High Performance Sequence-to-Sequence Model for Streaming Speech Recognition

Thai-Son Nguyen, Ngoc-Quan Pham, Sebastian Stüker, Alex Waibel

Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology

thai.nguyen@kit.edu

Abstract

Recently sequence-to-sequence models have started to achieve state-of-the-art performance on standard speech recognition tasks when processing audio data in batch mode, i.e., the complete audio data is available when starting processing. However, when it comes to performing run-on recognition on an input stream of audio data while producing recognition results in real-time and with low word-based latency, these models face several challenges. For many techniques, the whole audio sequence to be decoded needs to be available at the start of the processing, e.g., for the attention mechanism or the bidirectional LSTM (BLSTM). In this paper, we propose several techniques to mitigate these problems. We introduce an additional loss function controlling the uncertainty of the attention mechanism, a modified beam search identifying partial, stable hypotheses, ways of working with BLSTM in the encoder, and the use of chunked BLSTM. Our experiments show that with the right combination of these techniques, it is possible to perform run-on speech recognition with low word-based latency without sacrificing in word error rate performance.

Index Terms: end-to-end, sequence-to-sequence, online, streaming

1. Introduction

Sequence-to-sequence (S2S) attention-based models [1, 2] have become increasingly popular for end-to-end speech recognition. Several advances [3, 4, 5, 6] have been proposed to the architecture and the optimization of S2S models to achieve superior recognition performance. In offline scenarios, i.e., batch processing of audio files, the S2S models in [7, 8] have already shown state-of-the-art performance on standard benchmarks. However, methods for employing S2S models in online speech recognition, i.e., run-on recognition with low latency, still needs to be researched, to obtain the desired accuracy and latency.

[9, 10, 11] pointed out early that the shortcoming of an attention-based S2S model used in online condition lies in its attention mechanism, which must perform a pass over the entire input sequence for every element of the output sequence. [10, 11] proposed a so-called monotonic attention mechanism that enforces a monotonic alignment between the input and output sequence. Later on, [12, 13, 14] have addressed the latency issue of bidirectional encoders, which is also an obstacle for online speech recognition. In these studies, unidirectional and chunk-based encoder architectures replace the fully-bidirectional approach to control the latency.

In this work, we analyze the alignment behavior of the attention function of a high-performance S2S model and propose an additional constraint loss to make it capable of streaming inference. By discussing the problems that occurred when adapting a S2S model to be used for a streaming recognizer, we additionally show that the standard beam-search has no guarantee for low-latency inference results, and needs to be modified for providing partial hypotheses. Besides, we argue that the common real-time factor is not a proper choice for measuring the user-perceived latency in online and streaming setup, and propose a novel and suitable technique for the replacement.

In contrast to the earlier works in the literature, our experimental results proved that a bidirectional encoder could be combined with suitable inference methods to produce high accuracy and low latency speech recognition output. With a delay of 1.5 seconds in all output elements, our streaming recognizer can achieve the ideal performance of an offline system with the same configuration. To the best of our knowledge for the first time, a S2S speech recognition model can be used in online conditions without scarifying accuracy.

2. Sequence-to-Sequence Model

In this work, we modify the LSTM-based sequence-to-sequence encoder-decoder model proposed in [8] to perform high-accuracy online streaming ASR with very low latency. Our model can be decomposed using a set of neural network functions as follows:

$$
\begin{aligned}
enc &= LSTM(CNN(spectrogram)) \\
emb &= LSTM(Embedding(subwords)) \\
ctx, attn &= SoftAttention(emb, enc, enc) \\
y &= Distribution(ctx + emb)
\end{aligned}
$$

In principle, the functions are designed to map a sequence of acoustic features into a sequence of sub-words and can be grouped into two parts: encoder and decoder. In the encoder, acoustic vectors are down-sampled with two convolutional layers and then fed into several bidirectional LSTM layers to generate the encoder's hidden states enc. In the decoder, two unidirectional LSTM layers are used to embed a sub-word unit into a latent representation emb. The multi-head *soft-attention* function proposed in [15] is used to model the relationship between enc and emb, which results in a context vector ctx. All the functions are jointly trained via the sequence cross-entropy loss by plugging a softmax distribution on top of ctx and emb.

As shown in [8], this S2S model can achieve highly-competitive offline performance on the Switchboard speech recognition task. However, the model encounters latency issues when being used in online conditions since both, the attention function and the bidirectional encoder network, require the entire input sequence to achieve their optimal performance.

3. Streaming S2S ASR

In this section, we describe our modifications that enable the S2S model to perform online streaming speech recognition with low latency and without loss in performance. The modifications

include an additional loss to control the uncertainty of the attention function and search algorithms to infer high-accuracy partial hypothesis.

3.1. Discouraging Look-ahead Attention

The core of S2S models is the mechanism that autoregessively generates a context vector *ctx* for the prediction of the next token. For the model described in Section 2, *ctx* is computed as a sum of all the encoder's hidden states weighted by the *attention scores* which are calculated by the attention function. The attention scores calculated for a specific token typically reveals the positions within the encoder states (or spectral frames) corresponding to the token. So, the attention function can be considered as an *alignment model*. However, this unsupervised alignment does not resemble traditional forced-alignments (or human alignments) in speech recognition. As illustrated in Figure 1a, during a particular inference process, the attention scores produced for many tokens (e.g., #3, 8, 9, 16) are dominated by the start and end frames, which are not the proper alignments. In this case, the inference still produces the correct transcript, and so the attention function works as it is expected. The mismatch between the attention-based alignment and regular alignment reveals uncertainty that the attention function may have while being optimized with the sequence training likelihood. Although this uncertainty may not lead to inference errors, the attention function always employs all the encoder's hidden states, which hinders the model from being used in streaming inference. It is preferable for a streaming inference that for the prediction of a token S, the attention function only considers past frames until a particular time S_t (the endpoint) and disregards all future frames.

To build such a S2S model for streaming, we investigated the incorporation of an additional loss which discourages the attention function from using future frames during training. Specifically, given token S which belongs to word W in label sequence L, we find a region $R_s = (W_t, \infty)$ in which W_t is the end time of W provided by a Viterbi alignment. The attention-based constraint loss is computed as the sum of all attention scores within the region R_s for all S in L:

$$\mathcal{L}_{attn} = \alpha \sum_{S} \sum_{x}^{|R_s|} Ascore_S^x$$

The tuneable parameter α adjusts the influence of the constraint loss to the maximum likelihood loss of the label sequence during training. By minimizing both losses simultaneously, we expect that the attention function learns to produce *close-to-zero* scores for the constraint regions for all label tokens while still minimizing the main loss.

3.2. Inference for Partial Stable Hypothesis

Beam search is the most efficient approach for the inference of S2S models. Its basic idea is to maintain a search network in which network paths are extended with new nodes with the highest accumulated scores and then pruned away to keep only a set of active paths (or hypotheses). Typically, the most probable hypothesis for an utterance X is found and guaranteed when the search space constructed from the entire acoustic signals of X is supplied to the search. However, waiting for the complete acoustic signals of X to output inference results is not efficient for a streaming setup. A streaming recognizer must be able to produce partial output while processing partial input. In this section, we describe our search algorithm applied to the proposed S2S model to produce partial output while retaining high accuracy.

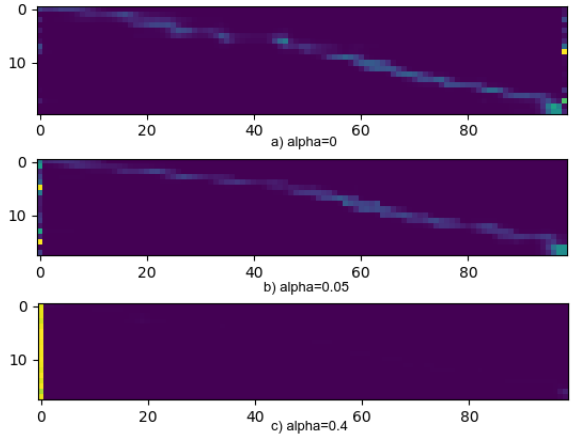

Figure 1: *Attention-based alignments provided by a) the regular attention function and b) c) the attention function trained with the constraint loss during the inference of an utterance of 4-seconds length (down-sampling of 4 frames after encoder's layers). The alignments for the tokens 3, 8, 9, 16 are dominated by both start and end frames in a), and dominated by start frames only in b).*

Assume that in a streaming setup, at time t we use the proposed S2S model to perform inference for t audio frames. Given a context sequence C, the attention function is used to generate t attention scores for the prediction of the next token. We find a time $t_c <= t$ such that the sum of all attention scores from the *covering* window $w = [0, t_c]$ is equal to a constant $\theta = \sum_x^{|t_c|} Ascore^x$. When $\theta = 0.95$, w covers all dominant attention scores and the context vector generated from w is almost the same as from $[0, t]$. If t_c is observed to be unchanged when t keeps growing, then we consider t_c as the endpoint of C. During stream processing, we use a term Δ to determine if endpoint t_c finally gets fixed as $t_c < t - \Delta$.

We then incorporate the information of endpoints into the beam search to find a partial stable hypothesis. Assume that our beam search can always perform in real-time for t audio frames to produce N considered hypotheses. If all N hypotheses share the same prefix sequence C and the endpoint of C is determined, then we consider C to be an *immortal* part that will not change anymore in the future. When more audio frames are available in the stream, C will be used as the prefix for all search hypotheses, and we repeat this step to find a longer stable hypothesis. Except the condition on endpoints, the idea of finding *immortal prefix* is similar to the partial trace-back [16, 17] used in HMM-based speech recognizers.

In addition to the immortal prefix, we also investigated a more straightforward method in which we only consider the *best-ranked* hypothesis and decide on a stable part C based solely on the term Δ. The inspiration comes from the incremental speech recognition approach proposed in [18].

3.3. Bidirectional Encoder

To achieve high performance, bidirectional LSTM have been the optimal choice for the encoder of LSTM-based S2S models. However, due to the backward LSTM, bidirectional LSTM are *not* suited to provide partial and low-latency output as needed for streaming recognizers. The addition of acoustic input will affect all of the encoder's hidden states, which then makes all

Table 1: *WER performance of the S2S model with bidirectional encoder trained with different scales of the constraint loss.*

Model	α	SWB	CH	Hub5'00
6x1024 BLSTM		5.9	11.8	8.9
	0.4	6.2	12.2	9.2
	0.2	6.1	12.1	9.1
	0.05	5.8	12.0	8.9

partial inference results unstable. This effect leads to the fact that stable output can be confidently inferred only when the input is complete. Therefore, earlier works [10, 19, 20] switched to unidirectional LSTM in their online models.

In this work, we try to utilize bidirectional LSTM for high-performance speech recognition in a streaming scenario. In the first setting, we investigated the use of the S2S model with a fully bidirectional encoder. First, we train the S2S model to achieve optimal parameters for the offline setup, and with the attention-based constraint loss proposed in Section 3.1. Then, during inference, we update the encoder's hidden states from all available acoustic input before performing the search approaches in Section 3.2 to find stable hypotheses. As will be shown later, the use of a bidirectional LSTM as this way is possible since the proposed inference methods rely on the determination of endpoints, and the update of encoder's hidden states leads to stabilizing this determination.

In addition to fully bidirectional LSTM, we also experimented with a chunk-based BLSTM approach. During training, we divide input sequences into many non-overlapping blocks of a fixed size of K, and then use a BLSTM to compute each block sequentially. To benefit from long-range contextual learning, we initialize the forward LSTM with its last hidden states after processing the previous chunk. The initialization of the backward LSTM can either be a constant or from the previous chunk. By doing so, the encoder's hidden states can be computed incrementally and efficiently as for unidirectional LSTM. This chunk-based approach is different from [21] and the latency-controlled BLSTM [22, 23] that adopt constant initialization of both directions.

4. Experiments

4.1. Experimental Setup

Our experiments were conducted on the Fisher+Switchboard corpus consisting of 2,000 hours of telephone conversation speech. The Hub5'00 evaluation data was used as the test set. All the experimental models use the same input features of 40 dimensional log-mel filterbanks to predict 4,000 BPE sub-word units generated with the SentencePiece [24] toolkit from all the training transcripts. The models with bidirectional encoder employ six layers of 1024 units while it is 1536 for the unidirectional encoders. We used only 1-head for the attention function in all setups. All models were trained with a dropout of 0.3. We further used the combination of two data augmentation methods *Dynamic Time Stretching* and *SpecAugment* proposed in [8] to reduce model overfitting. We use Adam [25] with an adaptive learning rate schedule to perform 12,000 updates during training. The model parameters of the 5 best epochs according to the perplexity on the cross-validation set are averaged to produce the final model.

For beam search, we use neither length normalization nor any language model. With a beam size of 8, the experimental models typically achieve their optimal accuracy.

Table 2: *Latency and accuracy of the S2S model with bidirectional encoder on Hub5'00 test set.*

Method	Beam Size	Δ	WER	Latency
Force-Alignment	8		8.9	0.60
	4		9.1	0.60
	2		9.3	0.60
Immortal Prefix	8	20	**8.9**	**0.93**
	8	30	8.9	0.93
	4	20	9.2	0.86
	4	30	**9.1**	**0.87**
	2	20	12.6	0.74
	2	40	10.1	0.79
	2	60	9.5	0.83
	2	80	9.3	0.86
1st-Ranked Prefix	8	30	11.2	0.75
	8	50	9.6	0.80
	8	70	9.3	0.84
	4	30	11.3	0.75
	4	50	9.6	0.80
	4	70	9.3	0.84
	2	10	25.8	0.62
	2	30	11.4	0.75
	2	50	9.7	0.80
	2	70	9.3	0.84
Combination	8	20-70	9.2	0.83
	4	30-70	**9.4**	**0.81**
	2	60-70	9.5	0.83

4.2. Latency Measure

Neither the commonly used *real-time factor* (RTF) nor commitment latency [26, 27] are sufficient to measure user-perceived latency for a streaming recognizer. For example, the transcript outputs for an 11-seconds sentence can appear 10 seconds later than a 1-second sentence, but the RTFs measured in two cases can be similar. In this study, we propose to use a different method for measuring streaming latency. Assume that a recognizer processes a sentence S of T seconds in streaming fashion and it outputs N token s_1, s_2,.. s_n at different timestamps t_1, t_2,.. t_n. And assume the inference time is always a small constant, then timestamp t_i is just when the recognizer is confident of producing s_i. The latency of recognizing S with regard to the transcript s_1, s_2,.. s_n is calculated as the average of all token latencies $t_i \in [1, n]$ normalized by the duration of S: $\sum t_i/(n * T)$. With this measure, the latency of an offline system is always 1 – as the offline system is only confident for all transcripts until end-of-sentence. In the same way, we simulate the latency of an *instant* recognizer by using a forced-alignment to find t_i for s_i.

5. Results

5.1. Effect of the Constraint Loss

In this section, we evaluate the influence of the constraint loss proposed in Section 3.1 on the training of the S2S model. We started by using a high value for α and exponentially decreased it to train several systems for comparison. As observed during training, the constraint loss gets small quickly to a stable value depended on α. Joint training slows down the convergence of the main loss but does not have a significant impact on the final performance. As shown in Table 1, WERs are slightly worse with high α and can be similar to the regular training when α is

small (e.g., 0.05). Different from that, the constraint loss may largely change the behavior of the attention function. For example, in Figure 1b, the attention function moves the high scores of the mismatched alignment to start frames, instead of end frames as in the regular training. We also found an extreme case when $\alpha = 0.4$. The attention-based alignment does not correspond at all to the proper alignment as illustrated Figure 1c.

Using the model trained with $\alpha = 0.05$, we follow the approach in Section 3.2 to extract the endpoints for all prefixes found during the inference of the evaluation set. We could verify that the extracted endpoints in all sentences match the expectation for streaming inference described in Section 3.1. So we keep this model for further experiments.

5.2. Latency on Various Conditions

Using the S2S model with a bidirectional encoder trained with the constraint loss scale $\alpha = 0.05$, we performed several experiments with the inference approaches described in Section 3.2. In the experiments, the streaming scenario is simulated by repeatedly feeding an additional audio chunk of 250 ms to the experimental systems for incremental inferences. All the inferences were performed on a single Nvidia Titan RTX GPU, which produced an average RTF of 0.065 with a beam size of 8. The RTF result shows that real-time capacity is not a bottleneck problem in this setup. So we focus on the latency measure proposed in Section 4.2.

For baselines, we computed the offline WER performance with beam sizes 8, 4, 2, and then used a force-alignment system to produce the *ideal* latency from the offline transcripts. The ideal latency is always 0.6. If we shift the time alignment of the transcripts with 250 ms (i.e., all the outputs have a delay of 250 ms), 500 ms, 1 second, and 1.5 seconds, then we obtained a latency of 0.71, 0.78, 0.86 and 0.91 respectively.

Table 2 presents the accuracy and the latency we achieved when using the *immortal prefix* and *1st-ranked prefix* inference methods with several settings of Δ. Overall, the two methods are consistent with the observations in the HMM-based systems [16, 17, 18]. Using the *immortal prefix* condition, the final accuracy can be guaranteed as for the offline inference for large beam sizes, e.g., 8 and 4. For a smaller beam size, this condition is not strong enough to deal with unstable partial results – probably due to the changes of the encoder's hidden states. In the *1st-ranked prefix* approach, increasing Δ allows for a flexible trade-off between the accuracy and the latency. The offline accuracy can also be achieved if a very large Δ is applied. These results consolidate our findings in two aspects. First, the integration of Δ is reliable and crucial for the streaming inferences to work efficiently. And second, the use of the bidirectional LSTM for the encoder is possible and results in high accuracy.

To achieve 8.9% WER (the offline accuracy), the system needs to delay outputs with an average duration of about 1.5 seconds. To obtain a lower latency of 1 second, the WER increases to 9.2%, e.g., by using the *immortal prefix* method with $\Delta = 20$ and $beamsize = 4$. The *combination* of both methods is efficient if we want to reach a latency of 0.81, which is closer to the average delay of 0.5 seconds.

5.3. Performance of Different Encoders

The bidirectional requires additional re-computation of the entire encoder's hidden states for every addition of input signal in the stream. In this section, we investigate two additional network architectures, *unidirectional* LSTM and *chunk-based* BLSTM described in Section 3.3, that improve the computational efficiency of the encoder. For chunk-based, we experimented with $K = 80$ and $K = 200$, as the chunk sizes of 800 ms and 2 seconds. We constantly found that initializing the backward LSTM from the last hidden states of the previous chunk is better than a constant, so we only present the results of this approach. We evaluated two types of encoders in two categories: the best accuracy and the accuracy that the systems retain when maintaining an average delay of 1 second. To do so, we use the same *immortal prefix* inference and experiment with different settings of beam size and Δ.

Table 3: *Latency and accuracy of the S2S models with unidirectional and chunk-based encoders using immortal prefix.*

Encoder	Beam Size	Δ	WER	Latency
Unidirectional	8	30	12.7	0.94
	8	∞	12.6	1.00
	2	20	13.6	0.82
	2	30	13.2	0.85
	2	∞	13.1	1.00
Chunk-based *K=80*	8	30	10.5	0.91
	8	∞	10.4	1.00
	2	20	11.1	0.80
	2	30	10.9	0.82
	2	∞	10.8	1.00
Chunk-based *K=200*	8	30	10.3	0.89
	8	∞	10.0	1.00
	2	30	11.3	0.79
	2	60	10.8	0.85
	2	∞	10.7	1.00

As shown in Table 3, there is a big gap between the best WER of the unidirectional and bidirectional encoders (12.6% vs. 8.9%). The chunk-based encoder, however, is closer to the performance of the bidirectional one when a large chunk size is used. As the encoder's states are fixed early, the inferences are already stable when $\Delta = 30$ for all beam sizes. To achieve 1-second delay, all the approaches need to trade-off for an accuracy reduction of 5% relatively. In term of latency, the chunk-based approach with $K = 80$ and $beamsize = 2$ and $\Delta = 30$ is the best setting in this setup.

6. Related work

[10, 11] pointed out the problems of the *soft-attention* mechanism on acquiring the entire encoder's states and proposed a trainable *monotonic* attention function to train sequence-to-sequence models for online application. Given a prefix, the monotonic attention function allows finding an encoder position [10] or the endpoint of a chunk [11] used for prediction of the next token. In our study, we showed that endpoints can also be estimated precisely and efficiently via the regular soft-attention function by controlling its uncertainty. We further showed that there are more issues to be addressed for high-performance online speech recognition, such as finalizing partial results of the beam search and the use of a bidirectional encoder, and proposed effective methods for addressing theses issues.

7. Conclusion

We have proposed and evaluated several techniques for applying S2S attention-based models to streaming speech recognition. Our results show that with these techniques it is possible to produce low latency online recognition results on the Switchboard task without a significant decrease in performance.

8. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015.

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

[3] 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*.

[4] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," *Proc. Interspeech 2018*.

[5] C. Weng, J. Cui, G. Wang, J. Wang, C. Yu, D. Su, and D. Yu, "Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition," *Proc. Interspeech 2018*.

[6] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Muller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. of Interspeech 2019*.

[7] 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," *Proc. of Interspeech 2019*.

[8] T.-S. Nguyen, S. Stueker, J. Niehues, and A. Waibel, "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation," *arXiv preprint arXiv:1910.13296*, 2019.

[9] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Advances in Neural Information Processing Systems*, 2016, pp. 5067–5075.

[10] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2837–2846.

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

[12] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, pp. 4390–4394, 2019.

[13] H. Miao, G. Cheng, P. Zhang, T. Li, and Y. Yan, "Online hybrid ctc/attention architecture for end-to-end speech recognition," *Proc. of Interspeech 2019*, pp. 2623–2627, 2019.

[14] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Towards online end-to-end transformer automatic speech recognition," 2019.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017.

[16] P. Brown, J. Spohrer, P. Hochschild, and J. Baker, "Partial traceback and dynamic programming," in *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7. IEEE, 1982, pp. 1629–1632.

[17] E. O. Selfridge, I. Arizmendi, P. A. Heeman, and J. D. Williams, "Stability and accuracy in incremental speech recognition," in *Proceedings of the SIGDIAL 2011 Conference*. Association for Computational Linguistics, 2011, pp. 110–119.

[18] S. Wachsmuth, G. A. Fink, and G. Sagerer, "Integration of parsing and incremental speech recognition," in *9th European Signal Processing Conference (EUSIPCO 1998)*. IEEE, 1998, pp. 1–4.

[19] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[20] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, D. Rybach, T. N. Sainath, and T. Strohman, "Recognizing long-form speech using streaming end-to-end models," *arXiv preprint arXiv:1910.11455*, 2019.

[21] K. Audhkhasi, G. Saon, Z. Tüske, B. Kingsbury, and M. Picheny, "Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition," *Proc. Interspeech 2019*, pp. 2618–2622, 2019.

[22] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, 2019.

[23] S. Xue and Z. Yan, "Improving latency-controlled blstm acoustic models for online speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5340–5344.

[24] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] T. S. Nguyen, J. Niehues, E. Cho, T.-L. Ha, K. Kilgour, M. Muller, M. Sperber, S. Stueker, and A. Waibel, "Low latency asr for simultaneous speech translation," 2020.

[27] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6069–6073.