INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Emitting Word Timings with End-to-End Models

Tara N. Sainath, Ruoming Pang, David Rybach, Basi García, Trevor Strohman

Google Inc., U.S.A

`{tsainath, rpang, rybach, basigarcia, strohman}@google.com`

Abstract

Having end-to-end (E2E) models emit the start and end times of words *on-device* is important for various applications. This unsolved problem presents challenges with respect to model size, latency and accuracy. In this paper, we present an approach to word timings by constraining the attention head of the Listen, Attend, Spell (LAS) 2nd-pass rescorer [1]. On a Voice-Search task, we show that this approach does not degrade accuracy compared to when no attention head is constrained. In addition, it meets on-device size and latency constraints. In comparison, constraining the alignment with a 1st-pass Recurrent Neural Network Transducer (RNN-T) model to emit word timings results in quality degradation. Furthermore, a low-frame-rate conventional acoustic model [2], which is trained with a constrained alignment and is used in many applications for word timings, is slower to detect start and end times compared to our proposed 2nd-pass LAS approach.

1. Introduction

End-to-end (E2E) models have shown very promising results on a variety of speech recognition tasks [3, 4, 5, 6, 7, 8, 9]. These models attempt to fold the acoustic, pronunciation and language models (AM, LM, PM) of a conventional system into one neural network. They are particularly attractive for on-device applications, as they are a fraction of the size of a conventional cloud-based system [2]. E2E models also bring added latency and privacy benefits compared to conventional cloud-based systems. To date these models have been deployed for applications such as assistant, dictation and video transcription [3, 4].

While E2E models have the simplicity and size benefits compared to conventional models, this simplicity can come at a cost. For example, because the model is not trained with alignments, like a conventional model, the model often delays its output predictions. This makes it very difficult to get the timing of words, which is needed for many transcription applications. Since conventional models are trained with a phoneme alignment, it is very straightforward to get accurate word timings.

There are numerous challenges in estimating word timings from hypotheses emitted from E2E models *on-device*. First, there is a size constraint that the word timing model must fit on device. Therefore, using a large conventional model to do a forced alignment from the top hypothesis is not a viable solution. In addition, having a post-processing module after the final recognition hypothesis increases overall latency, which is not preferable. Finally, constraining the alignment restricts the freedom of the model and could lead to a loss in quality [10], which is also not desired.

In light of these constraints, we explore emitting word timings by using a 2nd-pass model. Recently, a 2-pass RNN-T+LAS model was proposed in [1], where LAS rescores hypotheses from RNN-T in very minimal time (< 200ms). Recently, this model was launched for on-device assistant applications, as it allowed the E2E model to achieve a better quality and latency tradeoff compared to a conventional model [3].

The attention probabilities in LAS learn an alignment between the audio and predicted subword units (i.e., graphemes, wordpieces, etc.). We look to constrain the attention probabilities of LAS based on word level alignments, allowing us to obtain the start and end time of each word. Our idea to use attention probabilities for word alignments is also motivated by previous work in machine translation [11], which learns word alignments from attention probabilities.

Our experiments are conducted on a Voice Search task. We find that constraining the alignment of the 1st-pass RNN-T model results in a quality degradation. In comparison, constraining the attention head of the LAS rescorer does not result in any quality degradation. In addition, the predicted word start and end times are on average within less than 100ms of ground truth start and end times. In comparison, a conventional model [2], commonly used to obtain word timings, has predicted start and end word times that are within 170ms of the ground truth start and end times.

The rest of this paper is organized as follows. The 2-pass model architecture is presented in Section 2. Section 3 describes the various word timings models explored. Experiments and results are presented in Section 4 and Section 5, respectively. Finally, Section 6 concludes the paper and discusses future work.

2. Two-Pass E2E Architecture

The two-pass architecture [1] used in this work is illustrated in Figure 1. The input acoustic frames are denoted as $\mathbf{x} = (\mathbf{x}_1 \ldots \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^{128}$ is a frame of log-mel filterbank energies and T is the number of frames. We denote the ground-truth label sequence of length U as $\mathbf{y} = (y_1, \ldots, y_U)$, where $y_u \in \mathcal{Z}$ and $\mathcal{Z}$ corresponds to the set of word-piece [12] output units.

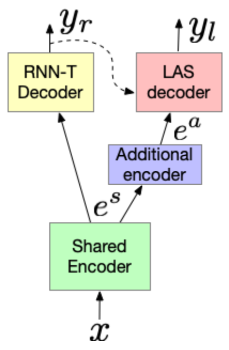

Figure 1: *Two-Pass Architecture*

Training takes place in two stages. First, the shared encoder and RNN-T decoder are trained to maximize $P(\mathbf{y}_r = \mathbf{y}|\mathbf{x})$. Next, the shared encoder is held fixed and the additional encoder (AE) + LAS decoder (including the attention context) are trained to maximize $P(\mathbf{y}_l = \mathbf{y}|\mathbf{x})$. We add the AE since it is found to be useful to adapt the encoder output to be more suitable for LAS 2nd-pass models. Decoding is performed in two passes. In the first pass each acoustic frame $\mathbf{x}_t$ is passed through a shared encoder, consisting of a multi-layer LSTM, to get output $\mathbf{e}^s_t$, which is passed to the RNN-T decoder which emits $\mathbf{y}_r$ in a streaming fashion. In the second pass, the shared encoder output for all frames, $\mathbf{e}^s = (\mathbf{e}^s_1 \ldots \mathbf{e}^s_T)$ is passed to a small AE to generate $\mathbf{e}^a = (\mathbf{e}^a_1 \ldots \mathbf{e}^a_T)$, which is then passed to a non-causal attention module to compute a context vector $\mathbf{c}_u$ which summarizes the encoder features $\mathbf{e}^a$ for a each output step $u \in U$.

In this work, the LAS decoder runs in *rescoring mode*, meaning decoder consumes the top-K hypotheses from the RNN-T decoder. The LAS decoder computes a score for each hypothesis by evaluating the network in teacher-forcing mode, with attention on $\mathbf{e}^a$. The resulting score combines the log probability of the sequence and the attention coverage penalty [13]. The sequence with the highest LAS score is chosen as the final output sequence.

3. Word Timing Models

3.1. Developing a Constrained Alignment Matrix

One challenge with word timings is that we need to emit the *start* and *end* of each word. Each word is tokenized into wordpieces, and often an entire word (e.g. `Google`) can be represented by a single wordpiece (e.g. `_Google`). If we estimate word timings by looking at the time each wordpiece was emitted, using wordpieces makes it difficult to detect both start and end times. It should be noted that using graphemes, which explicitly model silence characters between words, can better help detect start and end times. However, our choice of wordpieces over graphemes is mainly due to improved accuracy, faster decoding [4] and contextual biasing [14].

Therefore to allow wordpiece models to emit word timings, we introduce a `<wb>` symbol before each word in the transcript. Thus the transcript `the cat sat </s>` now becomes `<wb> the <wb> cat <wb> sat</s>` .

A second challenge with word timings is that our E2E models are trained to emit wordpieces. However, the alignment information for each utterance comes from performing a forced alignment with a conventional model [2] against the true word sequence, and this alignment is at the phoneme and word level only. Because we really only care about the start and end of each word, we constrain the `<wb>` token associated with each word to occur as close to the beginning of word alignment. Similarly, the last wordpiece of each word or the `</s>` symbol is constrained to occur as close to the end of the word alignment as possible. All other wordpieces in the word are constrained to occur anywhere within the start/end of the word.

Given a ground truth start/end time t_{truth} for each word, we allow a tunable t_{buffer} to the left and right of this time to get a an allowable $(t^u_{\text{start}}, t^u_{\text{end}})$ pair for each wordpiece unit u. In other words $t^u_{\text{start}} = t^u_{\text{truth}} - t^u_{\text{buffer}}$ and $t^u_{\text{end}} = t^u_{\text{truth}} + t^u_{\text{buffer}}$. Figure 2 shows a picture of a constrained alignment matrix with these pairs for each unit. For each unit $u \in U$ and time $t \in T$, we will denote each element in the constrained alignment matrix as $c(u,t)$. $c(u,t)$ takes value 1 at places of allowable alignment and 0 otherwise.

Next, we discuss how to apply a constrained alignment matrix to both 1st-pass RNN-T and 2nd-pass LAS training.

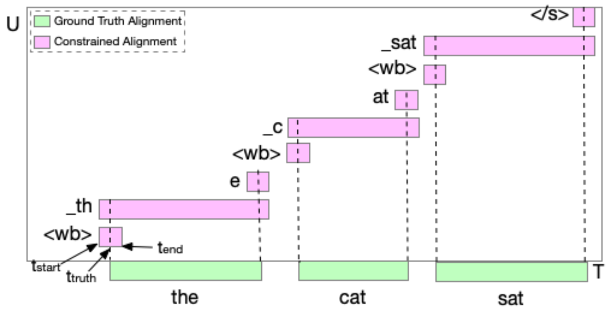

Figure 2: *Constrained Alignment*

3.2. Constrained Alignment with RNN-T

RNN-T generates a $U \times T$ matrix $\log P_{RNN-T}(\mathbf{y}|\mathbf{x})$ across all labels $\mathbf{y} = (y_1, \ldots, y_U)$ and time frames $\mathbf{x} = (\mathbf{x}_1 \ldots \mathbf{x}_T)$. This matrix represents all possible alignments of time frames and labels. Constraining the allowed alignments in this $U \times T$ matrix results in training RNN-T with a constrained alignment. This idea has been explored previously for both CTC training [15] and training RNN-T to emit the `</s>` label early [10].

Using the constrained alignment matrix from Section 3.1, we constrain $\log P_{RNN-T}(\mathbf{y}|\mathbf{x})$ in 2 ways. Extending the work of [10], first each ground-truth start times t^u_{start}, is defined as the frame index with the maximum non-zero index in $c(u,t)$. In other words

$$t^u_{\text{start}} = \operatorname*{argmin}_t c(u,t) \tag{1}$$

Next for all times t and units u, we linearly penalize the distance between the predicted time t and the true start time t^u_{start}, where α_{early} is a hyper parameter that controls the weight for the early penalty. This linear penalty is multiplied by $1 - c(u,t)$ so that we only penalize in places where the alignment is not allowed (i.e., $c(u,t) = 0$). The total penalty is given by Equation 2.

$$\mathcal{P}_{\text{early}} = \sum_{u=0}^{U-1} \sum_{t=0}^{t^u_{\text{start}}-1} \max(0,\ \alpha_{\text{early}} \times (1 - c(u,t)) \times (t^u_{\text{start}} - t)) \tag{2}$$

Similarly, we also use a late penalty to penalize for emitting too late. The ground-truth end time t_{end} time is defined as

$$t^u_{\text{end}} = \operatorname*{argmax}_t c(u,t) \tag{3}$$

Using this, the late penalty is defined as follows:

$$\mathcal{P}_{\text{late}} = \sum_{u=0}^{U-1} \sum_{t=t^u_{\text{end}}}^{T-1} \max(0,\ \alpha_{\text{late}} \times (1 - c(u,t)) \times (t - t^u_{\text{end}})) \tag{4}$$

where α_{late} is a hyper parameter that controls the weight for the late penalty.

During training, the total log probability of a RNN-T model with early and late penalty is hence a summation of three terms, as shown in Equation 5. The RNN-T model is optimized using the forward-backward algorithm [6].

$$\log P_{RNN-T}(\mathbf{y}|\mathbf{x}) - = (\mathcal{P}_{\text{early}} + \mathcal{P}_{\text{late}}). \tag{5}$$

During decoding, a frame synchronous beam search is used for RNN-T, which means at each time frame RNN-T emits a set of optional labels followed by blank. This ensure that we know the time frame that each non-blank label was emitted. Word start times are calculated by finding the location of `<wb>` tokens, and word-end times as the last wordpiece for each word or the `</s>` token for the last word.

3.3. Constrained LAS attention head

In LAS, the alignment is done via the attention mechanism, which produces a $U \times T$ matrix representing the alignment of the audio encoder features against the labels. We can constrain alignment in LAS by constraining the attention probability to the true alignment constraint matrix in Section 3.1. Our LAS models use multi-head attention (MHA) [16], which we have previously found to work well [1, 5] with LAS. We explore constraining just one of the attention heads, so the other attention heads have freedom to look anywhere within the utterance. We did not run any experiments constraining more than one attention head since we wanted to give other heads the same freedom as an unconstrained model since they were not be used to calculated word timings.

During training, given attention probabilities for selected "constrained" attention head $a(u, t)$ and the ground-truth constrained alignment matrix $c(u, t)$ for all (u, t) pairs, we penalize $a(u, t)$ if it is non-zero when the true constrained alignment $c(u, t)$is zero. In other worse, the attention alignment loss is calculated as follows, where β is a hyperparameter controlling the weight of the attention loss.

$$\mathcal{L}_{\text{attention}} = \beta \times \sum_{t=1}^{T} \sum_{u=1}^{U} a(u, t) \times (1 - c(u, t)). \tag{6}$$

During training, the total log loss of a LAS model includes the standard LAS loss, $P_{LAS}(\mathbf{y}|\mathbf{x})$, as well as the attention loss. The LAS model optimized using a cross-entropy criterion.

$$\mathcal{L} = \log P_{LAS}(\mathbf{y}|\mathbf{x}) + \mathcal{L}_{\text{attention}}. \tag{7}$$

During decoding, a label synchronous beam search is used for LAS, which means that it emits a unit u at each step in the beam search. The word timing for each unit is calculated by finding the index (i.e., time) that gives the maximum constrained attention head probability for this unit. Similar to RNN-T, word star times are calculated from the timing of `<wb>` and word end times from the time of the last wordpiece in each word or `</s>` token.

While the constrained alignment is somewhat monotonic in training, we cannot ensure this in decoding. Therefore, to obtain monotonic word timings, we modify the alignment times in a post-processing step after initial word timings are obtained from the attention heads. Specifically, if wordpiece units `a b` are such that `a` is emitted at $t = 10$ and b is emitted as $t = 3$, we modify the timing of b to be $t = 10$.

4. Experimental Details

4.1. Data Sets

The training set used for experiments, the same as [17], consists of multi-domain utterances spanning domains of search, farfield, telephony and YouTube English utterances. The search and farfield utterances are anonymized and hand-transcribed, and are representative of Google's voice search traffic. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation such that the overall SNR is between 0dB and 30dB, with an average SNR of 12dB [18]. The noise sources are from YouTube and noisy environmental recordings. To obtain alignment information for each utterance, a forced alignment is done with a conventional low frame rate (LFR) model [2].

Evaluation is done on a Voice Search *Mechanical Turk* set with around 1,200 utterances. The data is anonymized and hand-transcribed.

4.2. Modeling

All experiments use 128-dimensional log-Mel features, computed with a 32ms window and shifted every 10ms. Similar to [17], features for each frame are stacked with 3 frames to the left and then downsampled by three to a 30ms frame rate.

The encoder network follows [1], consisting of 8 LSTM layers, where each layer has 2,048 hidden units followed by a 640-dimensional projection layer. We insert a time-reduction layer (by a factor of 2) after the second encoder LSTM layer. This ensures that all encoder features (and subsequent decoder steps) occur at a 60ms frame rate. The RNN-T decoder contists of a prediction network and a joint network. The prediction network has 2 LSTM layers of 2,048 hidden units and a 640-dimensional projection per layer as well as an embedding layer of 128 units. The outputs of encoder and prediction network are fed to a 640 hidden unit joint network. The additional LAS-specific AE consists of 2 LSTM layers. The LAS decoder consists of multi-head attention [16] with four attention heads, computing context vectors which are fed into 2 LSTM layers of 2,048 hidden units with 640-dimensional projection. It has an embedding layer of 96 units. Both decoders are trained to predict 4,096 word pieces [12], which are derived from a large corpus of text.

The total size of the RNN-T model is 114M parameters and the second-pass LAS decoder is 33M parameters. All models are trained in Tensorflow [19] using the Lingvo [20] toolkit on a v2-128 Cloud TPU slice with a global batch size of 4,096.

The parameters $\alpha_{\text{early}}, \alpha_{\text{late}}, \beta$ in Equations 2, 4 and 6 are all tuned. We find 0.1 to be an optimal value. The behavior when changing the parameter t_{buffer} (i.e., the delay from the ground truth target t_{truth}) will be explored in Section 5.

4.3. Measuring Word Timings

As discussed in Section 3, start and end times of each word are estimated by looking at the time of the `<wb>` and last wordpiece or `</s>` token for each word, which can be calculated for LAS and RNN-T. These timings are then compared with word alignments obtained by running force alignment on the same data set with a conventional LFR model [2]. We then compare the word timings of matching transcripts from the ground truth and our model by looking at the differences in start time and end time for every matching word. We use these two separate measurements to allow us to differentiate whether the model is delaying the start or end of each word.

We measure the accuracy of word timings with 4 metrics. Average Start Time Delta (`Ave. ST` Δ) is the average start time change between the ground truth word time start and RNN-T word time start. Average End Time Delay (`Ave. ET` Δ) is similar but for the end time of the word. Percentage of Word

Start Times Less than 200ms (`% WS` $<$ `200ms`) is the % of word start times that are less than 200ms. Finally, percentage of Word End Times Less than 200ms (`% WE` $<$ `200ms`) is the same for word end times. Good word timing models typically have these last two metrics as close to 99% as possible.

5. Results

5.1. Inserting Word Boundary Token

Since generating word alignments requires us to train with a `<wb>` token to generate the start and end of each word, we first ensure that training with this token (without constraining alignment) does not degrade accuracy. Table 1 shows results with and without `<wb>` . Note that for the LAS rescoring models ($B1$ and $E1$), RNN-T is trained without `<wb>` . Instead, once hypotheses are generated from RNN-T, a known `<wb>` token is inserted between words to be rescored by the $E1$ LAS + `<wb>` model. The table shows that inserting a `<wb>` token does not degrade accuracy for both RNN-T and LAS. This is a first but important step in allowing us to train a model to emit word timings without possible degradations.

Table 1: *WER inserting* `<wb>` *token.*

ID	Model	WER
B0	RNN-T	23.2
E0	RNN-T + `<wb>`	23.2
B1	LAS	23.2
E1	LAS + `<wb>`	23.0

5.2. RNN-T Alignment

Next, we explore both WER and latency tradeoffs when we constrain RNN-T alignment. Table 2 shows WER and latency metrics as we vary t_{buffer}. Since the output rate of wordpieces is clocked at $60ms$, setting $t_{\text{buffer}} = 3$ results in a delay of 180ms from the ground truth time t_{truth}. The results show that a smaller buffer leads to more accurate word timing, but also more word errors.

Table 2: *Word Timing for RNN-T alignment*

ID	$E0$	$E2$	$E3$	$E4$
buffer time (ms)	none	180ms	360ms	540ms
WER	23.2	24.8	24.1	**22.8**
Ave. ST Δ	355.9	**169.7**	278.1	321.2
Ave. ET Δ	292.3	**173.2**	189.2	212.1
% WS $<$ 200ms	18.7	**86.7**	39.1	36.66
% WE $<$ 200ms	29.5	**70.8**	67.2	55

5.3. LAS Alignment

In this section, we analyze the WER versus latency tradeoff when we constrain one attention head with LAS rescoring. Table 3 shows these metrics as we vary the t_{buffer} time. The table shows that as the delay time increases, WER decreases but latency metrics increase. A delay of 180ms ($E6$) gives the best tradeoff between WER and latency. In addition, constraining alignment with LAS rescoring does not result in a quality degradation, like it does for RNN-T shown in Table 2. The $E6$ model has an average `Ave. ST/ET` Δ below 100ms, and the `% WS/WE` $<$ `200ms` is close to 99%.

Table 3: *Word Timing for LAS alignment*

ID	$E1$	$E5$	$E6$	$E7$
buffer time (ms)	none	60ms	180ms	300ms
WER	23.0	22.9	**22.7**	22.8
Ave. ST Δ	-	60.3	**54.8**	327.2
Ave. ET Δ	-	98.7	**92.7**	88.2
% WS $<$ 200ms	-	**99.5**	99.3	58.6
% WE $<$ 200ms	-	**99.3**	98.7	56.2

5.4. Analysis

5.4.1. Comparison With Conventional Model

We compare the latency and WER of LAS rescoring to a conventional model [2] in Table 4. The table indicates that LAS rescoring not only has an improved WER compared to the conventional model, but also has more accurate word timings. One hypothesis for this is that LAS is able to see the entire utterance and acts as a post-processing rescoring step. Therefore, seeing the entire utterance allows for more accuracy word timings compared to a conventional model which produces word timings in the 1st-pass.

Table 4: *Word Timing: LAS vs. Conventional*

ID	$B2$	$E6$
Model	Conv	LAS
WER	23.0	22.7
Ave. ST Δ	170.7	54.8
Ave. ET Δ	156.3	92.7
% WS $<$ 200ms	54.8	99.3
% WE $<$ 200ms	74.2	98.7

5.4.2. Attention Head

Figure 3 shows a plot of the constrained ground truth alignment $c(u, t)$ (red) and the constrained attention head (black), for a specific utterance. The figure shows that the attention head is quite well aligned with ground truth times. This shows us visually as well using the attention head to constrain the alignment does indeed result in accuracy word timings.

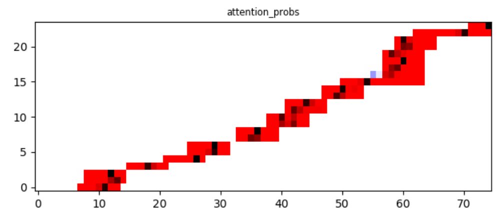

Figure 3: *Constrained Alignment (red) overlayed with Attention Head Probabilities (black)*

6. Conclusions

In this paper, we presented an approach to having end-to-end models emit word timings by constraining the attention head of a LAS 2nd-pass rescorer. We found that the word timings from this method were more accurate than a conventional model baseline, while not increasing error rate or result latency of our two-pass system.

7. Acknowledgements

Thank you to Antoine Bruguier, Rohit Prabhavalkar and Arun Narayanan for helpful discussions.

8. References

[1] T.N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu, I. McGraw, and C.C Chiu, "Two-Pass End-to-End Speech Recognition," in *Proc. Interspeech*, 2019.

[2] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[3] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S. Chang, W. Li, R. Alvarez, Z. Chen, C.C. Chiu, D. Garcia, A. Gruenstein, K. Hu, M. Jin, A. Kannan, Q. Liang, I. McGraw, C. Peyser, R. Prabhavalkar, G. Pundak, D. Rybach, Y. Shangguan, Y. Sheth, T. Strohman, M. Visontai, Y. Wu, Y. Zhang, and D. Zhao, "A Streaming On-Device End-to-End Model Surpassing Server-Side Conventional Model Quality and Latency," in *Proc. ICASSP*, 2020.

[4] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[5] C.C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-Art Speech Recognition With Sequence-to-Sequence Models," in *Proc. ICASSP*, 2018.

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

[7] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, Attend and Spell," *CoRR*, vol. abs/1508.01211, 2015.

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

[9] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *Proc. ICLR*, 2018.

[10] B. Li, S. Chang, T.N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards Fast and Accurate Streaming End-to-End ASR," in *Proc. ICASSP*.

[11] X. Li, G. Li, L. Liu, M. Meng, and S. Shi, "On the Word Alignment from Neural Machine Translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[12] Mike Schuster and Kaisuke Nakajima, "Japanese and Korean voice search," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[13] J. K. Chorowski and N. Jaitly, "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models," in *Proc. Interspeech*, 2017.

[14] D. Zhao, T.N. Sainath, D. Rybach, P. Rondon, D. Bhatia, B. Li, and R. Pang, "Shallow-Fusion End-to-End Contextual Biasing," in *Proc. Interspeech 2019*, 2019.

[15] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition," in *Proc. Interspeech*, 2015.

[16] 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.

[17] 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 *to appear in Proc. ASRU*, 2019.

[18] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated 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.

[19] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Available online: http://download.tensorflow.org/paper/whitepaper2015.pdf, 2015.

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