TOWARDS FAST AND ACCURATE STREAMING END-TO-END ASR

Bo Li, Shuo-yiin Chang, Tara N. Sainath, Ruoming Pang, Yanzhang He, Trevor Strohman, Yonghui Wu

Google LLC, USA

{boboli,shuoyiin,tsainath,rpang,yanzhanghe,strohman,yonghui}@google.com

ABSTRACT

End-to-end (E2E) models fold the acoustic, pronunciation and language models of a conventional speech recognition model into one neural network with a much smaller number of parameters than a conventional ASR system, thus making it suitable for on-device applications. For example, recurrent neural network transducer (RNN-T) as a streaming E2E model has shown promising potential for on-device ASR [1]. For such applications, quality and latency are two critical factors. We propose to reduce E2E model's latency by extending the RNN-T endpointer (RNN-T EP) model [2] with additional early and late penalties. By further applying the minimum word error rate (MWER) training technique [3], we achieved 8.0% relative word error rate (WER) reduction and 130ms 90-percentile latency reduction over [2] on a Voice Search test set. We also experimented with a second-pass Listen, Attend and Spell (LAS) rescorer [4]. Although it did not directly improve the first pass latency, the large WER reduction provides extra room to trade WER for latency. RNN-T EP+LAS, together with MWER training brings in 18.7% relative WER reduction and 160ms 90-percentile latency reductions compared to the original proposed RNN-T EP [2] model.

Index Terms— RNN-T, Endpointer, Latency

1. INTRODUCTION

End-to-end (E2E) models [1, 5–12] have attracted large interest in both academia and industry. These models fold in components of the conventional automatic speech recognition (ASR) systems, namely an acoustic model (AM), pronunciation model (PM) and language model (LM), into a single neural network and optimize them jointly. E2E models simplify ASR system building and maintenance. They can have a much smaller model size than conventional ASR systems and are therefore more suitable for systems that perform the recognition on mobile devices. Among E2E variants, recurrent neural network transducer (RNN-T) [6] has shown potential for on-device streaming ASR [1].

Besides recognition quality, latency is another critical metric for streaming ASR. In this paper, we define recognition latency as the time difference between when the user stops speaking and when the system produces its final text hypothesis. It is desirable for model latency to be low enough that the system responds to the user quickly, while still high enough that it does not cut off the user's speech. Building models that have a better trade-off between word error rate (WER) and latency is crucial to achieving fast and accurate streaming speech recognition [13–15]

The decision of whether a user has stopped speaking is usually generated by an endpointer (EP) model. A voice activity detector (VAD) that detects speech and filters out non-speech is one such model. It can be used to declare an end-of-query (EOQ) as soon as VAD observes speech followed by a fixed interval of silence. VAD is not optimized to distinguish within-speech and query-end silences and may generate many false positive endpointing decisions. EOQ-based models address these issues [16]. They are directly optimized to distinguish speech and different types of silence including initial, intermediate and final. They have been shown to give better latency and WER trade-offs.

Even with EOQ, the endpointer model and the ASR model are still optimized independently. Information captured by ASR models is not shared to the endpointer, which may be useful for making endpointing decisions. It would be better to optimize the endpointer and ASR models together. E2E models make this joint optimization simpler than with conventional modeling approaches. [2] does this by folding the EOQ detector into the RNN-T model by introducing a special token (</s>), signaling the end of speech, into RNN-T's output vocabulary. It is treated the same as all the other tokens during training. However, during inference it is used as one of the signals to end a search path. Premature </s> prediction may cause not only substitution errors but also deletions.

To achieve better WER and latency trade-offs, we not only need the joint optimization of endpointer and ASR, the </s> token should also be predicted as close to the end of the last word as possible. In this work we propose to extend the joint RNN-T endpointer (EP) model [2] in a number of ways. First, we introduce penalties for emitting </s> too early or late in training, to encourage the model to find a good WER and latency trade-off. These penalties are applied to the </s> token, where the ground truth is obtained from a forced alignment between the transcript and audio signals.

Second, premature </s> prediction causes a sequence level loss rather than a single token's. This leads us to explore whether sequence training [3, 17, 18] would address this problem. We hence investigated minimum word error rate [3] training for RNN-T EP models, which is found to yield both a WER and latency improvements. Third, we rescore RNN-T EP's hypotheses with a non-streaming model, namely Listen, Attend and Spell (LAS) [4]. The direct modeling of </s> in RNN-T makes the score combination with LAS, which emits </s> already, more consistent. While the rescoring model does not directly change the latency of RNN-T, WER gains it brings gives us more room for potential WER and latency trade-offs. The final setup, RNN-T EP with late penalty, LAS rescoring and MWER training, achieves a 18.7% relative WER reduction and 40ms median latency and 160ms 90-percentile latency reductions on a Voice Search task comparing to the original RNN-T EP [2].

The rest of the paper is organized as follows. Section 2 explains the model architecture of the RNN-T EP and then details the proposed improvements of the RNN-T EP model using early and late penalties, MWER training and LAS rescoring. Section 3 and 4 presents the experimental setup, results and analysis.

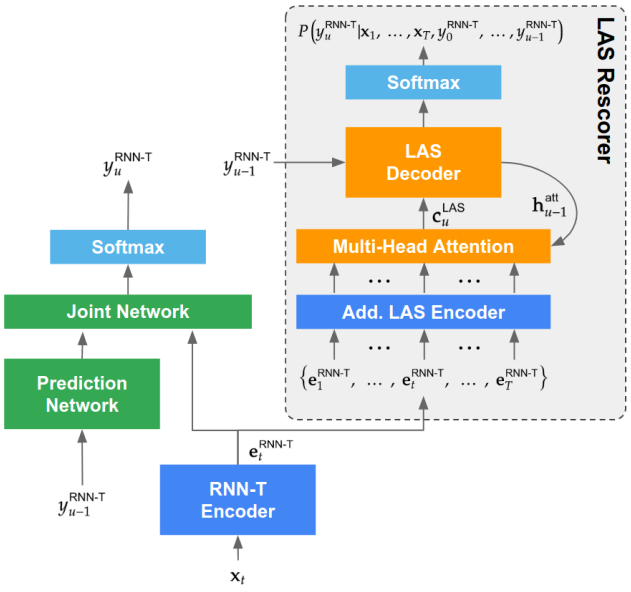

Fig. 1: Recurrent neural network transducer and endpointer (RNN-T EP) with non-streaming Listen, Attend and Spell (LAS) rescoring.

2. RNN TRANSDUCER AND ENDPOINTER

The recurrent neural network transducer and endpointer (RNN-T EP) model explored in this work is shown in Figure 1. Let us denote the input acoustic frames as $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^d$ are log-mel filterbank energies ($d = 512$) and T is the number of frames in $\mathbf{x}$. Each acoustic frame $\mathbf{x}_t$ is first passed through the RNN-T encoder, which consists of multiple layers of unidirectional LSTM layers. We denote the output of the RNN-T encoder as $\mathbf{e}_t$ and it is then forwarded to the RNN-T decoder for producing $y_t^{\text{RNN-T}}$. The output is decoded as soon as the input is encoded, without introducing additional latency incurred when processing the entire utterance at once. In this work, RNN-T is trained to directly predict word piece token sequence $\mathbf{y} = \{y_1, \ldots, y_U\}$ where the last label y_U is the special token </s> .

2.1. Early and Late Penalties

$$\log \mathrm{P}_{\text{RNN-T}}(y_U|\mathbf{x}_t) \; -= \; \big(\max(0,\; \alpha_{\text{early}} * (t_{</s>} - t)) + \max(0,\; \alpha_{\text{late}} * (t - t_{</s>} - t_{\text{buffer}}))\big) \tag{1}$$

Extending RNN-T's output vocabulary with a special token </s> helps improve its latency [2], as the endpointing decision is made jointly with the model rather than with a separate endpointer. However, there is no constraint on when </s> should occur during training. A premature </s> prediction can result in deletion errors, while late predictions of </s> can increase latency as </s> is used to inform the system when the speech ends. In this paper, we address these issues by applying additional early and late penalties on the </s> token (Equation (1)). Specifically, during training for every input frame in $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ and every label $\{y_1, \ldots, y_U\}$, RNN-T computes a $U \times T$ matrix $\mathbf{P}_{\text{RNN-T}}(\mathbf{y}|\mathbf{x})$, which is used in the training loss computation. The last label y_U is always </s> . We denote $t_{</s>}$ as the frame index after the last non-silence phoneme, obtained from the forced alignment of the audio with a conventional model. The RNN-T log-probability $\log \mathbf{P}_{\text{RNN-T}}(y_U|\mathbf{x})$ is modified to include a penalty at each time step t for predicting </s> too early or too late. t_{buffer} gives a grace period after the reference $t_{</s>}$ before the late penalty is applied. α_{early} and α_{late} are scales on the early and late penalties respectively. All hyper parameters are tuned experimentally.

2.2. MWER Training

Minimizing RNN-T loss corresponds to improving the log-likelihood of the training data. However, ASR system performance is measured in terms of WER, not log-likelihood. To address this mismatch, [19] proposes to minimize expected WER of the RNN-T model by approximating the expectation with samples draw from the model. Minimum word error rate training (MWER) is later applied to attention based LAS E2E models [3].

During the beam search decoding of the RNN-T EP model, the inference is terminated when either a blank symbol is generated at the last input frame or an </s> token is predicted. Premature </s> prediction results in deletion of the remaining reference target sequence, leading to a large sequence loss. This makes it more suitable for sequence training techniques. In this work, we hence investigate MWER training with N-best hypotheses for the RNN-T EP model.

2.3. Listen, Attend and Spell Rescoring

Non-streaming E2E models such as Listen, Attend and Spell (LAS) has shown better performance than streaming ones such as RNN-T. LAS has been explored to serve as a second pass rescorer [4], that can still fit within the on-device latency constraints. As illustrated in Figure 1, the model first collects the output of the RNN-T encoder of all the frames $\mathbf{e} = [\mathbf{e_1}, \ldots, \mathbf{e_T}]$. They are then forwarded through an extra LAS encoder to generate a new set of encoder features for the LAS decoder. The decoder then computes output $\mathbf{y}_{\text{LAS}}$ accordingly. During inference, we first pick the top-K hypotheses from the RNN-T decoder. We then run the LAS model on each sequence in the teacher-forcing mode to compute a score, which combines log probability of the sequence and the attention coverage penalty [20]. The sequence with the highest LAS score is picked as the output sequence.

One of the issues in [4] is that RNN-T did not produce a score for </s> , while LAS is indeed trained to produce a score for it. Thus, when rescoring RNN-T hypotheses, an "artificial" </s> score for RNN-T was added to the </s> from LAS. One can argue that including a score for </s> generated from RNN-T based on the inputs should help recognition, as it gives more confidence as to if the sentence should actually be completed. In this work, we look at improving LAS rescoring with the RNN-T EP model by including a score for </s> . The use of </s> token in RNN-T makes the score combination with LAS more consistent acorss all the output units. It is important to note that LAS rescoring cannot make the RNN-T model emit </s> faster; it can only improve the WER of RNN-T. However, the improvement of WER may provide additional room to trade WER for latency.

3. EXPERIMENTAL SETUPS

3.1. Dataset

We use the same multidomain dataset as [21] for training. Multistyle training (MTR) is used for noise robustness [22]. During training, a

noise configuration, which defines mixing conditions like the size of the room, reverberation time, position of the microphone, speech and noise sources, signal to noise ratio (SNR), etc, for each utterance is randomly sampled from a collection of 3 million pre-generated configurations. The detailed noise configuration can be found in [21]. The test set we use consists of 14K Voice Search utterances with duration less than 5.5 seconds long. They are all anonymized and hand-transcribed, and are representative of Google traffic.

3.2. Modeling

The input waveforms are framed using a 32 msec window with 10 msec shift. Globally normalized 128 dimension logmel features extracted from frequencyies spanning from 125 Hz to 7.5kHz are used as inputs. The input window size is 4, consisting of 3 frames on the left and no future context. It is further subsampled by a factor of 3 making the system operate at 33 Hz [23].

Similar to [21], multidomain models are trained with domain id as an additional input for learning domain-dependent variations. Following [1], all LSTM layers in the model are unidirectional, with 2048 units and a projection layer with 640 units. The RNN-T encoder consists of 8 LSTM layers, with a time-reduction layer after the second layer. The RNN-T decoder consists of a prediction network with 2 LSTM layers, and a joint network with a single feed-forward layer with 640 units. The additional LAS encoder consists of 2 LSTM layers. The LAS decoder consists of multi-head attention [24] with 4 attention heads, which is fed into 2 LSTM layers. All models are trained on 8x8 Cloud TPU using the Tensorflow Lingvo toolkit [25] to predict 4,096 word pieces including the </s> token.

3.3. Inference

Despite the use of multidomain training, this work focuses only on the Voice Search task. We append the </s> token only to the Voice Search queries and keep the other data untouched. We report both the recognition performance in terms of word error rate (WER) and the latency of the models for Voice Search only. The latency metrics used in this paper includes median latency (EP50), 90 percentile latency (EP90) and the endpointing coverage (EOU) which represents the percentage of the test data actually receives an end-of-utterance signal from the endpointer model.

There is a trade-off between accuracy and latency, which is often depicted by ROC curves. For EOU EPs, it is obtained by adjusting the endpointing decision threshold. For RNN-T EPs, the endpointing decision is defined by:

$$p(\texttt{</s>}|\mathbf{x}_1, \ldots, \mathbf{x}_t, y_0^{\text{RNN-T}}, \ldots, y_{t-1}^{\text{RNN-T}})^{\alpha_{\texttt{</s>}}} \geq \beta. \quad (2)$$

$\alpha_{\texttt{</s>}}$ is a penalty term for the posterior of </s> that modifies the ordering for the hypothesis with </s> . β is a predefined threshold that determines if </s> is allowed in the search beam [2]. Sweeping $\alpha_{\texttt{</s>}}$ and β gives us a ROC curve of the WER and latency trade-off. For simplicity, we most of the time report a single trade-off point and only show the ROC curves at the end for the final comparisons.

4. RESULTS

4.1. Baseline

We first train a RNN-T model to predict 4,096 word pieces for the ASR task only (no </s>) as was done in past [1]. This RNN-T cannot be used to output an endpointing decision and an external EOQ EP is used [2, 16] (B1 in Table 1). The endpointer and the RNN-T ASR model are trained independently and at the inference time, the information from RNN-T's hypotheses cannot be used for endpointing decisions. To address this issue, we also trained a joint endpointing and recognition RNN-T EP model proposed in [2] (B2 in Table 1). As suggested in [2], we also use the independnetly trained EOQ EP as a backup for the RNN-T EP model. From Table 1, the RNN-T EP (B2) shows good latency gains (130ms EP50 and 200ms EP90 latency reductions and a 5.4% absolute EOU coverage improvement) but has an increase of 0.3% WER. One assumption of this regression is that during training </s> is treated the same as all the other tokens, with no constraint on how early or late </s> should occur; however in inference, a path ends when a </s> token is predicted. Predicting EOS prematurely brings in deletion errors.

Table 1: Quality and latency performance of the baseline models.

Exp.	WER (%)	EP50 (ms)	EP90 (ms)	EOU (%)
B1 RNN-T	**7.2**	540	910	86.7
B2 RNN-T EP	7.5	**410**	**710**	**92.1**

Table 2: Quality and latency performance of models with early and late penalties.

Exp.	WER (%)	EP50 (ms)	EP90 (ms)	EOU (%)
E1 Early	7.2	430	830	90.7
E2 E1 + 3Frame_Late	7.2	**380**	850	88.3
E3 E1 + 5Frame_Late	7.2	400	**790**	**91.5**
E4 E1 + 7Frame_Late	7.2	540	860	90.8

4.2. Early and Late Penalties

To address the potential premature </s> prediction, we adopt an early penalty term to the training. It is added only if </s> is predicted at any frame earlier than its ground truth time. When adding the early penalty, we scale it by a factor of 0.1 which is found to work well. This (E1 in Table 2) reduces the WER from 7.5% to 7.2% but degrades on latency comparing to B2. The use of early penalty does help the model to address premature </s> prediction but has the risk of the model learning to over-delay its predictions, which leads to worse latency. The regression on EP90 is more sever which is because many tail cases are not endpointed by RNN-T EP and they simply fall back to the EOQ EP.

We further introduce a late penalty term to penalize the </s> prediction that happens too late comparing to the ground truth. During training the granularity of the time is frame (particularly 60ms in our setup). We experimented with $t_{\text{buffer}} = \{3, 5, 7\}$ which corresponds to a grace period of 180ms, 300ms and 420ms after the reference </s> label. The results are presented in Table 2. With 3 frames' buffer, we obtain the best median latency but 5-frame gives the best 90-percentile latency which is still worse than B2. We take model E2, namely the RNN-T EP model with early penalty and 3-frame late penalty, as the setup for following experiments.

Table 3: Quality and latency performance of models w and w/o MWER training.

Exp.	WER (%)	EP50 (ms)	EP90 (ms)	EOU (%)
B1 RNN-T	7.2	540	910	86.7
B2 RNN-T EP	7.5	410	710	92.1
B3 B1 + MWER	**6.9**	540	910	86.7
E2 B2 + Early + Late	7.2	**380**	850	88.3
E4 E2 + MWER	7.2	430	630	**97.3**
E5 E4 - Early	**6.9**	**380**	**580**	95.5

Table 4: Quality and latency performance of models with 2nd pass LAS rescoring.

Exp.	WER (%)	EP50 (ms)	EP90 (ms)	EOU (%)
B2 RNN-T EP	7.5	410	710	92.1
E2 B2 + Early + Late	7.2	380	850	88.3
E6 E2 + LAS	6.4	380	850	88.3
+ re-sweep	6.4	370	740	91.4
+ ignore RNN-T </s> score	6.6	370	740	91.4
E7 E6 + MWER LAS only	6.2	**350**	620	92.4
E8 E7 + MWER All	**6.1**	370	**550**	**95.2**

4.3. MWER training

For the RNN-T EP model, a wrong prediction of </s> leads to not just a token error but a sequence level loss as it is used to terminate a path in beam search. MWER training optimizes sequence level loss and penalizes WER when </s> is emitted too early, thus prompting our investigation in this section.

We conducted MWER training for the RNN-T model without </s> (B1) and the best RNN-T EP (E2). Both the pre- and post-MWER results are reported in Table 3. For B1, the latency is controlled by a separate EOU EP and hence remains the same after MWER training (B3). But the WER reduces from 7.2% to 6.9%. While for E2, MWER training (E4) maintains the same 7.2% WER as E2 but achieves 220ms EP90 reduction with 50ms regression on EP50. Because optimizing MWER already penalizes premature </s> predictions, we turn off the early penalty for MWER training. This (E5 in Table 3) reduces WER from 7.2% to 6.9% WER and more importantly it still yields a 270ms EP90 latency reduction while maintaining the same EP50 latency as E2. MWER training of the RNN-T EP model with only late penalty can bring in both WER and latency improvements. Comparing to B2, E5 gives 8.0% relative WER reduction and 30ms EP50 and 130ms EP90 latency reductions.

4.4. LAS Rescoring

So far we see good latency reductions, but the WER gains are small. In the literature, two-pass model that runs RNN-T as the first pass streaming model for fast response and LAS as the rescorer has been shown to be effective in WER reductions. We hence investigate the effect of LAS rescoring on RNN-T EP model. We took the pre-MWER model E2 and added an additional encoder with two LSTM layers and an extra LAS decoder (Figure 1). They are trained with cross entropy (CE) loss with the RNN-T weights frozen. The results are presented as E6 in Table 4. In this work, the latency is only measured for the first pass model. With the same decoding configuration as E2, LAS rescoring reduces the WER by 11.1% relative from 7.2% to 6.4%. Although LAS rescoring cannot directly affect first pass latency, with the WER gains, we may be able to trade WER for latency. We further swept the penalty scale for </s> and obtained an operation point with the same 6.4% WER but 10ms EP50 and 130ms EP90 reductions. As mentioned in Section 2.3, one problem for LAS rescoring of RNN-T without </s> as done in [4] is that RNN-T does not generate an explicit </s> score to combine with that from LAS. To simulate that effect, we zeroed out the </s> score from RNN-T EP and swept a global value to be combined with LAS </s> score. The result (E6 + ignore RNN-T </s> score in Table 4) shows an increase in WER from 6.4% to 6.6%, highlighting the benefit of using </s> in RNN-T EP for LAS rescoring.

Instead of CE loss, MWER loss of RNN-T outputs can be used to update the LAS rescorer (E7 in Table 4). It further reduces the WER down to 6.2% and obtains 100ms EP90 reductions. Moreover, when we update both RNN-T and LAS during MWER (E8), we can obtain another 70ms EP90 reduction. Comparing to B2, the RNN-T EP + LAS with MWER training gives a 18.7% relative WER reduction and 40ms EP50 and 160ms EP90 reductions.

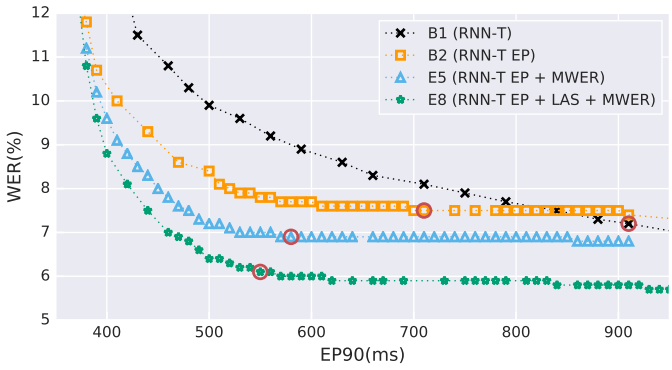

Fig. 2: ROC curves of WER and 90-percentile latency (EP90) trade-offs for RNN-T (B1), the original RNN-T EP (B2), the proposed RNN-T EP with late penaly and MWER training (E5) and with additional LAS rescoring (E8). Red circles represent operation points reported in early sections.

4.5. Analysis

The proposed RNN-T EP with late penalty and MWER training (E5) gives us both WER and latency improvement over the RNN-T (B1) and the original RNN-T EP (B2). Further WER improvement is achieved via a second pass LAS rescorer (E8). In this section, we compare these systems across different operating points. We plotted the WER vs latency (EP90) curve for these four models (B1, B2, E5, E8) in Figure 2 by varying the penalty scale $\alpha_{\texttt{</s>}}$ and threshold β. Lower curves are better. RNN-T (B1) tends to delay outputs and has worse latency. With </s> , RNN-T EP (B2) addresses the latency problem but with some WER degradations. With the modifications proposed in this work, namely late penalty, MWER and LAS rescoring, both E5 and E8 have much better WER and latency trade-offs.

5. REFERENCES

[1] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-Yiin Chang, Kanishka Rao, and Alexander Gruenstein, "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[2] Shuo-Yiin Chang, Rohit Prabhavalkar, Yanzhang He, Tara N. Sainath, and Gabor Simko, "Joint Endpointing and Decoding with End-to-end Models," in *Proc. ICASSP*. IEEE, 2019, pp. 5626–5630.

[3] Rohit Prabhavalkar, Tara N. Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan, "Minimum Word Error Rate Training for Attention-based Sequence-to-Sequence Models," in *Proc. ICASSP*, 2018.

[4] Tara N. Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirko Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, Ian McGraw, and Chung-Cheng Chiu, "Two-Pass End-to-End Speech Recognition," *Proc. Interspeech*, 2019.

[5] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art Speech Recognition With Sequence-to-Sequence Models," in *Proc. ICASSP*, 2018.

[6] Alex Graves, "Sequence Transduction with Recurrent Neural Networks," *CoRR*, vol. abs/1211.3711, 2012.

[7] Kanishka Rao, Hasim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

[8] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, Attend and Spell," *CoRR*, vol. abs/1508.01211, 2015.

[9] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[10] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[11] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "RWTH ASR systems for LibriSpeech: Hybrid vs Attention," *Proc. Interspeech*, 2019.

[12] Haoran Miao, Gaofeng Cheng, Pengyuan Zhang, Ta Li, and Yonghong Yan, "Online Hybrid CTC/Attention Architecture for End-to-End Speech Recognition," *Proc. Interspeech*, pp. 2623–2627, 2019.

[13] Shuo-Yiin Chang, Bo Li, and Gabor Simko, "A unified end-pointer using multitask and multidomain training," in *Proc. ASRU*. IEEE, 2019.

[14] Shuo-Yiin Chang, Bo Li, Tara N. Sainath, Gabor Simko, and Carolina Parada, "Endpoint Detection Using Grid Long Short-Term Memory Networks for Streaming Speech Recognition," in *Proc. Interspeech*, 2017.

[15] Shuo-Yiin Chang, Bo Li, Tara N. Sainath, Gabor Simko, Anshuman Tripath, Aaron van den Oord, and Oriol Vinyals, "Temporal modeling using dilated convolution and gating for voice-activity-detection," in *Proc. ICASSP*, 2018.

[16] Matt Shannon, Gabor Simko, Shuo-Yiin Chang, and Carolina Parada, "Improved end-of-query detection for streaming speech recognition," in *Proc. Interspeech*, 2017.

[17] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*. IEEE, 2009, pp. 3761–3764.

[18] Karel Veselỳ, Arnab Ghoshal, Lukas Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, vol. 2013, pp. 2345–2349.

[19] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[20] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *Proc. Interspeech*, 2016.

[21] Arun Narayanan, Ananya Misra, Khe Chai Sim, Golan Pundak, Anshuman Tripathi, Mohamed Elfeky, Parisa Haghani, Trevor Strohman, and Michiel Bacchiani, "Toward domain-invariant speech recognition via large scale training," in *Proc. SLT*. IEEE, 2018, pp. 441–447.

[22] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara N. Sainath, and Michiel Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," *Proc. Interspeech*, 2017.

[23] Golan Pundak and Tara N. Sainath, "Lower frame rate neural network acoustic models," 2016.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[25] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, et al., "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," *arXiv preprint arXiv:1902.08295*, 2019.