JOINT ENDPOINTING AND DECODING WITH END-TO-END MODELS

Shuo-Yiin Chang, Rohit Prabhavalkar, Yanzhang He, Tara N. Sainath, Gabor Simko

Google Inc., U.S.A

{shuoyiin, prabhavalkar, yanzhanghe, tsainath, gsimko,}@google.com

ABSTRACT

The tradeoff between word error rate (WER) and latency is very important for streaming automatic speech recognition (ASR) applications. We want the system to endpoint and close the microphone as quickly as possible, without degrading WER. Conventional ASR systems rely on a separately trained endpointing module, which interacts with the acoustic, pronunciation and language model (AM, PM, and LM) components, and can result in a higher WER or a larger latency. In going with the all-neural spirit of end-to-end (E2E) models, which fold the AM, PM and LM into a single neural network, in this work we look at folding the endpointer into this E2E model to assist with the endpointing task. We refer to this jointly optimized model – which performs both recognition and endpointing – as an E2E enpointer. On a large vocabulary Voice Search task, we show that the combination of such an E2E endpoiner with a conventional endpointer results in no quality degradation, while reducing latency by more than a factor of 2 compared to using a separate endpointer with the E2E model.

1. INTRODUCTION

The *endpointer* is an essential component that is responsible for determining when the user has finished speaking to ensure natural and fast voice interaction in streaming speech recognition applications such as voice assistant and voice search. It is desirable to close the microphone as soon as the user finishes speaking to minimize latency. However, it is also important to avoid cutting off users while they are still speaking. The errors in endpointing can have drastic impact on user experience: if the system waits too long to close the microphone, the user experience feels slow; if the system is too aggressive, the user will gets cutoff, which is also unsatisfactory.

A voice-activity detector (VAD) [1, 2, 3, 4], trained to classify each frame of audio as either speech or silence, has been widely used for endpointing. A conventional approach is to close the microphone as soon as a VAD system observes speech followed by a long silence interval. However, silence detection and endpointing are fundamentally different tasks, and the criterion used during VAD training may not be optimal. In particular, it ignores potential acoustic cues such as filler sounds, speaking rhythm or fundamental frequency to inform the decision of whether a human talker intends to continue speaking after a given pause. In [5, 6], an *end-of-query* (*EOQ*) classifier is trained to directly predict whether or not the user has finished speaking at a given time. Figure 1 (b) shows an example of the EOQ targets, namely *speech*, *initial silence*, *intermediate silence*, and *final silence*. The framewise posteriors of final silence are thresholded to obtain a hard microphone closing decision. As observed in [5], the EOQ classifier has shown around 100 ms latency improvement over VAD based endpointer at the same WER.

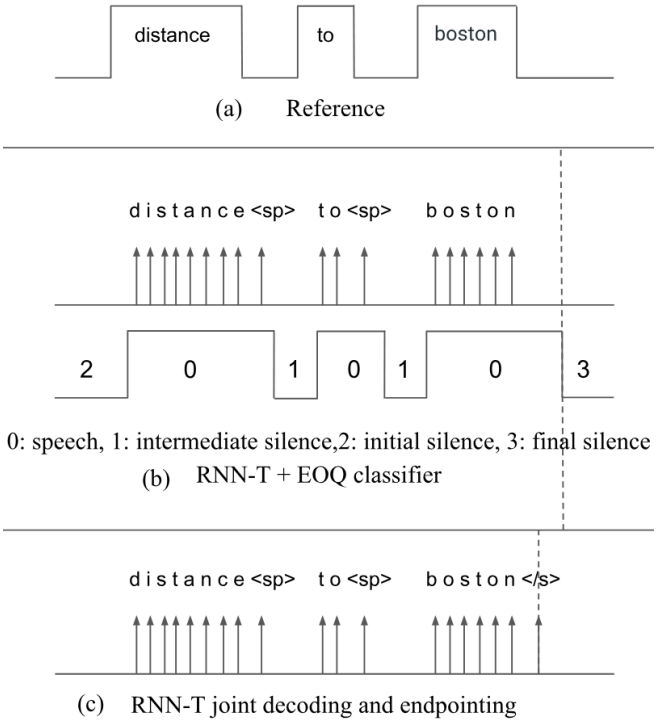


Fig. 1: *An illustration of endpointing for an example utterance (depicted in (a)). The end-of-query classifier (depicted in (b)) applies a separately trained model to the input acoustic frames. The RNN-T E2E endpointer (depicted in (c)), performs both decoding and endpointing.* $\langle sp \rangle$ *denotes the space symbol.* $\langle /s \rangle$ *denotes the end of utterance detection.*

Still, there are at least two drawbacks to using a VAD or EOQ-based endpointing system. First, both the VAD and EOQ classifiers make endpointing decisions based solely on acoustic information, while ignoring information from the language model ([7] explored decoder events as extra features to gain more information). Second, these classifiers are trained independently from the rest of the components of the ASR pipeline, namely the acoustic model, pronunciation model and language model. In this paper, we focus on an endpointing solution to address both of these issues by training an end-to-end (E2E) model to do decoding and endpointing jointly.

There has been a growing interest in building E2E models for ASR [8, 9, 10, 11, 12, 13, 14]. Such models replace the components of a traditional speech recognizer (the AM, PM, and LM) with a single neural network, thus simplifying the recognition pipeline. An

important component of many production ASR systems is that the ASR model must be streaming, that is, the model must emit decisions as the user is speaking and cannot wait until the end of the utterance to perform decoding. While many attention-based encoder-decoder architectures, including Listen, Attend and Spell (LAS) [8], have shown promising results [14], these models are not streaming. One such E2E model, the recurrent neural network transducer (RNN-T) [10, 11], emits decisions per frame and is streaming, and will thus be the model used in the present work.

Inference in E2E models is performed using beam-search. For some E2E models (e.g., LAS [8]) decoding terminates when the model outputs a special end-of-sentence token ⟨/s⟩. In models such as RNN-T, however, decoding is terminated when the last frame of the input has been processed. In this work, in order to enable joint decoding and endpointing with an RNN-T model, we look to train the RNN-T model with the end-of-sentence token, which allows us to terminate the beam search when ⟨/s⟩ has been output as shown in Figure 1 (c). This allows us to fold endpointing jointly into the E2E model, minimizing the dependence on an external VAD or EOQ. Since we have found conventional models achieve better latency with an EOQ rather than VAD [5], as a comparison we show results when an external EOQ system first filters frames, and then RNN-T terminates decoding on the last frame.

Our results are conducted on Voice Search task (over $10,000$ hours), where we find that at the optimal operating point, the combination of the EOQ and the E2E endpointer that jointly folding the endpointer into the RNN-T model provides a median latency of 230 ms compared to using only separate EOQ based endpointer, which has similar WER but a latency of 500 ms. This shows that by jointly training the E2E model to decode and endpointer we can halve the endpointing latency.

The rest of this paper is as follows. In Section 2, we describe the proposed neural network architecture. The experimental setup is described in Section 3, while the results are presented in Section 4. Finally, Section 5 concludes the paper.

2. NEURAL NETWORK ARCHITECTURE

2.1. RNN-T Model

Figure 2 illustrates the architecture for an *RNN-Transducer* (RNN-T) [10]. In the architecture, the encoder is analogous to an acoustic model that receives acoustic feature vectors $\mathbf{x}_t \in \mathbb{R}^d$, while the prediction network acts as a language model that accepts the previous grapheme label prediction y_{u-1} as input, and computes an output vector $\mathbf{p}_u$. For each combination of acoustic frame input t and label u, the encoder outputs $\mathbf{h}_t$ and the prediction outputs $\mathbf{p}_u$ are passed to a joint network to compute output logits and then fed in a softmax layer which defines a probability distribution over the set of output targets. Hence, the RNN-T is often described as an *end-to-end* model because it can be configured to directly output graphemes directly without the aid of an additional external language model.

The conditional probability distribution for RNN-T can be expressed as:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{y} \in \mathcal{A}(\mathbf{x},\mathbf{y})} \prod_{t=1}^{T} P(\hat{y}_t|\mathbf{x}_1, \cdots, \mathbf{x}_t, y_0, y_1, \ldots, y_{u(t-1)}) \tag{1}$$

where $\mathbf{x}_i$ is a feature vector, which is 80-dimensional log-Mel filterbank features for each frame $1 \ldots T$. We denote the ground-truth label sequence of length U as $y_1, y_2, \ldots, y_u$ where $y_u \in \mathcal{S}$ ($\mathcal{S}$ is the set of grapheme symbols). We use a special symbol, $y_0 = \langle \texttt{sos} \rangle$, which indicates the start of the sequence. For the convenience of formulation, we augment $\mathcal{S}$ with an additional *blank* symbol, $\langle \texttt{b} \rangle$, and describe the set of all possible alignments as follows: $\hat{\mathbf{y}} = (\hat{y}_1 \ldots \hat{y}_T) \in \mathcal{A}(\mathbf{x}, \mathbf{y})$, where $\mathcal{A}(\mathbf{x}, \mathbf{y})$ represents all label sequences $\hat{\mathbf{y}} \in \{\mathcal{S} \cup \langle \texttt{b} \rangle\}^T$ such that $\hat{y}$ is equal to $\mathbf{y}$ when $\langle \texttt{b} \rangle$ is removed. With this notation in place, the conditional probability of labeling given the acoustics $P(\mathbf{y}|\mathbf{x})$ then could be obtained by simply summing over the alignments.

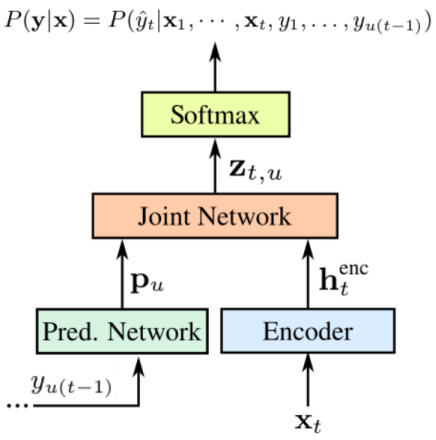


Fig. 2: *A schematic representation RNN-T.*

As shown in Equation 1, the probability of seeing some label in an alignment $\hat{y}_t$ is conditioned on the acoustic features up to time t and the history of non-blank labels, $y_1 \ldots y_{u(t-1)}$, emitted so far. The only independence assumption we have made is that the probability of a partial alignment $\hat{\mathbf{y}}_{1\ldots t}$ up to time t cannot depend on acoustic features from future frames. This enables us to perform inference in a streaming fashion, alleviating the need to wait for all of the audio before beginning the computation. This not only speeds up execution, but also permits us to emit partial recognition results as the audio is being processed, which enables joint endpointing as described in the next section.

2.2. RNN-T as an End-to-End Endpointer

To expand RNN-T with endpointing decisions, we incorporate a special symbol ⟨/s⟩ which indicates the end of the utterance as part of the expected label sequence. The RNN-T model then makes microphone closing decision when *top-beam* contains ⟨/s⟩. Therefore, the model acts jointly as a decoder and an endpointer.

However, mis-prediction of ⟨/s⟩ could account for much larger impact on quality than other symbols especially if ⟨/s⟩ prediction is too early. Hence, we precisely control posterior of ⟨/s⟩ in two different aspects when performing beam-search for decoding. First, the ⟨/s⟩ label is penalized with a positive scale α as shown in Eq 2. By controlling α, we directly modify the posterior of ⟨/s⟩ that competes with other symbols. If we set α greater than 1, extra penalty is added to ⟨/s⟩. In this case, the hypothesis including ⟨/s⟩ has higher cost than the others in the search beam, so it is less likely to present on top. Since declaring endpointing decision replies on *top* hypothesis, the modification makes endpointing decision less aggressive. On the contrary, using smaller α makes endpointing more aggressive and could hurt WER by introducing deletion errors.

Second, we expand the search space with ⟨/s⟩ only if the modified posterior is above a predefined threshold β to further reduce

early endpointing, i.e., ⟨/s⟩ is added to the search beam only if:

$$P(\langle/\text{s}\rangle \,|\mathbf{x}_1, \cdots, \mathbf{x}_t, y_0, \ldots, y_{u(t-1)})^{\alpha} \geq \beta \quad (2)$$

Sweeping β allows us to discard or allow the ⟨/s⟩ symbol when expanding the hypotheses during the search. Hence, β determines if ⟨/s⟩ symbol is allowed to present in the search beam while α affects the ordering for the hypothesis with ⟨/s⟩.

3. EXPERIMENTAL DETAILS

3.1. Data Sets

The training set used for the experiments consists of 20 million English utterances (over 10,000 hours). The training utterances are anonymized and hand-transcribed, and are representative of Googles 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 [15]. The noise sources are drawn from YouTube and daily life noisy environmental recordings. The main test set we report results on includes 14K anonymized, hand-transcribed voice search utterances extracted from Google traffic.

3.2. RNN-T Model

All experiments use the Recurrent Neural Network Transducer (RNN-T) model [10]. The RNN-T models use 80-dimensional log-Mel features with a frame step of 10 ms computed using a 25ms window. These features are stacked with 3 frames to the left and downsampled to a 30ms frame rate. The encoder network architecture consists of 8 long short-term memory LSTMs [16], where each layer has 2,048 hidden units followed by a 640-dimensional projection layer. The decoder is 2 LSTM layers with 2,000 hidden units and a 640-dimensional projection per layer. To stabilize training, we found it effective to put a layer-norm layer [17] after each LSTM layer in the encoder and decoder. The encoder and decoder are fed to a joint-network that has 640 hidden units. The joint network is fed to a softmax layer, with a total of 76 grapheme units. All RNN-T models are trained in Tensorflow [18] on 8×8 Tensor Processing Units (TPU) slices with a global batch size of 4,096.

3.3. EOQ based Endpointer

The baseline system consists of an EOQ detector for endpointing decision proposed in [5]. The input acoustic feature vector sequence consists of 40-dimensional log mel filterbanks with an upper limit of 4 kHz and a frame step of 10 ms using a 25 ms window. The EOQ classifier uses a convolutional, long short-term memory, deep neural network (CLDNN) (combination of convolutional, stacked LSTM and DNN layers) [19]. The features are passed into a frequency convolutional layer with a filter width of 8 frequency bands and pooling with stride 3, followed by a 64-node ReLU DNN layer, a sequence of two 64-cell LSTM layers, another 64-node ReLU DNN layer, and a 4-node softmax layer. LSTM-based architectures, and in particular CLDNNs, have previously been shown to work well for the EOQ classifier [5]. In the previous experiments we observed around 100 ms gains from using EOQ detector comparing to VAD based endpointer. Hence, we use EOQ classifier-based endpointer as our baseline.

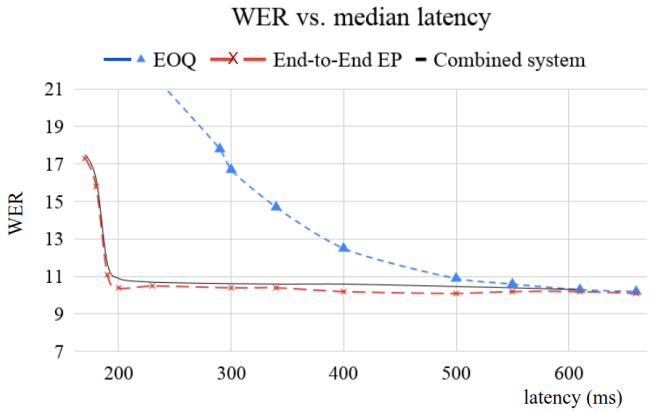


Fig. 3: *WER vs median latency for EOQ (blue),E2E endpointer (red) and combined system (black).*

4. RESULTS

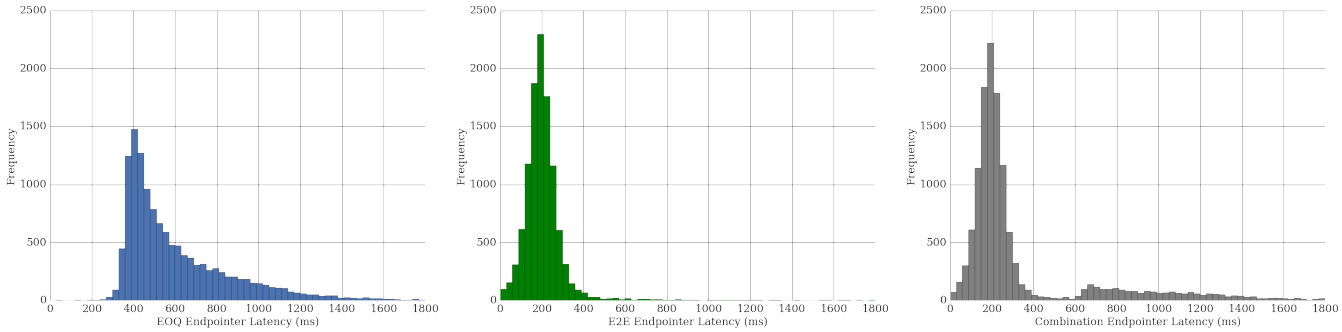
In this section we present the experimental results comparing (1) the EOQ, (2) E2E endpointer and (3) combined system that triggers an endpointing decision using *both* EOQ and E2E endpointer, based on which system triggers first.

The problem of endpointing is to find the best tradeoff between fast endpointing (measured by latency) and WER. An aggressive endpointer may provide a faster response at the expense of hurting WERs, while a passive endpointer may reduce WER but increase latency. Hence, we first report WER vs median latency in Figure 3 for each system.

For E2E endpointer, the tradeoff is produced by sweeping the scale α and the threshold β in Equation 2. A larger α means adding a larger penalty to ⟨/s⟩, which makes endpointing slower but also avoids deletions. Similarly, using larger β in Equation 2 avoids expanding ⟨/s⟩ in the search space, making the endpointing decision less aggressive. We sweep both parameters jointly to find best operating point.

Figure 3 shows that E2E endpointer has a better tradeoff between WER and latency, especially when the latency is close to 200 ms. As shown in Figure 3, E2E endpointer reduce the median latency down to 200 ms without significant WER degradation while the WER increases rapidly when latency is lower than that. We obtain $\alpha = 2.0$ and $\beta = 0.65$ in Equation 2 for the optimal point. In Figure 3, the combined system is dominated by E2E endpointer since it is much faster than EOQ endpointer.

In Table 1, we compare the final operating points for each system. For EOQ baseline, we sweep the endpointing decision threshold and select a WER of 10.9% such that the operating point of the EOQ system is consistent with our current production traffic [6]. In addition to WER and median latency, we also report the percentage of utterances that are actually endpointed and refer it as endpointing coverage. As shown in Table 1, at this operating point the E2E endpointer is better than the EOQ and reduces median latency from 500 ms to 200 ms (60% relative improvement). However, while E2E endpointer is better than EOQ for both latency and WER, it shows around 10% endpointing coverage loss. This indicates that when E2E endpointer decides to endpoint, it endpoints very quickly with much lower latency than the EOQ endpointer, but for a subset of utterances it has a chance of not endpointing at all. On the other hand, the combined system compensates for this coverage loss with



(a) *Histogram for EOQ endpointer latency.* (b) *Histogram for E2E endpointer latency.* (c) *Histogram for E2E EP + EOQ EP latency.*

Fig. 4: Endpointer latency histogram for individual systems.

the aid of EOQ system while still provide fast endpointing decisions from the E2E endpointer for the majority of the utterances. We still observe a 270 ms improvement (54% relative improvement) from the combined system without WER or endpoiting coverage degradation comparing to EOQ baseline. In the combined system, the E2E endpointer is the major endpointer that declares decision for 82% of the utterances while EOQ endpointer covers the reset of 16% of the utterances. Hence, the combined system can maintain both the high coverage from the EOQ endpointer and the low latency from the E2E endpointer.

In this next section, we analyze the latency improvements further.

	EOQ	E2E EP	E2E EP + EOQ
WER	10.9%	10.4%	10.9%
median latency	500 ms	200 ms	230 ms
endpointing coverage	97%	85%	98%

Table 1: WER, latency and endpointing coverage for EOQ, E2E endpointer and combined system.

4.1. Analysis

In this section, we analyze the latency differences of the three systems. First, Figure 4a shows the histogram for the EOQ endpointer latency on the utterances that have been actually endpointed. Only very few utterances have endpointer latency lower than 300ms. The latency is mainly distributed between 300ms and 1400ms. Figure 4b shows a similar histogram but for the E2E endpointer latency. In this case, almost all latency is between 0 to 400ms. In Figure 4c, we plot the histogram for the E2E plus EOQ endpointer. We could observe that the majority of the utterances are still within 400 ms latency by E2E endpointer so the system acts quickly. The long tail in Figure 4c is expected since EOQ handles small amount of corner cases with the latency distributed from 600 ms to 1400 ms.

5. CONCLUSIONS

In this work we incorporate an endpointer into a unified end-to-end RNN-Transducer by jointly training the model to decode and endpoint. By comparing to a separate endpointer, results and analyses show that joint optimization of the endpointer with the E2E model reduces latency from 500 ms to 200 ms at the cost of not endpointing in about 10% of the cases. The combination of an E2E endpointer and a separate EOQ model acts as fast as the E2E endpointer and compensates for the endpointing coverage degradation. The final system reduces latency by half without WER and endpointing coverage degradation.

6. REFERENCES

[1] R. Zazo, Tara N. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform cldnns for voice activity detection," in *Proc. Interspeech*, 2016.

[2] S. Thomas, G. Saon, M. V. Segbroeck, and S. Naranyanan, "Improvements to the IBM speech activity detection system for the darpa rats program," in *Proc. ICASSP*, 2015.

[3] M. Graciarena et al., "All for one: feature combination for highly channel-degraded speech activity detection," in *Proc. Interspeech*, 2013.

[4] S. Chang, B. Li, G. Simko, T. Sainath, A. Tripathi, A. Oord, and O. Vinyals., "Temporal modeling using dilated convolution and gating for voice-activity-detection," in *Proc. Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[5] G. Simko, M. Sahannon, S. Chang, and C. Parada, "Improved end-of-query detection for streaming speech recognition," in *Interspeech*, 2017.

[6] S. Chang, B. Li, G. Simko, Tara N. Sainath, and C. Parada, "Endpoint detection using grid long short-term memory networks for streaming speech recognition," in *Interspeech*, 2017.

[7] R. Mass, 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 *Proc. Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[8] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *CoRR*, vol. abs/1508.01211, 2015.

[9] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-End Attention-based Large Vocabulary Speech Recognition," in *Proc. ICASSP*, 2016.

[10] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012.

[11] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep neural networks," in *Proc. ICASSP*, 2012.

[12] 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.

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

[14] C. C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, N. Jaitly, B. Li, and J. Chorowski, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[15] 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.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[17] J. L. Ba, R. Kiros, and G. E. Hinton, "Layer Normalization," *CoRR*, vol. abs/1607.06450, 2016.

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

[19] Tara N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. ICASSP*, 2015.