstrates that the truncated history could reduce the runtime cost and memory effectively without affecting WER performance.

However, the RTF of T-T is still higher than 1 even if the model uses limited context in the zero look-ahead scenario. We find that the bottleneck of T-T is the encoder runtime cost, which occupies about 90% time in the whole inference stage. The frame-by-frame computation is time-consuming as it does not fully utilize the parallel computation of transformer models. Motivated by this, we introduce a trade-off between the latency and runtime cost by grouping several frames to form a batch for computation. For RNN-T, we could also form the batch and feed it to the LSTM based audio encoder. Due to the recurrent connection in LSTM, it can only partially parallelize the computation and the speedup is expected to be slower than Transformer. Table 2 reports RTF across different batch sizes. With a larger batch size, a faster decoding speed can be achieved. If we encode 2 frames (60ms latency) for each computation, RTF is less than 1. The RTF can be further reduced to 0.2 for T-T when the batch size is 15. However, 15 batch size introduces a latency of 450ms (15*30ms). Moreover, Table 2 indicates that the transformer is more suitable than RNN-T for batch operation, and an appropriate batch size makes T-T achieve similar RTF with RNN-T.

	#hist len	WER (%)	RTF (#batch size) 1	2	5	10	15
RNN-T	$+\infty$	9.86	0.46	0.31	0.26	0.21	0.20
T-T	60	8.88	1.75	0.69	0.38	0.26	0.19
C-T	60	8.80	1.83	0.95	0.48	0.36	0.25

Table 2: WER and RTF results of accumulating various number of frames for batch computation for RNN-T, T-T and C-T trained with zero lookahead. 4 threads are used for evaluation.

4.2.2. Evaluation with the small look-ahead setting

According to the zero look-ahead experiment, T-T has to trade latency for less computation cost. An absolute zero look-ahead is impossible for T-T, and it has to encode frames with a batch. Therefore, an idea is to do speech recognition (SR) with a small look-ahead, which enables the chunk-wise decoding in a natural way. When T-T and C-T take 24 frames lookahead, it generates a latency window between [0, 24] frames, and the averaged latency turns to be 24*30 ms /2=360ms, while RNN-T did not apply the chunk-based decoding and its latency is fixed as 12 frames (360ms). We also copy the number from [21] to show the performance hybrid system, streamable Transformer Seq2Seq, and offline Transformer Seq2Seq model on the test set, where the hybrid model is a highly optimized contextual layer trajectory LSTM (cltLSTM) [36], the streamable Transformer Seq2Seq is based on the chunk-wise trigger attention method, [14, 15], and the streamable RNN S2S is based on MoCha[13].

	#hist frame	#lookahead (ms)	WER (%)	RTF (#thread) 1	4
Hybrid	$+\infty$	480	9.34	-	-
RNN S2S	$+\infty$	720	9.61	-	-
Trans. S2S	$+\infty$	[480, 960]	9.16	-	-
Trans. S2S	$+\infty$	$+\infty$	7.82	-	-
RNN-T	$+\infty$	360	9.11	1.52	0.43
T-T	60	[0,720]	8.28	0.40	0.16
C-T	60	[0,720]	8.19	0.45	0.22
T-T	$+\infty$	$+\infty$	7.78	0.39	0.15
C-T	$+\infty$	$+\infty$	7.69	0.36	0.15

Table 3: WER and RTF comparison for different streaming models with lookahead. The first block results using Hybrid and S2S models are from [21].

Table 3 presents the performance of different models with a small lookahead. The results indicate that transducer models are more powerful than S2S in the streaming scenario, since RNN-T and T-T outperform streamable RNN S2S and streamable Transformer S2S respectively. T-T and C-T are the better choices for the scenario with a small look-ahead, as it shows strong accuracy while achieves an acceptable runtime cost. One unanticipated finding was that T-T with a small lookahead is very close to the performance of T-T using the entire utterance, which suggests that our simple strategy for T-T can avoid huge performance drop compared to the offline model.

4.2.3. 8-bit Optimization

The final experiment compared the effect of INT8 quantization. The INT8 quantization can reduce memory consumption and speed up inference effectively while keeping the performance. Table 4 shows the WER and speed results by using INT8 with 1 thread. INT8 presents 3.6 times speedup for RNN-T without affecting WER performance. In contrast, INT8 introduces slight WER degradation on T-T and C-T and yields about 2 times speedup. One possible explanation is that the softmax in Transformer layer is still operated in float precision, while softmax is computationally expensive on CPU.

	Precision	WER (%)	RTF
RNN-T	float32	9.11	1.56
	int8	9.13	0.43
T-T	float32	8.28	0.40
	int8	8.50	0.22
C-T	float32	8.19	0.45
	int8	8.40	0.26

Table 4: WER and speed results with INT8 quantization for Transducer models with lookahead.

5. CONCLUSION

We develop streaming T-T and C-T speech recognition model for real-time speech recognition, in the hope that the powerful Transformer encoder and streaming natural transducer architecture could take advantages from each other. We combine the idea of Transformer-XL and chunk-wise streaming processing to avoid latency grows linearly with the number of transformer layers. The experiment results show that T-T and C-T outperform hybrid model, RNN-T model, and streamable Transformer AED model in terms of accuracy in the streaming scenario. T-T and C-T can achieve comparable or better RTF compared to RNN-T given a small latency.

6. REFERENCES

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

[2] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Proc. Interspeech*, 2017, pp. 939–943.

[3] Eric Battenberg, Jitong Chen, et al., "Exploring neural transducers for end-to-end speech recognition," in *Proc. ASRU*, 2017, pp. 206–213.

[4] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017.

[5] Chung-Cheng Chiu, Sainath, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[6] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing acoustic-to-word CTC model," in *Proc. ICASSP*, 2018.

[7] Yanzhang He, Tara N Sainath, , et al., "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[8] Jinyu Li, , Rui Zhao, Zhong Meng, et al., "Developing RNN-T models surpassing high-performance hybrid models with customization capability," in *Proc. Interspeech*, 2020.

[9] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," *arXiv preprint arXiv:2010.10759*, 2020.

[10] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015.

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

[13] Chung-Cheng Chiu and Colin Raffel, "Monotonic chunkwise attention," in *Proc. ICLR*, 2018.

[14] Niko Moritz, Takaaki Hori, and Jonathan Le Roux, "Triggered attention for end-to-end speech recognition," in *ICASSP 2019*. 2019, pp. 5666–5670, IEEE.

[15] Chengyi Wang, Yu Wu, Shujie Liu, Jinyu Li, Liang Lu, Guoli Ye, and Ming Zhou, "Reducing the latency of end-to-end streaming speech recognition models with a scout network," in *Proc. Interspeech*, 2020.

[16] Mahaveer Jain, Kjell Schubert, Jay Mahadeokar, et al., "RNN-T for latency controlled ASR with improved beam search," *arXiv preprint arXiv:1911.01629*, 2019.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[19] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[20] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, et al., "A comparative study on transformer vs RNN in speech applications," in *Proc. ASRU*, 2019.

[21] Jinyu Li, Yu Wu, Yashesh Gaur, Chengyi Wang, Rui Zhao, and Shujie Liu, "On the comparison of popular end-to-end models for large scale speech recognition," in *Proc. Interspeech*, 2020.

[22] Qian Zhang, Han Lu, et al., "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[23] Anmol Gulati, James Qin, et al., "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[24] Niko Moritz, Takaaki Hori, and Jonathan Le Roux, "Streaming automatic speech recognition with the transformer model," in *ICASSP*, 2020.

[25] Jiahui Yu, Wei Han, et al., "Universal ASR: Unify and improve streaming ASR with full-context modeling," *arXiv preprint arXiv:2010.06030*, 2020.

[26] Anshuman Tripathi, Jaeyoung Kim, Qian Zhang, Han Lu, and Hasim Sak, "Transformer transducer: One model unifying streaming and non-streaming speech recognition," *arXiv preprint arXiv:2010.03192*, 2020.

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

[28] Chunyang Wu, Yongqiang Wang, Yangyang Shi, Ching-Feng Yeh, and Frank Zhang, "Streaming transformer-based acoustic models using self-attention with augmented memory," *arXiv preprint arXiv:2005.08042*, 2020.

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

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

[31] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. ACL*, 2019, pp. 2978–2988.

[32] Yongqiang Wang, Abdelrahman Mohamed, et al., "Transformer-based acoustic modeling for hybrid speech recognition," in *Proc. ICASSP*, 2020, pp. 6874–6878.

[33] Ching-Feng Yeh, Jay Mahadeokar, et al., "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[34] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani, "Self-attention with relative position representations," in *Proc. NAACL*, 2018.

[35] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass, "Highway long short-term memory rnns for distant speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 5755–5759.

[36] Jinyu Li, Liang Lu, Changliang Liu, and Yifan Gong, "Improving layer trajectory lstm with future context frames," in *Proc. ICASSP*. IEEE, 2019, pp. 6550–6554.