INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

An Analysis of "Attention" in Sequence-to-Sequence Models

Rohit Prabhavalkar[1], *Tara N. Sainath*[1], *Bo Li*[1], *Kanishka Rao*[1], *Navdeep Jaitly*[2†]

[1]Google, Inc., U.S.A
[2]NVIDIA, U.S.A.

{prabhavalkar,tsainath,boboli,kanishkarao}@google.com, njaitly@nvidia.com

Abstract

In this paper, we conduct a detailed investigation of attention-based models for automatic speech recognition (ASR). First, we explore different types of attention, including "online" and "full-sequence" attention. Second, we explore different sub-word units to see how much of the end-to-end ASR process can reasonably be captured by an attention model. In experimental evaluations, we find that although attention is typically focused over a small region of the acoustics during each step of next label prediction, "full-sequence" attention outperforms "online" attention, although this gap can be significantly reduced by increasing the length of the segments over which attention is computed. Furthermore, we find that context-independent phonemes are a reasonable sub-word unit for attention models. When used in the second-pass to rescore N-best hypotheses, these models provide over a 10% relative improvement in word error rate.

1. Introduction

Sequence-to-sequence modeling with attention [1, 2, 3, 4, 5] has recently gained a lot of interest as a potential way to simplify the process of automatic speech recognition (ASR), by reducing the number of complicated modules present in state-of-the-art systems [6, 7]. In particular, these models do not make the same conditional independence assumptions as in standard hidden Markov model-based (HMM-based), hybrid systems [8].

Attention-based models are comprised of an *encoder*, which consists of multiple recurrent neural network (RNN) layers that model the acoustics, and a *decoder*, which consists of one or more RNN layers that predict the output sub-word sequence. An *attention* layer acts as the interface between the encoder and the decoder: it selects frames in the encoder representation that the decoder should attend to in order to predict the next sub-word unit. Thus, these systems fold the acoustic, pronunciation and language models (AM, PM, and LMs) of a traditional ASR system into a single model, rather than treating them as separate entities. While attention models have many attractive features, to date these models have not been shown to outperform a state-of-the-art frame-based system which uses separate AM, PM, and LM modules, on a large vocabulary continuous speech recognition (LVCSR) task (cf., [2, 9]).

Most attention models, e.g., Listen, Attend, and Spell (LAS) [2], compute attention over the entire sequence of encoder features, and are thus not time synchronous and are difficult to deploy in tasks which require streaming recognition. However, in previous work applying attention-based modeling to speech tasks [2], it has been observed that the attention vectors appears to focus on a very small number of local time frames for each predicted output unit. It is, thus, unclear whether conditioning on the full encoder representation is really required for speech tasks. Our first goal, therefore, is to examine the benefits of conditioning output sequence predictions on the entire acoustic sequence. In particular, in recent work, Jaitly et al. [4] proposed the Neural Transducer (NT) model, that limits the number of encoder frames that the model can attend to in predicting the next output unit. Specifically, the NT model has a parameter known as *block size*, which restricts attention to be computed over a small window of the encoder space, unlike the LAS model where attention is computed over the entire utterance. For a large value of the block size, NT appears more like LAS. However, for smaller block size values, the NT attends to smaller regions of the encoder input space and is able to run more time synchronously. We therefore compare these two attention modeling strategies, by varying the block size to investigate it's impact on performance. Concurrently, when analyzing how important attention is, we also compare different methodologies to compute attention. Specifically, we explore both *location-based* strategies ("tanh attention" [1]), which use the previous attention vector to influence the attention vector used to predict the next output symbol, as well as *content-based* strategies (dot-product attention [2]).

Our experiments are conducted on a ∼12,500 hour LVCSR task. We find that attention is beneficial, and that increasing the block size, i.e., tending towards full sequence attention as in the LAS model, improves performance. When we evaluate models in a second-pass rescoring framework, we find that attention-based models trained to predict context-independent phonemes (CIP) as sub-word units achieve more than a 10% relative improvement in word error rate (WER) over a state-of-the-art sequence-trained context-dependent phoneme (CDP) baseline, thus demonstrating the value of attention models.

The rest of this paper is organized as follows: In Section 2, we review various types of attention models, including LAS and NT. We present our experimental setup in Section 3, and discuss our results in Section 4. Finally, Section 5 concludes the paper.

2. Attention Models

Given an input sequence of frame-level features (e.g., log-mel-filterbank energies), $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$, and an output sequence of sub-word units (e.g., graphemes, or phonemes) $\mathbf{y} = \{y_1, y_2, \ldots y_N\}$, the goal of a speech recognition system is to model the distribution over output sequences conditioned on the input, $P(\mathbf{y}|\mathbf{x})$. Typically, in most ASR systems, the prediction of each sub-word unit, y_i, is treated as independent of the previous predictions, $\mathbf{y}_{<i}$. However, in attention modeling, the probability distribution of each sub-word unit is conditioned on the previous history of sub-word unit predictions, $\mathbf{y}_{<i}$, and the input signal, $\mathbf{x}$, as described in subsequent sections:

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, \mathbf{y}_{<i}) \tag{1}$$

[†] Work performed while at Google.

http://dx.doi.org/10.21437/Interspeech.2017-232

2.1. Full-Sequence Attention

Full-sequence attention models compute attention over the entire utterance [1, 2]. While there are a few variants of full-sequence attention models, which mainly differ in their method of computing attention, in this paper we will focus on the Listen, Attend, and Spell (LAS) model [2]. LAS consists of three modules: a *listener*, an *attender* and a *speller*, which together define a probability distribution over the next sub-word unit conditioned on the acoustics and the sequence of previous predictions.

The listener module is similar to a typical neural network acoustic model, and it takes the original signal, $\mathbf{x}$, and maps it into a higher level representation, $\mathbf{h} = \{h_1, h_2, \ldots, h_P\}$.[1]

$$\mathbf{h} = \text{Listen}(\mathbf{x}) \tag{2}$$

The goal of the attender and speller is to take the output of the listener (i.e., $\mathbf{h}$) and produce a probability distribution over sub-word units, as given by Equation 1. The attention module determines which encoder features in $\mathbf{h}$ should be attended to in order to predict the next output symbol, y_i. There are many different ways to compute attention, which we highlight in subsequent sections, but at a high-level, the goal of the attention module is to take the representation of the input generated by the listener, $\mathbf{h}$, and the previous state of the decoder, $\mathbf{s}_{i-1}$, and produce a fixed-dimensional context vector, $\mathbf{c}_i$, which extracts the relevant portions of $\mathbf{h}$ which are used to predict the next output label, y_i:

$$\mathbf{c}_i = \text{AttentionContext}(\mathbf{s}_i, \mathbf{h}) \tag{3}$$

The speller module first uses an RNN, which takes the context vector, $\mathbf{c}_i$, the previous state of the speller RNN, $\mathbf{s}_{i-1}$, and the previous prediction, y_{i-1}, and updates the state of the RNN, as follows:

$$\mathbf{s}_i = \text{RNN}(\mathbf{s}_{i-1}, y_{i-1}, \mathbf{c}_i) \tag{4}$$

Finally, the output of the RNN is passed to a Character Distribution module, which is an MLP with softmax outputs over the sub-word units, to produce a probability distribution over the current output prediction, y_i.

$$P(\mathbf{y}_i|\mathbf{x}, \mathbf{y}_{<i}) = \text{CharacterDistribution}(\mathbf{s}_i, \mathbf{c}_i) \tag{5}$$

2.2. Limited-Sequence Attention Models

One of the limitations of full-sequence attention models is that the output prediction at each step, i, is conditioned on the entire input acoustic sequence, $\mathbf{x}$, making it unsuitable for tasks which require streaming decoding results. The Neural Transducer (NT) model [4] is a limited-sequence attention model that addresses this issue by limiting attention to fixed-size blocks of the encoder space.

Given the input sequence, $\mathbf{x}$, of length T, and a block size of length W, the input sequence is divided equally into $B = \lceil \frac{T}{W} \rceil$ blocks of length W, except for the last block which might contain fewer than W frames. The NT model examines each block in turn, starting with the left-most block (i.e., the earliest frames). In this model, attention is only computed over the frames in each block. Within a block, the NT model produces a sequence of k outputs, $y_i, \ldots, y_{i+k}$; it is found to be useful to limit the maximum number of outputs that can be produced within a block to M symbols, so that $0 \leq k \leq M$. Once it has produced all of the required labels within a block, the model outputs an <epsilon> symbol, which signifies the end of block processing. The model then proceeds to compute attention over the next block, and so on, until all blocks have been processed. The <epsilon> symbol is analogous to the *blank* symbol in connectionist temporal classification (CTC) [10]. In particular, we note that a block must output a minimum of one symbol (<epsilon>), before proceeding to the next block.

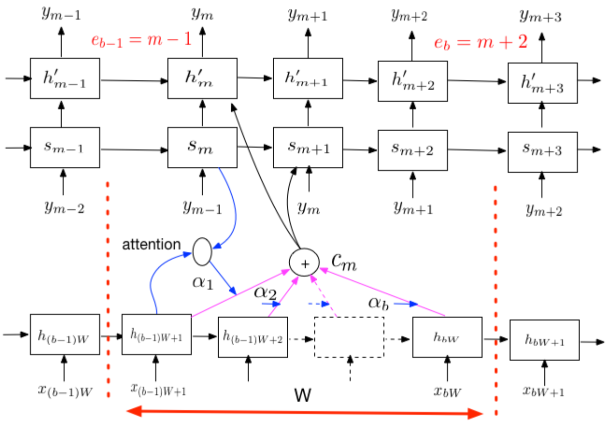

Figure 1: *Neural Transducer Attention Model.*

The model now computes $P(y_{1,\ldots,(N+B)}|\mathbf{x}_{1\ldots T})$, which outputs a sequence which is length B longer than the LAS model since the model must produce an <epsilon> at every block. Within each block $b \in B$, the model computes the following probability in Equation 6, where y_{e_b} = <epsilon> is the symbol at the end of each block. In other words, the prediction y_i at the current step, i, is based on the previous predictions $\mathbf{y}_{1\ldots e_{(i-1)}}$, similar to LAS, but in this case using acoustic evidence only up to the current block, $\mathbf{x}_{1\ldots bW}$:

$$P(\mathbf{y}_{(e_{b-1}+1)\ldots e_b}|\mathbf{x}_{1\ldots bW}, \mathbf{y}_{1\ldots e_{b-1}}) = \prod_{i=e_{(b-1)}+1}^{e_b} P(y_i|\mathbf{x}_{1\ldots bW}, \mathbf{y}_{1\ldots e_{(i-1)}}) \tag{6}$$

Thus, the listener module of the NT computes an embedding of the encoding vector only up to the current block:

$$\mathbf{h}_{1\ldots bW} = \text{Listen}(\mathbf{x}_{1\ldots bW}) \tag{7}$$

which is still implemented as a bidirectional RNN. The attention and speller modules operate similar to LAS, but only work on the partial output, $\mathbf{h}_{1\ldots bW}$, of the encoder up until the current block. The NT model is illustrated in Figure 1.

The NT allows us to control the block size to see how important attention is. At one end of the spectrum, for a large block size, NT behaves like LAS; at the other end of the spectrum, for small block sizes, it behaves like CTC, with the exception that NT is trained with a cross-entropy maximum-likelihood loss whereas CTC computes the loss via the forward-backward algorithm.

2.3. Computing Attention

For each decoder timestep, i, the attention vector determines where the model should attend to in the encoder sequence in order to predict, y_i. We highlight the two forms of attention explored in this paper below.

[1] In general, P may be different from T. In this work, $P < T$.

2.3.1. *Dot-Product Attention*

Dot-product attention is an example of a *content-based* attention strategy. In dot-product attention, the scalar energy, $e_{i,t}$, is computed for each frame $t \in T$ of the encoder, as a dot-product between the embedding of the decoder state, s_i, and an embedding of the encoder features, h_t, as given by Equation 8, where ϕ and ψ are MLP networks. An attention vector, α_i, is then created by passing the energy features through a softmax function to create a probability distribution, as shown in Equation 9. Finally, the attention vector is used to mask the encoder features, $\mathbf{h}_t$, and pass the relevant context vector, $\mathbf{c}_i$, to the decoder, as given in Equation 10:

$$e_{i,t} = \langle \phi(\mathbf{s}_i), \psi(\mathbf{h}_t) \rangle \tag{8}$$

$$\alpha_{i,t} = \frac{\exp(e_{i,t})}{\sum_{t'} \exp(e_{i,t'})} \tag{9}$$

$$\mathbf{c}_i = \sum_t \alpha_{i,t} \mathbf{h}_t \tag{10}$$

2.3.2. *Tanh Attention*

The goal of tanh attention is to make the attention vector *location aware*. First, as shown by Equation 11, the previous attention vector, α_{i-1}, is convolved with a matrix $F \in \Re^{k \times r}$ to produce k vectors, $\mathbf{f}_{i,t}$, for every position t of the previous vector. These additional vectors $\mathbf{f}_{i,t}$ are also passed through an MLP, θ, and used as an additional feature when computing the scalar energy, as shown in Equation 12.

$$\mathbf{f}_i = F * \alpha_{i-1} \tag{11}$$

$$e_{i,t} = \mathbf{w}^T \tanh(\phi(\mathbf{s}_{i-1}) + \psi(\mathbf{h}_t) + \theta(\mathbf{f}_{i,t})) \tag{12}$$

where, $\phi(\cdot)$, $\psi(\cdot)$, and $\theta(\cdot)$ are MLP networks. Once the scalar energies, $e_{i,t}$, have been computed, a context vector is computed using Equations 8 and 9, as before.

3. Experimental Details

Our experiments are conducted on a ~12,500 hour training set consisting of 15 million English utterances. The training 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. The noise sources are from YouTube and daily life noisy environmental recordings.

We report results on a set of ~13,000 anonymized, hand-transcribed utterances from the domain of open-ended dictation extracted from Google traffic. We also create a noisy version of this test set by adding noise drawn from the same distribution as training.

As we mentioned in Section 1, our goal in this work is to compare various attention modeling strategies. Since we examine phone-based sub-word units, decoding word sequences from these models is considerably more difficult than decoding grapheme-based models since the phoneme sequences must be combined with a lexicon during beam-search decoding (and possibly an LM). Further complexity is introduced by the fact that "online" attention models, such as the neural transducer implicitly consider various segmentations of the same output label sequence. Thus, in order to compare these various attention models we chose to avoid the complexity of the search process, by evaluating these models in a second pass N-best rescoring framework. For this purpose, we use an existing state-of-the-art speech recognition system to generate N-best lists which are rescored using the various attention models.

The baseline acoustic model consists of 5 layers of 700 unidirectional LSTM cells each, and is trained to output 8,192 context-dependent phonemes. The acoustic model is first trained to optimize the CTC loss function, following which it is sequence-trained to optimize the state-level minimum Bayes risk criterion [11]. The baseline model is decoded using a pruned 5-gram LM, which is then rescored using a much larger 5-gram LM. The baseline is decoded to produce N-best lists which are then rescored with the attention-based model: we linearly interpolate the baseline LM costs with a scaled negative log-likelihood from the attention-model to generate a new score which is used to rank hypotheses.

All experiments use 80-dimensional log-mel features, computed with a 25-ms window and shifted every 10ms. Similar to [12, 13], at the current frame, t, these features are stacked with 3 frames to the left and downsampled to a 30ms frame rate. The encoder network architecture consists of 5 bidirectional [14] long short-term memory [15] (BLSTM) layers with 700 cells per layer. We note that in all experiments, we always initialize the encoder using a pre-trained CTC model, since this was found to significantly improve convergence speed (see Section 4.3 for more details). The decoder consists of a single layer of 700 gated recurrent units (GRUs) [16]. Unless otherwise specified, dot product attention is used and no scheduled sampling [17] is used during training. In addition, unless otherwise specified, all networks are trained to predict context-independent phonemes. We report results obtained by rescoring up to 20 top word hypotheses since rescoring deeper N-best lists did not show a significant difference in performance. All neural networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) optimization [18] and are trained using TensorFlow [19].

4. Results

4.1. Changing Attention Window Size

Our first set of experiments explore the impact of the size of the attention window on performance, by comparing "online" attention over varying block sizes against full-sequence attention. Training NT systems is more complicated than an LAS system when the alignments of the output labels with respect to the input blocks is unknown [4]. Therefore, in order to simplify the training process, we compare the LAS and NT models with context-independent phonemes as output targets. The alignments required for training are generated by forced-alignment using a baseline system, which are assumed to be fixed during training. We also train a separate CTC system with the same configuration as the encoder used in the attention-based models (i.e., 5 layers of 700 BLSTM cells), and use this model to rescore the baseline, which serves as another comparison against the attention-based systems. This trained CTC model is used to initialize the encoder network in all attention-models, since this was found to significantly speed-up convergence (See Section 4.3).

In Figure 2, we plot an example of the attention matrix for an utterance trained using LAS which uses full-sequence attention. As can be seen in the figure, the attention matrix appears very localized, similar to findings in [2, 4, 5], which leads us to our first set of experiments to quantitatively understand how

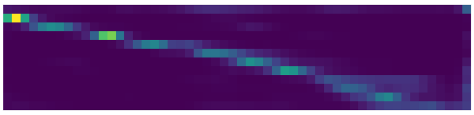

Figure 2: *Dot-product attention from LAS model. The x-axis represents input frames, and the y-axis corresponds to output labels (row i corresponds to the i-th label). Larger values appear yellow, and lower values appear purple.*

Table 1: *WERs obtained by rescoring the baseline with various models. The LAS model which uses full-sequence attention outperforms NT models, especially for small block sizes.*

Model	Block size	Clean	Noisy
CD-phoneme Baseline	-	6.9	9.6
+ CI-phoneme CTC	-	6.7	9.1
+ NT	2	7.2	10.0
+ NT	3	6.9	9.4
+ NT	5	6.6	9.1
+ NT	10	6.4	8.8
+ NT	20	6.3	8.7
+ LAS	full	**6.2**	**8.5**

changing the attention window effects WER.

Our results are presented in Table 1. As can be seen in the table, for small block sizes, the NT approaches the performance of CTC. However, as we increase the attention size, NT approaches the performance of LAS. We note here that all models use a bidirectional encoder which has access to the entire utterance; in spite of this, the results suggest that using larger attention windows improves performance, with the best performance obtained from a model that uses full-sequence attention. Finally, we note that the LAS system provides over a 10% relative improvement in WER as compared to the strong baseline system. To our knowledge, this is the first time that attention-models have demonstrated benefits over a state-of-the-art system.

4.2. Attention Type: "Dot-Product" v.s. "Tanh" Attention

We further examine the performance obtained from different techniques for computing attention. In Table 2, we compare "dot-product" and "tanh" attention, on an LAS model trained to predict CI-phonemes as the sub-word unit, where we find that "dot-product" attention appears to outperform "tanh" attention. We hypothesize that content-based approaches such as "dot-product" attention are sufficient for the utterances in our test set, which are generally relatively short. It would be interesting to investigate whether the same conclusion holds on a test set where the utterance lengths are much longer, which we leave as future work.

4.3. Attention Model Training and Sub-word Units

As we mentioned previously, we always initialize the encoder network in the attention-based models using a CTC-trained model. The use of this initialization was found to significantly speed up converegence, as illustrated in Figure 3.

Table 2: *WER obtained by rescoring a CI-phoneme LAS model with either "dot-product" or "tanh" attention.*

Model	attention type	clean	noisy
CD-phoneme Baseline	-	6.9	9.6
+ LAS	dot-product	6.2	8.5
+ LAS	tanh	6.4	8.8

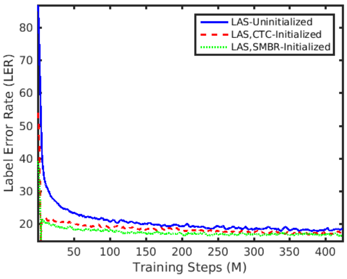

Figure 3: *LERs (%) on held-out test set for LAS model trained to predict CI-phonemes. Initialization from a previous CTC or sMBR trained model significantly speeds up convergence.*

Table 3: *WER obtained by rescoring baseline system using CTC or LAS model with CI or CD phonemes as sub-word units.*

Unit	Model Type	clean	noisy
CD-phoneme Baseline	-	6.9	9.6
+ CDP	CTC	6.4	8.9
+ CDP	LAS	6.5	9.1
+ CIP	CTC	6.7	9.1
+ CIP	LAS	**6.2**	**8.5**
Grapheme [20]	LAS (first-pass)	6.6	8.7

In the context of the full-sequence LAS models, we also examined whether performance is impacted by: the use of scheduled sampling during training [2] (we used a fixed 5% probability of sampling from the model, rather than ground-truth); initializing the encoder network with an sMBR trained model; or, using a model with additional layers in the decoder network. In each of these cases, performance was found to be similar to the LAS model with CI-phonemes as output targets, whose performance was reported in Table 1.

In our final set of experiments, we report results obtained using either CIP or CDP as sub-word units. We report rescoring results for both LAS and CTC models (i.e., a model with only an encoder). For comparison we also report results obtained using a grapheme-based LAS system which is directly decoded in the first-pass to obtain word sequences using a beam-search algorithm (but without rescoring with an LM) [20]. As can be seen in the Table 3, rescoring the baseline system with CI-phonemes achieves the best performance. It would be interesting to rescore with the grapheme LAS system, which we leave as future work.

5. Conclusions

In this paper, we conducted a detailed analysis of attention-based models in the context of ASR. In particular, we compared "full-sequence" attention models [2] against more recently proposed "online" attention models, where we found that the use of full-sequence attention achieves the best performance. We also conducted a number of experiments to determine the effect of different kinds of attention, and considered models trained with different sub-word units, other than graphemes that have been considered in previous work.

Overall, we find that rescoring a state-of-the-art baseline system can be improved by more than 10% relative using an LAS model trained to predict CI-phonemes.

6. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Proc. NIPS*, 2015.

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

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

[4] N. Jaitly, D. Sussillo, Q. V. Le, O. Vinyals, I. Sutskever, and S. Bengio, "An Online Sequence-to-sequence Model Using Partial Conditioning," in *Proc. NIPS*, 2016.

[5] Y. Zhang, W. Chan, and N. Jaitly, "Very Deep Convolutional Networks for End-to-End Speech Recognition," in *Proc. ICASSP*, 2017.

[6] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *Proc. Interspeech*, 2014.

[7] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in *Proc. ICASSP*, 2015.

[8] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer, 1994.

[9] L. Lu, X. Zhang, and S. Renals, "On Training the Recurrent Neural Network Encoder-Decoder for Large Vocabulary End-to-End Speech Recognition," in *Proc. ICASSP*, 2016.

[10] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labeling Unsegmented Seuqnece Data with Recurrent Neural Networks," in *Proc. ICML*, 2006.

[11] B. Kingsbury, "Lattice-Based Optimization of Sequence Classification Criteria for Neural-Network Acoustic Modeling," in *Proc. ICASSP*, 2009.

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

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

[14] M. Schuster and K. K. Paliwal, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," *Artificial Neural Networks: Formal Models and Their Applications-ICANN*, pp. 799–804, 2005.

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

[16] K. Cho, B. van Merriënboer, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proc. EMNLP*, 2014.

[17] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks," in *Proc. NIPS*, 2015, pp. 1171–1179.

[18] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.

[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] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A Comparison of Sequence-to-Sequence Models for Speech Recognition," 2017 (accepted).