Low Latency End-to-End Streaming Speech Recognition with a Scout Network

Chengyi Wang[1], *Yu Wu*[2], *Liang Lu*[3], *Shujie Liu*[2], *Jinyu Li*[3], *Guoli Ye*[3], *Ming Zhou*[2]

[1] Nankai University, Tianjin, China
[2] Microsoft Research Asia, Beijing, China
[3]Microsoft Speech and Language Group, America

cywang@mail.nankai.edu.cn
{Wu.Yu, lial, shujliu, jinyli, guoye, mingzhou}@microsoft.com

Abstract

The attention-based Transformer model has achieved promising results for speech recognition (SR) in the offline mode. However, in the streaming mode, the Transformer model usually incurs significant latency to maintain its recognition accuracy when applying a fixed-length look-ahead window in each encoder layer. In this paper, we propose a novel low-latency streaming approach for Transformer models, which consists of a scout network and a recognition network. The scout network detects the whole word boundary without seeing any future frames, while the recognition network predicts the next subword by utilizing the information from all the frames before the predicted boundary. Our model achieves the best performance (2.7/6.4 WER) with only an average of 639 ms latency on the test-clean and test-other data sets of Librispeech.

Index Terms: online speech recognition, adaptive look-ahead, streaming model

1. Introduction

Recently, there has been a surge of end-to-end (E2E) automatic speech recognition (ASR) models, including the connectionist temporal classification (CTC) [1, 2, 3], the RNN-Transducer [4, 5, 6] and the attention-based encoder-decoder (AED) models [7, 8, 9, 10] due to their simple training procedure, desirable decoding efficiency and promising performance on large-scale speech benchmarks. In particular, the Transformer model [11] has been successfully applied to the E2E ASR, which enjoys faster training and better performance compared with RNNs. However, it is nontrivial to apply Transformer in an online recognition system due to the global encoder-decoder attention mechanism and the self-attention encoder, which requires the entire utterance. To solve the first problem, Transformer based monotonic chunkwise attention (MoChA) [12] and trigger attention mechanism (TA) [13] have been proposed to replace the global encoder-decoder attention. Regarding the streaming encoder, existing approaches can be categorized into look-ahead based method [13, 14] and chunk-based method [15, 12, 16, 17], as shown in Figure 1(a) and Figure 1(b). The former sets a look-ahead window for each frame to incorporate the necessary context information. However, latency will increase linearly with the number of layers. The chunk-wise approach segments the utterance into several fixed-length chunks and feeds them to the encoder one by one. There are always overlaps between chunks to improve the performance. However, this hurts the training parallelism of self-attention layers, and degrades the recognition accuracy when the chunk size is small.

In this paper, we propose an adaptive look-ahead approach to trade-off latency and WER, where the context window size is modified dynamically. We hypothesize that the most valuable contextual information to produce an output token is from the speech segment which corresponds to this word. Therefore, we introduce a neural component to detect the boundaries in the speech where a word starts and stops. We refer to this component as the scout network (SN), with a metaphor as a scout sent out ahead of the main force to gather the valuable information. Then the recognition network (RN) only looks ahead the frames up to the detected word boundaries. In this way, the recognition latency is not fixed but dependent on the duration of words and the segmentation performance of the SN.

To train the SN, we formulate it as a simple sequence labeling problem, where each frame is classified as either a boundary or not. We use the force-alignment results as ground truth to train the network. SN does not see any future information for the boundary detection. Hence, there is no additional latency overhead in the SN. After the word boundaries are detected, any end-to-end (E2E) model can be employed as the RN. In this paper, we use the TA based Transformer model [13] as the RN to conduct frame-synchronous one-pass decoding by looking ahead to the detected word boundaries.

Our experiments are conducted on Librispeech benchmark [18]. The results show that our proposed method can not only significantly reduce the latency, but also achieve the state-of-the-art recognition quality. Our base model with 78M parameters and large model with 138M parameters achieve 2.9/7.4 and 2.7/6.4 on test-clean and test-other datasets. To understand the effect of our proposed SN, we conduct experiments to analyze the relationship between the word error rate (WER) of the RN and the threshold in the SN. Experiments show that the higher threshold of the prediction precision (with lower recall of the boundaries and higher latency), leads to better recognition quality. The latency and the quality of the recognition can be balanced with the precision threshold of the SN.

2. Background

2.1. Transformer ASR

Transformer achieves promising results in ASR [19, 20]. Given a T-length speech feature sequence $\mathbf{X}$, the encoder transforms it to an intermediate representation $\mathbf{H}$, then the decoder predicts the following word y_i based on $\mathbf{H}$ and previous outputs $\mathbf{Y}_{[1:i-1]}$. The Transformer encoder consists of a convolution block and N_e encoder blocks, each of which has a multi-head self-attention layer and a feedforward layer. The decoder is composed of N_d decoder blocks, including a self-attention layer, an encoder-decoder attention layer and a feed-forward layer. Multihead attention mechanism is proposed, where in each head, weights are formed from queries ($\mathbf{Q} \in \mathcal{R}^d$) and

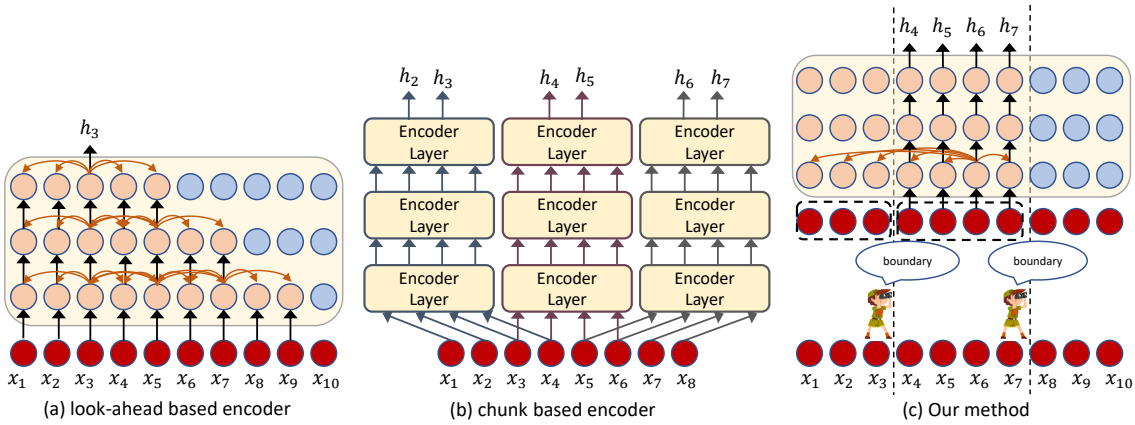

Figure 1: *A comparison between three Transformer based streaming models.*

keys ($\mathbf{K} \in \mathcal{R}^d$) and then applied to values ($\mathbf{V} \in \mathcal{R}^d$) as

$$\text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_{1:\text{m}})\mathbf{W}^{\text{head}} \quad (1)$$

$$\text{where head}_\text{i} = \text{softmax}(\frac{\mathbf{Q}\mathbf{W}_\mathbf{q}\mathbf{W}_\mathbf{k}^\mathbf{T}\mathbf{K}^\mathbf{T}}{\sqrt{\text{d}}})\mathbf{V}\mathbf{W}_\mathbf{v} \quad (2)$$

$\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ have the same dimension d. m is the number of attention heads. Residual connections and layer normalization are applied for each block.

2.2. Streaming ASR

To build a streaming AED based ASR system, both the encoder and the decoder are allowed to access limited future context. For the Transformer encoder, the self-attention mechanism provides a flexible way to control the range of context by masking the attention scores [13, 14]. However, the receptive field and latency will increase linearly with the number of layers. As shown in Figure 1(a), suppose the right look-ahead window size is w_r, the latency is $N_e \times w_r$ frames. Motivated by Transformer-XL [21], other works [12, 15, 16, 17, 22] use a chunk-wise approach, where the entire utterance is segmented into several fixed-length chunks as shown in Figure 1(b).

In terms of the decoder, the key is to learn the online monotonic alignments. [23, 24] restricts the range of the attention to a fixed-size window with position determined by previous attention distribution. Monotonic chunkwise attention (MoChA) [25] and its extension [17, 26] use a trainable energy function to shift the window. [22] applies it to multi-head encoder-decoder attention. [27] proposes trigger attention where a CTC module triggers the computation of the attention.

In this work, we use a Scout Network to detect word boundaries, enabling the encoder and the decoder to have an adaptive right context window. We believe the context window to the word boundary provides the most useful information for recognition and brings a better tradeoff between latency and accuracy.

3. Method

3.1. Scout Network

The purpose of the SN is to perform word boundary detection. In principle, SN can be implemented using any type of neural networks, while in this work, we use a similar architecture as the ASR encoder. It uses CNN as the pre-processing layer with a sampling rate r, followed by N_s self-attention layers. Given the acoustic input sequence $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, in each self-attention layer, the output of the current frame i is conditioned only on the previous states, so that there is no additional latency incurred in the SN. We denote the output hidden sequence as $\mathbf{H}^S = (\boldsymbol{h}_1^s, \ldots, \boldsymbol{h}_{T'}^s)$, where $T' = \lceil \frac{T}{r} \rceil$. Then a linear layer and a sigmoid layer predicts a boundary probability p_i. We train the SN by minimizing the cross-entropy loss as:

$$\mathcal{L}_{SN} = \sum_{T'} b_i \log(p_i) = \sum_{T'} b_i \log(\text{Sigmoid}(\mathbf{W}\boldsymbol{h}_i^s)) \quad (3)$$

where $b_i \in \{0, 1\}$ is the ground truth of the boundary decision. In this work, we used the Montreal Forced Aligner [28] to perform word-level force-alignment and obtained the labels. During inference, we set a threshold $\sigma \in (0, 1)$ to determine whether a frame is a boundary or not. Once $p_i \geq \sigma$, we denote the i-th frame as a boundary. σ is tuned on the validation set.

3.2. Streaming ASR with the Scout Network

3.2.1. Recognition Network Training

For the recognition network encoder, the CNN pre-processing block is the same as in the SN. Thus, RN and SN have the same downsampling rate. During training of RN, we sample $\tilde{b}$ according to the probability p produced by the SN as the predicted boundary. Suppose $\tilde{b}_{g_j} = 1$ is the j-th predicted boundary, we note g_j as the end time tag of the j-th word. Then we update the i-th hidden state in l-th self-attention layer as follows:

$$\tilde{h}_i^l = \text{Multihead}(h_i^{l-1}, \mathbf{H}_{[1:g_j]}^{l-1}, \mathbf{H}_{[1:g_j]}^{l-1}) \quad (4)$$

where $i \in (g_{j-1}, g_j)$. The i-th state can only access the states up to the time step g_j. $\mathbf{H}_{[1:g_j]}^{l-1}$ is regarded as key and value. It can also be seen as chunk-based encoder where the chunk size is adaptive and there is no overlap between chunks. During training, we rely on the mask strategy to avoid the model seeing context after the g_j-th frame.

We can use any type of the streaming decoder in our recognition network such as MoChA or Triggered Attention (TA) [27]. In this work, we used TA with an adaptive look-ahead window in the decoder. Once the CTC predicts a token y_k at frame i ($i \in (g_{j-1}, g_j)$), the decoder is triggered and the encoder-decoder attention is computed based on the segment of encoder output states $\mathbf{H}_{[1:g_j]}$. It should be noted that j may not equal to the ground truth boundary due to prediction error of SN and that one word may tokenized to several tokens. We

weighted average the CTC loss and the seq2seq loss to train our model as:

$$\mathcal{L} = -\gamma \log P_{s2s}(\mathbf{Y}|\mathbf{X}) - (1-\gamma)\log P_{ctc}(\mathbf{Y}|\mathbf{X}). \quad (5)$$

$$\text{where } P_{s2s}(\mathbf{Y}|\mathbf{X}) = \sum_{y_j \in \mathbf{Y}} P_{s2s}(y_j|\mathbf{Y}_{[1:j-1]}, \mathbf{H}_{[1:g_j]}) \quad (6)$$

We set γ as 0.7 in this work. The model is initialized from an offline Transformer and fine-tuned in a streaming manner.

To train the decoder, the alignment between CTC paths and the label sequence $\mathbf{Y}$ is required. Different from [13] which performs Viterbi alignment during training, we used the path with the highest Viterbi alignment score generated by the pre-trained offline model to trigger the decoder.

3.2.2. Decoding

Algorithm 1 Streaming Transformer Decoding with Scout Network

1: **procedure** SCOUT-THEN-DECODE($\mathbf{X}$, $K, \sigma, \sigma_0, \lambda, \alpha, \beta$)
2: $k \leftarrow 0$, $g_k \leftarrow 0$
3: $\ell \leftarrow (\langle sos \rangle,)$, $\Omega \leftarrow \{l\}$
4: **for** i = 0 to T **do**
5: $p_{i'} \leftarrow \text{SN}(x_i)$, $i' \leftarrow \frac{i}{r}$ ▷ Scout Boundary
6: **if** $p_{i'} > \sigma$ **then**
7: $k \leftarrow k+1$, $g_k \leftarrow i'$
8: $\mathbf{H}_{[g_{k-1}+1:g_k]} \leftarrow \textbf{ENC}(\mathbf{X}_{[1:i]})$
9: $\Omega, p_{\text{joint}} \leftarrow$DECODE($\Omega$, $\mathbf{H}_{[1:g_k]}$, $g_{k-1}+1$, g_k, K, σ_0, λ, α, β)
10: **return** MAX($\Omega, p_{\text{joint}}, 1$)
11: **procedure** DECODE(Ω, $\mathbf{H}$, start, end, K, σ_0, λ, α, β)
12: $\hat{\Omega}_{\text{ta}} \leftarrow \varnothing$
13: **for** j = start to end **do**
14: $\Omega_{\text{ctc}}, p_{\text{ctc}} \leftarrow$ CTCPREFIX($\Omega, \sigma_0, \boldsymbol{h}_j$)
15: **for** ℓ in Ω_{ctc} **do**
16: **if** ℓ not in $\hat{\Omega}_{\text{ta}}$ **then**
17: $p_{\text{ta}}(\ell) \leftarrow P_{s2s}(\ell|\mathbf{H})$
18: add ℓ to $\hat{\Omega}_{\text{ta}}$
19: $p_{\text{local}}(\ell) \leftarrow \log p_{\text{ctc}} + \alpha \log p_{\text{LM}} + \beta|\ell|$
20: $p_{\text{joint}}(\ell) \leftarrow \lambda \log p_{\text{ctc}} + (1-\lambda)\log p_{\text{ta}} + \alpha \log p_{\text{LM}} + \beta|\ell|$
21: $\Omega_{\text{local}} \leftarrow \text{MAX}(\Omega_{\text{ctc}}, p_{\text{local}}, K)$ ▷ Beam Pruning
22: $\Omega_{\text{ta}} \leftarrow \text{MAX}(\Omega_{\text{ctc}}, p_{\text{ta}}, K)$
23: $\Omega_{\text{joint}} \leftarrow \text{MAX}(\Omega_{\text{ctc}}, p_{\text{joint}}, K)$
24: $\Omega \leftarrow \Omega_{\text{local}} \cup \Omega_{\text{ta}} \cup \Omega_{\text{joint}}$
25: **return** Ω, p_{joint}

Algorithm 1 gives the decoding procedure for streaming Transformer with the Scout Network. The hyper-parameters used by the function include beam width K, boundary decision threshold σ, CTC decoding threshold σ_0, CTC decoding weight λ, language model weight α and length penalty β. In the 2nd line of the algorithm, we first initialize the index of speech segment $k = 0$ and the end boundary of the k-th segment g_k. The hypothesis set Ω is initialized in line 3 with the prefix sequence ℓ containing only the start of the sequence label $\langle \text{sos} \rangle$. In line 4-6, the frame-level feature is fed into the Scout Network and an instantaneous decision is made by threshold σ. Once a word boundary is detected, the ASR decoding is triggered at line 8-9.

The DECODE function shown from line 11 to 25 is similar to the decoding scheme in [13] with a small modification in beam pruning. In line 14, we perform the CTC prefix beam search [29] based on the current encoder state $\boldsymbol{h}_j$ and prefix set Ω, generating candidates set Ω_{ctc}. Then the trigger attention decoder scores every candidate. To avoid duplicated computation, we store all the scored candidates in $\hat{\Omega}_{\text{ta}}$. In line 19 to 20, a local score and a joint score are assigned to each candidate. Then we select the top-K candidates based on local score, attention score and joint score respectively, and combine them as the hypothesis set for the next step. When the process finishes, we select the prefix with the highest joint score as the decoding output.

Table 1: *An example of the accuracy evaluation for the Scout Network. E' and E are downsampled positions.*

Y	it gave an imposing appearance to most of the wholesale houses
E'	7 12 14 28 42 45 52 54 56 ** 68
E	7 12 15 28 42 45 52 ** 56 61 68
evaluate	sub del ins
word latency (ms)	30 30 70 30 30 30 30 110 30 – 30

4. Experiments

4.1. Setup

We performed our experiments on the LibriSpeech dataset [18], which contains 960 hours of audio in the training set. We use dev-clean as the validation set and report results on the test-clean and test-other sets.

Our approach is implemented based on ESPnet [19]. We used 80-dim log Mel-filter bank features with 3-dim pitch features [30] with 10ms sampling rate. The text is tokenized using SentencePiece [31] and we set the vocabulary size to 5000. We evaluated our method in two settings, the Base setting and the Large setting. For the Base setting, we used the same architecture as [13] with $d_{model} = 512, d_{ff} = 2048, d_h = 4, N_e = 12$, and $N_d = 6$ for a fair comparison with previous works. We used the released Transformer model provided by ESPnet as the pretrained offline model. For the Large setting, we used the architecture described in our previous work [32] with $N_e = 24$ and $N_d = 12$. For both settings, the down-sampling rate r is 4, so that the self-attention layers operate at the sampling rate of 40ms per frame. We updated the SN using the Adam optimizer with a learning rate of 0.001. The RN was trained using a warmup step of 2500 and a learning rate coefficient of 1.0 following the schedule in [11]. SpecAugment [33] is applied following the recipe in ESPnet. We run the fine-tuning stage for about 20 epochs. For decoding, we averaged the last 5 checkpoints as the final model. For decoding hyper-parameters, we set $K = 10, \sigma_0 = 0.0005, \lambda = 0.5, \alpha = 0.5, \beta = 2.0$ as in Algorithm 1. The N-best was re-scored using an LSTM language model provided by ESPnet.

4.2. Scout Network Evaluation

We first evaluate the accuracy of the Scout Network for word boundary prediction. Given a predicted boundary E and a reference boundary E' from the force-alignment, we use the edit distance as the metric, which can be decomposed into substitution (sub), deletion (del), insertion (ins) rates. An example is given in Table 1. In Table 2, we show the validation results of different number of layers N_s and threshold σ. We observe a high precision but a low recall with a high threshold such as $\sigma = 0.9$. And not surprisingly, a shallower Transformer degrades the accuracy of the SN slightly. To draw the connection between the

Table 2: *Segmentation Evaluation on the dev clean dataset. Precision can be estimated as (1.0 - sub - ins) and recall can be estimated as (1.0 - sub - del) The first two columns are averaged per frame time cost.*

N_s	GPU	CPU	σ	sub	del	ins
12	11.1ms	29.8ms	0.5	0.13	0.072	0.019
			0.7	0.08	0.21	0.006
			0.9	0.003	0.426	0.001
8	7.4ms	19.7ms	0.9	0.025	0.514	0.001
4	3.7ms	10.6ms	0.9	0.024	0.562	0.001

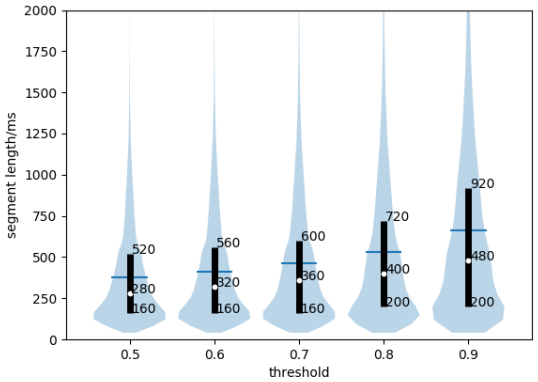

Figure 2: *The distribution of segment length for different thresholds. The blue area shows the density. The deep blue lines are the mean values. The white markers represent the median values and the black vertical lines is the interquartile ranges.*

threshold and the latency of the recognition network, we further evaluate the distribution of the segment lengths on the joint test sets, which is shown in Figure 2, where we used the base SN with $N_s = 12$. For each segment, the length is computed as $(g_j - g_{j-1}) \times 40$ms, where g_j is the predicted end boundary. As shown in the figure, even with a threshold of 0.9, there are still 50% segments shorter than 480ms and 75% segments shorter than 920ms. Though some segments have a length longer than 2s, we can apply a threshold to control the maximum segment length. We also show the computation time for a Scout Network to process each frame when running on a P40 GPU and 24 cores Intel(R) 2.60GHz CPU in Table 2. Since the duration of each frame is 40ms, the Scout Networks can run faster than real-time on either GPU or CPU in our experiments.

4.3. Recognition Results

To measure latency, we define two metrics without considering the computation time, namely the frame-level latency and the word-level latency. The frame-level latency is defined as how many future frames are consumed for prediction at each timestep i, which is same as that in [13]. For our model, it is computed as $(g_j - i) \times 40$ms + 30ms (from CNNs), where $g_{j-1} < i \leq g_j$. Word-level latency is defined as the time difference between the actual end boundary and the end boundary produced by SN. For words whose boundaries are predicted correctly, their word-level latency is 30ms. For words whose boundaries are missed or predicted after the reference boundary, we compute the distance between the next boundary g_j and the real boundary g'_j as $(g'_j - g_j) \times 40$ms + 30ms.[1] An example

[1]The latency is computed in word-level. As the SN is latency-free, the reported result is total latency of our system.

Table 3: *The comparison with literature baselines and reimplemented baselines. The offline AED models adopt hybrid CTC/Attention decoding algorithm. ∞ denotes the whole utterance. We reproduce the TA baselines with our decoding algorithm, marked with *.*

Models	WER		latency(ms)	
	clean	other	frame	word
Base setting (∼78M)				
Contextual [22]	4.6	13.1	∞	
TA (78M) [13]	2.7	6.1	∞	
TA-1	3.2/3.4*	8.2/9.5*	750	
TA-2	2.9/3.0*	7.8/8.5*	1230	
TA-4	2.8	7.3	2190	
Our method offline	2.7	5.9	∞	
+SN-12 σ=0.5	3.5	9.4	317	63
+SN-12 σ=0.7	3.3	8.5	404	133
+SN-12 σ=0.9	2.9	7.4	619	338
+SN-4 σ=0.9	2.8	7.1	844	546
+golden	2.1	5.3	299	30
Large setting (∼140M)				
Transducer [14]	2.4	5.6	∞	
Transducer-2	3.0	7.7	1080	
Transducer-6	2.8	6.9	3240	
Our method offline	2.2	5.2	∞	
+SN-12 σ=0.9	2.7	6.4	639	352

is shown in Table 1. We report the average latency.

We compare our method with the approaches from the literature in Table 3. Contextual Block [22] system uses a chunk-based encoder with a contextual embedding to utilize global information and a decoder with global attention. The TA system [13] adopts a look ahead based encoder and a trigger-attention based decoder. In Large setting, [14] proposes Transformer Transducer with look-ahead based speech and label encoders.

In the base setting, we present model performance across different thresholds of the SN. Compared with TA baselines, our model with SN-12 and thresholds 0.9 can achieve a similar WER with their best model while reducing the latency from 2190ms to 619ms. Model with SN-4 achieves better WER but sacrificing latency. We can see that the higher segmentation threshold (less substitution and insertion segmentation errors), lower the WER. Tuning the threshold can give us more room for potential WER and latency tradeoffs. We also show the results when the golden segmentation is obtained. The result is surprising as it even outperforms the offline model. In the Large setting, our model has lower WER and lower latency compared with Transformer Transducer [14], which to our knowledge is the state-of-the-art results for streaming E2E ASR system on the LibriSpeech benchmark.

5. Conclusion

We propose a new strategy for E2E speech recognition model where a scout network detects the current word boundary and then a recognition network conducts frame-synchronous one-pass decoding by looking ahead to the predicted boundary. Our method produces a good tradeoff between WER and latency. It achieves 2.7% and 6.4% WER score on test-clean and test-other sets with an average of 640ms frame latency and 352ms word latency, which to our knowledge is the best publisted results for E2E streaming ASR model on LibriSpeech benchmark.

6. References

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

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

[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 *ICML 2016*, 2016, pp. 173–182.

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

[5] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP 2013*. IEEE, 2013, pp. 6645–6649.

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

[7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NeurIPS 2015*, 2015, pp. 577–585.

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

[9] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP 2018*. IEEE, 2018, pp. 4774–4778.

[10] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS 2017*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008.

[12] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Towards online end-to-end transformer automatic speech recognition," *CoRR*, vol. abs/1910.11871, 2019.

[13] N. Moritz, T. Hori, and J. L. Roux, "Streaming automatic speech recognition with the transformer model," *CoRR*, vol. abs/2001.02674, 2020.

[14] 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," *arXiv preprint arXiv:2002.02562*, 2020.

[15] L. Dong, F. Wang, and B. Xu, "Self-attention aligner: A latency-control end-to-end model for ASR using self-attention network and chunk-hopping," in *ICASSP 2019*. IEEE, 2019, pp. 5656–5660.

[16] Z. Tian, J. Yi, Y. Bai, J. Tao, S. Zhang, and Z. Wen, "Synchronous transformers for end-to-end speech recognition," *CoRR*, vol. abs/1912.02958, 2019.

[17] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online ctc/attention end-to-end speech recognition architecture," *CoRR*, vol. abs/2001.08290, 2020.

[18] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP 2015*, 2015, pp. 5206–5210.

[19] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, "A comparative study on transformer vs RNN in speech applications," *CoRR*, vol. abs/1909.06317, 2019.

[20] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," *arXiv preprint arXiv:1910.09799*, 2019.

[21] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *ACL 2019*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 2978–2988.

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

[23] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP 2016*. IEEE, 2016, pp. 4945–4949.

[24] W. Chan and I. Lane, "On online attention-based speech recognition and joint mandarin character-pinyin training," in *Interspeech 2016*, N. Morgan, Ed. ISCA, 2016, pp. 3404–3408.

[25] C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *ICLR 2018*. OpenReview.net, 2018.

[26] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *arXiv preprint arXiv:1811.05247*, 2018.

[27] N. Moritz, T. Hori, and J. L. Roux, "Triggered attention for end-to-end speech recognition," in *ICASSP 2019*. IEEE, 2019, pp. 5666–5670.

[28] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.

[29] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *arXiv preprint arXiv:1408.2873*, 2014.

[30] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *ICASSP 2014*, 2014, pp. 2494–2498.

[31] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[32] C. Wang, Y. Wu, Y. Du, J. Li, S. Liu, L. Lu, S. Ren, G. Ye, S. Zhao, and M. Zhou, "Semantic mask for transformer based end-to-end speech recognition," *arXiv preprint arXiv:1912.03010*, 2019.

[33] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680