Interspeech 2022
18-22 September 2022, Incheon, Korea

E2E Segmenter: Joint Segmenting and Decoding for Long-Form ASR

W. Ronny Huang, Shuo-yiin Chang, David Rybach, Rohit Prabhavalkar,
Tara N. Sainath, Cyril Allauzen, Cal Peyser, Zhiyun Lu

Google Research, USA

{wrh, shuoyiin, rybach, prabhavalkar, tsainath, allauzen, cpeyser, zhiyunlu}@google.com

Abstract

Improving the performance of end-to-end ASR models on long utterances ranging from minutes to hours is an ongoing challenge in speech recognition. A common solution is to segment the audio in advance using a separate voice activity detector (VAD) that decides segment boundary locations based purely on acoustic speech/non-speech information. VAD segmenters, however, may be sub-optimal for real-world speech where, e.g., a complete sentence that should be taken as a whole may contain hesitations in the middle ("set an alarm for... 5 o'clock").

We propose to replace the VAD with an end-to-end ASR model capable of predicting segment boundaries in a streaming fashion, allowing the segmentation decision to be conditioned not only on better acoustic features but also on semantic features from the decoded text with negligible extra computation. In experiments on real world long-form audio (YouTube) with lengths of up to 30 minutes long, we demonstrate 8.5% relative WER improvement and 250 ms reduction in median end-of-segment latency compared to the VAD baseline on a state-of-the-art Conformer RNN-T model.

Index Terms: speech recognition, speech segmentation, decoding algorithms, beam search

1. Introduction

Streaming End-to-end (E2E) models for ASR have achieved low word error rates (WERs) for short to medium length utterances of up to a few minutes long [1, 2]. However, E2E models have high WERs and suffer from deletion error on *long-form* utterances of tens of minutes to hours long [3, 4, 5]. Such utterances are found in tasks like meetings, lectures, and video captions.

A common practice for processing long-form utterances is to first segment the audio upstream with a separate voice activity detector (VAD). Whenever the VAD detects a long silence, it splits the audio at that location into two segments [6, 7], which are then processed independently by the E2E model. At each segment boundary, the beam search *finalizes* the top hypothesis by discarding all other hypotheses. This introduces more diversity into the beam search by occasionally clearing away stale hypotheses and making room for new ones, ultimately improving WER by seeing more potentially correct hypotheses. Maintaining beam diversity is particularly important for E2E models which are typically decoded with small beams.

Despite its crucial role in segmenting audio and regimenting the beam search, very little attention has been paid to improving *end-of-segment* prediction task [8, 9]. Current segmenters suffer from high latency because the VAD, by design, must wait through a long silence before deciding to segment. This delays subsequent functions like rescoring [10] or prefetching [11] that must wait for the hypotheses to be finalized. Improving the latency is important because it can improve user experience by making smart assistants more responsive via faster prefetching, or by helping dictation or captioning apps reduce the amount of "flickering" due to switching between top hypotheses. Current segmenters also suffer from high segmentation error because the VAD bases its decision purely on the audio and not the decoded text [12], which can contain semantic clues as to when to segment. Improving segmentation correctness is important because it can improve WER. As a motivating example, consider two segmentations of the spoken audio below. Note the segment boundary is denoted by "|".

Audio: "Shaq... dunks—game over!"
S1: **shaq dunks | game over**
S2: **shaq | dunks game over**

An ideal segmenter would give S1 because it separates the speech into semantically consistent chunks. However, a VAD segmenter would give S2 because the speaker pauses after saying "Shaq...". Importantly, S2's suboptimal segmentation can lead to a word error if "shack" was the top hypothesis during finalization. This is because by the time "dunks" is spoken, there is no more opportunity to revise "shack" to "shaq" due to the finalization. On the other hand, not segmenting at all would lead to bloating the beam with no diversity in the hypotheses, which could also induce word errors [13].

A related problem exists for end-of-query (EOQ) prediction, or endpointing, which historically also used audio-based VAD or EOQ detectors [14]. Recently, WER and latency gains have been achieved by combining endpointing and ASR into a single E2E model that is jointly optimized on both tasks, allowing them to share acoustic and semantic information [15, 16, 17, 18, 19].

Taking inspiration from the E2E endpointing work above, we now introduce E2E Segmenter, an E2E model jointly optimized on both end-of-segment detection and ASR tasks. A central challenge to end-to-end segmenting is that unlike the end-of-query label which indisputably belongs at the end of the transcript, there is no ground truth for where end-of-segment labels ought to be—making supervised training difficult. We address this challenge by proposing a novel end-of-segment annotation scheme based on modeling hesitations and word timings. To avoid degrading wordpiece prediction, we also introduce a new joint layer in the RNN-T architecture that independently predicts the end-of-segment token while leveraging shared acoustic and semantic features. Compared to the VAD baseline, E2E segmenter achieves quality improvements of up to 8.5% WER relative while simultaneously reducing 50th percentile latency by 250 ms on the YouTube captioning task.

2. Method

The primary job of the segmenter is to send *segment boundary signals* to the beam search in a streaming fashion. Upon

Copyright © 2022 ISCA

10.21437/Interspeech.2022-38

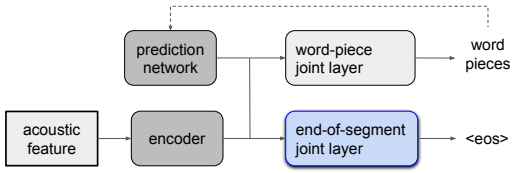

Figure 1: *RNN-T with additional joint layer for emitting the end-of-segment posterior.*

receiving this signal, the beam search finalizes the top hypothesis, clears the beam, resets the encoder state, and passes the top-hypothesis decoder state to the new segment. The decision of *when* to send the segment boundary signal is conventionally made by an upstream VAD model; but here, the signal is produced by the decoder itself whenever the top hypothesis in the beam search predicts it has reached the end-of-segment with confidence above a threshold. We now discuss how the E2E model is designed to perform this end-of-segment (<eos>) prediction task.

2.1. End-of-segment joint layer

Figure 1 illustrates our architecture, which is similar to that in [20]. The original RNN-T wordpiece joint network is a shallow, single layer of the RNN-T model that fuses both acoustic (from the encoder) and linguistic (from the prediction network) sources of information and emits token posteriors. A natural way of conferring the end-of-segment prediction task to the RNN-T decoder would be assign the joint layer an additional output logit representing <eos>, as is done for endpointing [16, 17], but we found in pilot experiments that this interferes with wordpiece decoding and hurts WER. Instead, to decouple wordpiece prediction from end-of-segment prediction, we add a second joint layer—the end-of-segment joint layer—that emits an <eos> posterior, i.e.

$$P(\texttt{<eos>}|\mathbf{x}_1, \cdots, \mathbf{x}_t, y_1, \cdots, y_u) \quad (1)$$

where $\mathbf{x}_i$ is the i-th audio frame and y_i is the i-th decoded token in the beam. The end-of-segment joint layer is identical in structure to the wordpiece joint layer, containing all wordpieces as logits. Standard wordpiece training of the RNN-T model with the wordpiece joint layer first occurs; then the end-of-segment joint layer is initialized with the same weights as the wordpiece joint layer and fine-tuned on the training data with <eos> prediction included. During inference, the wordpiece joint layer is used for wordpiece prediction while the end-of-segment joint layer is used for end-of-segment prediction.

2.2. End-of-segment annotation

While the architecture now allows for emitting the <eos> token, how can we train the model to emit it at the appropriate time? What patterns from the audio or text data can be used as supervision for when an <eos> *ought to* occur? Human annotation is expensive and inconsistent—it is not even clear in principle where best to insert segment boundaries. Thus, we opt for a heuristic-based, weak supervision approach where <eos> ground truth labels are automatically inserted into the training transcripts based on the rules shown in Table 1.

These heuristics include rules for inserting an <eos> when there is a long silence ($\geq$1.2s) or at the end of the utterance. To eliminate common mis-insertions, we also specify two exceptions for patterns where the model might otherwise insert <eos>, but are in fact places where the speaker is likely not finished with the sentence. Refer to Figure 2 for an example. Specifically these include silences following lengthened words (heyyy) or filler words (um) which signal speaker hesitation. We identify as lengthened words those with a phoneme duration exceeding 5 times the standard deviation; and we use an in-house model to detect filler words. Implementing these heuristics required obtaining silence, word, and phoneme timings via running a forced alignment model on all audio-text pairs in the training set.

Table 1: *Rules and exceptions for inserting <eos> annotations.*

	When this happens...	It's likely because...	Therefore, insert...
Rule 1	Long silence between words	Speaker finished	<eos>
Rule 2	Silence following last word	Speaker finished	<eos>
Exception 1	Silence following lengthened word (e.g. heyyy)	Speaker not finished	nothing
Exception 2	Silence following filler word (e.g. um)	Speaker not finished	nothing

Figure 2: *Example of <eos> annotation. "sil" = silence.*

2.3. FastEmit training

Now that the model emits <eos> *correctly*, we wish to make it emit *quickly*. After all, one of the advantages of E2E segmenting is that it does not need to wait a fixed silence duration before emitting <eos> like the VAD. Therefore, we train our model with the FastEmit regularization term [21] which encourages each token to be emitted as soon as sufficient context is available. During inference, the FastEmit-trained model can emit <eos> sooner than the silence duration required to insert that token during the ground truth annotation procedure (Table 1, Rule 1).

3. Setup

3.1. Dataset

YouTube videos cover many domains (TV shows, sports, conversations, etc.) and are often be very long [22], making YouTube captioning an ideal task for our long-form study. Thus, we evaluate on two standard YouTube testsets used in [23, 24, 3]: YT_LONG is sampled from YouTube video-on-demand and YT_SHORT is sampled from Google Preferred channels on YouTube. Table 2 shows their length statistics.

The training set, identical to that in [25], is a sample of Google traffic from multiple domains such as voice search,

Table 2: *Length statistics of YouTube testsets.*

Test set	# Utt.	Tot words	Tot. length	Percentile 50th	Percentile 75th
YT_LONG	77	207191	22.2h	14.8m	30m
YT_SHORT	105	84862	9.0h	6.3m	7.4m

Table 3: *(a) Main results. (b) End-of-segment threshold ablation study. Naming convention is E2E-{eos_threshold_value}*

	Segmenter	YT_LONG WER	EOS50	EOS75	# Seg.	# State	YT_SHORT WER	EOS50	EOS75	# Seg.	# State
(a)	B1: Fixed-10s	20.05	-	-	99.8	5496	13.63	-	-	33.9	3965
	B2: Fixed-20s	18.22	-	-	51.3	5753	11.57	-	-	17.5	4048
	B3: VAD	18.16	260	490	95.2	8396	11.46	460	660	28.3	5689
	E1: E2E (best)	**17.05**	**130**	**353**	56.2	8098	**10.49**	**210**	**365**	18.2	5672
	⇒ E1 vs. B3	-6.1%	-130	-137			-8.5%	-250	-265		
(b)	E2: E2E-0.0	17.60	180	550	16.4	7244	11.10	510	855	5.8	5416
	E3: E2E-0.5	17.38	180	550	18.7	7958	10.91	460	800	6.8	5705
	E4: E2E-1.0	17.19	220	535	26.3	8513	10.68	390	745	9.6	5771
	E5: E2E-1.5	17.09	195	445	38.6	8366	10.63	280	395	13.3	5700
	E6: E2E-2.0	**17.05**	130	353	56.2	8098	**10.49**	210	365	18.2	5672
	E7: E2E-2.5	17.08	100	355	80.7	7917	**10.49**	200	345	25.7	5371
	E8: E2E-3.0	17.06	100	415	116.3	7362	10.58	180	**230**	35.6	5039
	E9: E2E-3.5	17.22	90	280	178.5	6926	10.56	180	450	49.8	4790
	E10: E2E-4.0	17.48	75	**210**	300.0	6237	10.58	180	380	74.7	4569
	E11: E2E-4.5	17.76	**90**	255	477.6	5410	10.78	**140**	280	119.1	3975

farfield, telephony, and YouTube, making up about 300M utterances with 400k hours of audio. All utterances are anonymized and hand-transcribed, with the exception of YouTube being semi-supervised [26]. Note the YouTube utterances used for training are cut into small chunks no more than about 20 seconds long. The data is diversified via multi-style training [27], random down-sampling from 16 to 8 kHz [28], and SpecAug [29].

3.2. Model

Our RNN-T model is similar to the first-pass network of [2]. The encoder is a streaming 12-layer, 512 dimensional Conformer encoder with causal convolution kernels of size 15 and 8 left-context self-attention heads. The decoder consists of a stateless prediction network [30] with output dimension 640. The joint layers (both wordpiece and end-of-segment) are single layers which input the concatenation of encoder and prediction network features. In total, the model has 140M parameters, of which less than 1M are due to the additional end-of-segment joint layer. The model emits 4096 wordpieces, with the blank token factored out with HAT factorization [31]. Model training minimizes the RNN-T and MWER loss [32]. We also add the FastEmit regularization term [21] with a weight of 5e-3. The optimizer was Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. A transformer learning rate schedule [33] with peak learning rate of 1.8e-3 and 32k steps of warm-up is used, along with exponential-moving-average-stabilized gradient updates. All models are implemented in Lingvo [34] and trained on 64 TPU chips with a global batch size of 4096 for 500k steps.

3.3. Beam search

We use a frame-synchronous beam search with a beam size of 8 and a pruning threshold of 5; i.e. partial hypotheses with negative log posterior exceeding that of the top hypothesis by 5 are removed. At each frame, we apply a breadth-first search for possible expansions similar to [35], ignoring any expansion with a negative log posterior of 5 or greater, and limiting the search depth to 10 expansions. The production streaming client we run on has a maximum segment duration of 65 seconds before it forces a finalization.

3.4. Voice activity detector

Our pipeline contains a lightweight voice activity detector [36] upstream of the E2E model that classifies each frame as silence or speech in a streaming fashion. Whenever it detects 0.2 seconds of continued silence, it sends a segment boundary signal forcing the beam search to reset encoder state and discard all except the top hypothesis. VAD-based segment finalization is turned on only in our baselines; it is turned off for all E2E segmenter experiments.

4. Results

In Table 3, we run the ASR pipeline with different segmenters on YT_LONG and YT_SHORT. Other than the segmenter, all aspects of the ASR pipeline are identical. We track the following metrics for each experiment:

- **WER**: Word error rate—measure of overall ASR quality.
- **EOS50**, **EOS75**: End-of-segment latency in milliseconds, i.e., how long after speaking does the transcription get finalized. Since the only segment boundary that can be considered ground truth is the one at the end of the utterance, we measure the time difference from the end of the last word (whose timing is determined by forced alignment) to the last segment boundary, averaged across utterances. We report the 50th and 75th percentile EOS latencies. Anomalous latencies below -0.5s or exceeding 2s are left out of the percentile calculation.
- **# Segment**: Average number of segments for each utterance.
- **# State**: Average number of model states in the beam search for each utterance. This metric, used also in [13], is equivalent to the number of joint network forward passes and is thus a measure of the beam search efficiency.

4.1. Main results

In Table 3a, we first show that the quality of the segmentation matters by presenting two baseline fixed-interval segmenters, B1 and B2, and comparing it to the VAD segmenter, B3. The VAD segmentation is determined by silences and achieves better WER than the fixed-length segmenters which do not depend on any features.

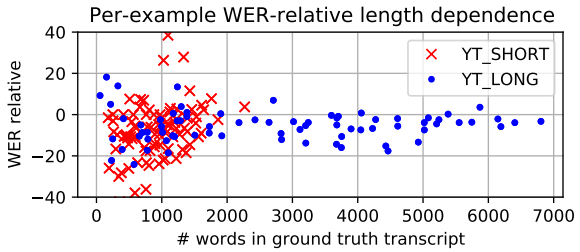

Figure 3: *Per-example WER-relative of E2E (E1) to VAD (B3) segmenters versus utterance length. Lower is better.*

Next we pick E1, our best E2E segmenter (operating point determined in §4.2), and compare it against B3, the VAD segmenter. E1 outperforms B3 by 6.1% WER relative on YT_LONG and 8.5% on YT_SHORT, highlighting the segmenter's ability to improve overall quality. E1 also finalizes the segments faster than the VAD by 130/137 ms on YT_LONG, measured at 50th/75th percentiles, and by 250/265 ms on YT_SHORT. These improvements are within perceivable range for user experience.

E1 also achieves a slightly better beam search efficiency (lower number of states), which may be due to the fact that its hypotheses are more stable, obviating the need for many joint expansions. The number of segments in the VAD and E2E are similar to those in the two fixed-length segmenters (B1-B2), which isolates out the fact that segmentation correctness matters as opposed to the number of segments.

4.2. <eos> threshold ablation study

Table 3b shows an ablation study on the <eos> threshold. When the <eos> negative log-posterior from the model falls below the <eos> threshold, the segment is finalized. Higher thresholds finalize more aggressively, leading to lower latency and more segments, but at the cost of more segmentation errors, e.g., finalizing in the middle of a sentence. Conversely, lower thresholds may not finalize as often as is needed to maintain beam diversity. The sweet spot for WER occurs—for both test sets—at a threshold of 2.0, and we pick that as our operating point.

4.3. Utterance length dependence

In Figure 3, we evaluate the *per-example* WER-relative between E1 and B3 as a function of utterance length. We define utterance length here as the number of words in the ground truth transcript rather than the audio duration (though they are correlated) because it corresponds more closely to the beam search lattice length. This allows us to analyze whether our WER gains are limited to long-form utterances. Surprisingly, WER-relative is rather invariant of utterance length for both test sets, even for utterances with a few hundred words (a few minutes). This suggests that E2E segmentation can be applied more widely to medium-form utterances as well.

4.4. Results with frame filtering

In Table 4, we evaluate the VAD and E2E segmenters with frame filtering turned *on*. Like VAD-based finalization, frame filtering starts when the VAD detects 0.2 seconds of silence, jettisoning forthcoming frames until speech is detected again. This is a practical measure to save computation for on-device deployments, because it prevents silence frames from being unnecessarily processed by the expensive E2E model. Segmenting and frame filtering are conventionally tightly coupled; when the VAD decides to segment, it simultaneously kicks off frame filtering, ensuring that the segmentation decision has access to all the audio frames. Replacing the segmenting with an E2E model requires an assessment of how it interacts with the VAD-controlled frame filtering, since segmenting may now happen before or after frame filtering begins.

Table 4: *Segmenting with frame filtering for YT_LONG. YT_SHORT results are similar and not displayed for brevity. Naming convention is {segmenter}-{margin_length}*

	YT_LONG			
Segmenter	WER	EOS50	# Seg	# State
B4: VAD-0s	20.59	260	94.4	7108
E12: E2E-0s	**19.94**	**140**	28.1	**6799**
⇒ E12 vs. B4	-3.1%	-120		
E13: E2E-1s	19.55	85	33.0	6834
E14: E2E-2s	19.21	85	41.9	6776
E15: E2E-4s	18.71	**80**	48.9	**6743**
E16: E2E-8s	18.05	105	53.1	6981
E17: E2E-16s	17.52	90	56.2	7050
E18: E2E-32s	17.18	90	56.8	7023
E19: E2E-64s	17.12	90	56.3	7228
E20: E2E-128s	**17.12**	90	56.8	7058

A first observation is that frame filtering increases absolute WER by around 2% compared to no frame filtering due to the reduced acoustic context (see B4 vs. B3 and E12 vs. E1). However E2E still prevails over VAD by about 3.1% WER relative (E12 vs. B4) and 120 ms EOS50 latency. It also achieves better beam search efficiency (lower number of states), which is aligned with frame filtering's goal of reducing computational load. Compared to no frame filtering (E1), E12's number of segments is decreased from 56.1 to 28.1. This is because the model, though trained with FastEmit, still needs to see some silence in order to confidently predict end-of-segment, and overly aggressive frame filtering prevents that silence from being seen.

The frame filtering can be gradually reduced by increasing its *margin*, or the additional silence time beyond 0.2 seconds the VAD must detect before initializing frame filtering. As the margin is increased in E13-E20, the WER converges towards its value without frame filtering (17.05%), at the cost of slightly increasing computation. The EOS50 latency is also reduced; investigating the cause of this is a point of future work. This table suggests that a trade-off between quality and beam search efficiency must be made in resource-constrained situations with frame filtering.

5. Conclusion

Our work presents a way to improve streaming long-form audio decoding by replacing the VAD-based segmenter with an E2E model. We proposed an E2E architecture that predicts segment boundaries and provided an automatic end-of-segment data annotation strategy required for learning that task in an end-to-end fashion. Our results demonstrate significant WER and end-of-segment latency improvements compared to a VAD baseline on a long-form YouTube captioning task.

6. References

[1] J. Liu, R. V. Swaminathan, S. H. K. Parthasarathi, C. Lyu, A. Mouchtaris, and S. Kunzmann, "Exploiting large-scale teacher-student training for on-device acoustic models," in *International Conference on Text, Speech, and Dialogue*. Springer, 2021, pp. 413–424.

[2] T. N. Sainath, Y. He, A. Narayanan *et al.*, "An Efficient Streaming Non-Recurrent On-Device End-to-End Model with Improvements to Rare-Word Modeling," in *Proc. of Interspeech*, 2021.

[3] C.-C. Chiu, A. Narayanan, W. Han, R. Prabhavalkar, Y. Zhang, N. Jaitly, R. Pang, T. N. Sainath, P. Nguyen, L. Cao *et al.*, "Rnn-t models fail to generalize to out-of-domain audio: Causes and solutions," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 873–880.

[4] Z. Lu, Y. Pan, T. Doutre, L. Cao, R. Prabhavalkar, C. Zhang, and T. Strohman, "Input length matters: An empirical study of rnn-t and mwer training for long-form telephony speech recognition," *arXiv preprint arXiv:2110.03841*, 2021.

[5] J. Wang, X. Tong, J. Guo, D. He, and R. Maas, "Vadoi: Voice-activity-detection overlapping inference for end-to-end long-form speech recognition," *arXiv preprint arXiv:2202.10593*, 2022.

[6] J. Ramirez, J. M. Górriz, and J. C. Segura, "Voice activity detection. fundamentals and speech recognition system robustness," *Robust speech recognition and understanding*, vol. 6, no. 9, pp. 1–22, 2007.

[7] T. Yoshimura, T. Hayashi, K. Takeda, and S. Watanabe, "End-to-end automatic speech recognition integrated with ctc-based voice activity detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6999–7003.

[8] Z. Ali and M. Talha, "Innovative method for unsupervised voice activity detection and classification of audio segments," *Ieee Access*, vol. 6, pp. 15 494–15 504, 2018.

[9] J. Hou, W. Guo, Y. Song, and L.-R. Dai, "Segment boundary detection directed attention for online end-to-end speech recognition," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, pp. 1–16, 2020.

[10] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu *et al.*, "Two-pass end-to-end speech recognition," *Proc. Interspeech 2019*, pp. 2773–2777, 2019.

[11] S.-Y. Chang, B. Li, D. Rybach *et al.*, "Low latency speech recognition using end-to-end prefetching," in *Proc. Interspeech*, 2020.

[12] M. Li, S. Zhou, and B. Xu, "Long-running speech recognizer: An end-to-end multi-task learning framework for online asr and vad," *arXiv preprint arXiv:2103.01661*, 2021.

[13] R. Prabhavalkar, Y. He, D. Rybach, S. Campbell, A. Narayanan, T. Strohman, and T. N. Sainath, "Less is more: Improved rnn-t decoding using limited label context and path merging," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5659–5663.

[14] M. Shannon, G. Simko, S.-Y. Chang, and C. Parada, "Improved end-of-query detection for streaming speech recognition." in *Interspeech*, 2017, pp. 1909–1913.

[15] R. Maas, A. Rastrow, C. Ma, G. Lan, K. Goehner, G. Tiwari, S. Joseph, and B. Hoffmeister, "Combining acoustic embeddings and decoding features for end-of-utterance detection in real-time far-field speech recognition systems," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5544–5548.

[16] S. Chang, R. Prabhavalkar, Y. He, T. Sainath, and G. Simko, "Joint Endpointing and Decoding with End-to-End Models," in *in Proc. ICASSP*, May 2019.

[17] B. Li, S.-Y. Chang, T. N. Sainath *et al.*, "Towards fast and accurate streaming end-to-end ASR," in *Proc. ICASSP*, 2020.

[18] I. Hwang and J.-H. Chang, "End-to-end speech endpoint detection utilizing acoustic and language modeling knowledge for online low-latency speech recognition," *IEEE Access*, vol. 8, pp. 161 109–161 123, 2020.

[19] L. Lu, J. Li, and Y. Gong, "Endpoint detection for streaming end-to-end multi-talker asr," *arXiv preprint arXiv:2201.09979*, 2022.

[20] S. Chang, B. Li, T. N. Sainath, C. Zhang, T. D. Strohman, Q. Liang, and Y. He, "Turn taking model for natural conversational voice system," *Interspeech*, 2022.

[21] J. Yu, C.-C. Chiu, B. Li *et al.*, "FastEmit: Low-latency Streaming ASR with Sequence-level Emission Regularization," in *Proc. ICASSP*, 2021.

[22] 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," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 920–927.

[23] H. Soltau, H. Liao, and H. Sak, "Reducing the Computational Complexity for Whole Word Models," in *Proc. ASRU*, 2017.

[24] C.-C. Chiu, W. Han, Y. Zhang *et al.*, "A comparison of end-to-end models for long-form speech recognition," in *Proc. ASRU*, 2019.

[25] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S.-y. Chang, W. Li, R. Alvarez, Z. Chen *et al.*, "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6059–6063.

[26] H. Liao, E. McDermott, and A. Senior, "Large Scale Deep Neural Network Acoustic Modeling with Semi-supervised Training Data for YouTube Video Transcription," in *Proc. ASRU*, 2013.

[27] C. Kim, A. Misra, K. Chin *et al.*, "Generation of Large-Scale Simulated Utterances in Virtual Rooms to Train Deep-Neural Networks for Far-Field Speech Recognition in Google Home," in *Proc. Interspeech*, 2017.

[28] J. Li, D. Yu, J.-T. Huang, and Y. Gong, "Improving wideband speech recognition using mixed-bandwidth training data in cd-dnn-hmm," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 131–136.

[29] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. Cubuk, and Q. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019.

[30] R. Botros, T. Sainath, R. David, E. Guzman, W. Li, and Y. He, "Tied & reduced rnn-t decoder," in *Proc. Interspeech*, 2021.

[31] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (hat)," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6139–6143.

[32] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C. C. Chiu, and A. Kannan, "Minimum Word Error Rate Training for Attention-based Sequence-to-sequence Models," in *Proc. ICASSP*, 2018.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[34] J. Shen, P. Nguyen, Y. Wu, Z. Chen, M. X. Chen, Y. Jia, A. Kannan, T. Sainath, Y. Cao, C.-C. Chiu *et al.*, "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," *arXiv preprint arXiv:1902.08295*, 2019.

[35] A. Tripathi, H. Lu, H. Sak, and H. Soltau, "Monotonic recurrent neural network transducer and decoding strategies," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 944–948.

[36] R. Zazo Candil, T. N. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform cldnns for voice activity detection," 2016.