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

Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping

Haşim Sak, Matt Shannon, Kanishka Rao, Françoise Beaufays

Google Inc., USA

hasim@google.com

Abstract

We introduce an encoder-decoder recurrent neural network model called Recurrent Neural Aligner (RNA) that can be used for sequence to sequence mapping tasks. Like connectionist temporal classification (CTC) models, RNA defines a probability distribution over target label sequences including blank labels corresponding to each time step in input. The probability of a label sequence is calculated by marginalizing over all possible blank label positions. Unlike CTC, RNA does not make a conditional independence assumption for label predictions; it uses the predicted label at time $t-1$ as an additional input to the recurrent model when predicting the label at time t. We apply this model to end-to-end speech recognition. RNA is capable of streaming recognition since the decoder does not employ attention mechanism. The model is trained on transcribed acoustic data to predict graphemes and no external language and pronunciation models are used for decoding. We employ an approximate dynamic programming method to optimize negative log likelihood, and a sampling-based sequence discriminative training technique to fine-tune the model to minimize expected word error rate. We show that the model achieves competitive accuracy without using an external language model nor doing beam search decoding.

Index Terms: recurrent neural network, acoustic modeling, end-to-end speech recognition, sequence training

1. Introduction

Recurrent neural networks (RNNs) have been successfully applied to many sequence modeling and sequence to sequence mapping tasks, such as language modeling, machine translation and speech recognition. Neural networks have been mostly used in combination with other models by replacing one component of the system with a neural network, such as acoustic models for speech recognition. More recently, there has been considerable amount of work to build a single neural network system for a task that can be trained end-to-end on the training data without using other prior information. For example, one such model for speech recognition is a neural network model trained to output grapheme or word sequences for a given utterance without using any external language or pronunciation lexicon.

Connectionist temporal classification (CTC) [1] training technique has been widely used for sequence to sequence mapping problems where alignment between input and target sequences is not known. CTC estimates a probability distribution over target label sequences while conditioning only on the input sequence and makes conditional independence assumption for label predictions. Therefore, CTC models require another sequence model that estimates a probability distribution over label sequences when predicting the next label in the sequence. The scores from these models are combined for decoding the most likely output sequence using beam search. RNN transducer [2, 3] has been proposed to address this problem of training two independent models. It combines the output of an RNN running over the input sequence with the output of another RNN running over label sequences using a feed-forward neural network with softmax output layer. RNN transducer has been used for end-to-end training of speech recognition models since it allows joint acoustic and language model training.

Recurrent neural network models within encoder-decoder framework with attention mechanism have been commonly used for machine translation [4, 5] and speech recognition [6, 7, 8, 9, 10, 11]. In this framework, a neural network (encoder) first encodes input sequence to a representation – commonly a sequence of network activation vectors. Then, a decoder network is conditioned on this representation and its previous predictions to generate the target label sequence. The decoder network commonly employs a soft attention mechanism over the encoded representation to focus the network on relevant information based on the partially predicted label sequence.

A segmental RNN model has been also proposed for speech recognition [12]. This model uses an RNN encoder for feature extraction combined with a segmental conditional random field (CRF) to define the sequence-level conditional probability of segment labels.

We have previously shown that CTC acoustic models with words as output units work very well as an end-to-end all-neural speech recognizer when we have large amount of transcribed acoustic training data [13, 14]. In this paper, we present a neural network model that can be trained end-to-end to optimize log-likelihood and expected losses and apply it to speech recognition.

2. Recurrent Neural Aligner

Recurrent Neural Aligner (RNA) is a neural network model within the encoder-decoder framework that can be trained end-to-end to map input sequences to target sequences. Given an input sequence of real-valued vectors of length T, $\mathbf{x} = (x_1, x_2, ..., x_T)$, RNA model tries to predict the target sequence of labels $\mathbf{y} = (y_1, y_2, ..., y_N)$ of length N. The model is a recurrent neural network model trained on such $(\mathbf{x}, \mathbf{y})$ sequence pairs. The input sequence can be raw feature vectors such as log-mel filterbank energy features for speech recognition task or they can be neural network encoded features. Since the model is a recurrent model processing input sequentially and we do not employ the attention mechanism to move the focus over the input, we allow the model to make null predictions by extending the output label space with a blank label as is done in CTC models. This allows the network to output blank labels if input sequence length T is larger than target sequence length N.

RNA defines a conditional distribution $P(\mathbf{z}|\mathbf{x})$ where $\mathbf{z} = (z_1, z_2, ..., z_T)$ is a sequence of labels of length T possibly

http://dx.doi.org/10.21437/Interspeech.2017-1705

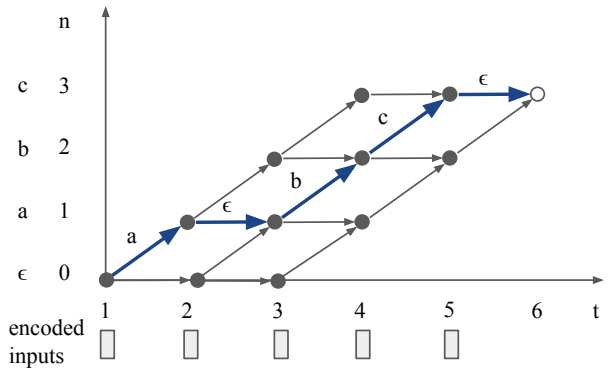

Figure 1: *Lattice showing all possible alignments between input and target sequences. The horizontal axis represents encoded features in RNA model. The vertical axis represents labels processed by RNA decoder. Each node (t, n) represents an RNA decoder state. The diagonal transitions represent the label probabilities as output from the model and horizontal transitions represent blank label probabilities. RNA decoder network runs over the paths in the lattice. The inputs are the encoded features and the transition labels encoded as one-hot vectors.*

with blank labels which when removed gives label sequence y. Therefore, $\mathbf{z}$ represents one of the possible alignments between input sequence $\mathbf{x}$ and label sequence $\mathbf{y}$. Then, we can marginalize over all possible alignments to estimate the distribution over target label sequence $\mathbf{y}$.

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) \quad (1)$$

RNA model has an encoder network which can be a unidirectional or bidirectional RNN, or any other neural network such as convolutional to encode raw input as input sequence $\mathbf{x}$. The decoder network is a recurrent neural network with a softmax output layer with size $L+1$ units where L is the number of labels in the output space and there is an additional unit for the blank label. Having a blank label enables RNA model to output a label (maybe blank) for each input vector. The input to the decoder network at time t for a given alignment z is the concatenation of input vector x_t and one-hot encoded label vector z_{t-1}. Therefore, the output layer estimates conditional distribution $P(\mathbf{z}|\mathbf{x}) = \prod_t P(z_t|z_1^{t-1}, \mathbf{x})$. This is where RNA model differs from CTC model which makes a conditional independence assumption for the distribution; $P(\mathbf{z}|\mathbf{x}) = \prod_t P(z_t|\mathbf{x})$.

For training RNA model, we use two different loss functions. We first train the model to minimize negative log-likelihood $\sum_{(\mathbf{x},\mathbf{y})} -\log(P(\mathbf{y}|\mathbf{x}))$ for all training example pairs $(\mathbf{x}, \mathbf{y})$. Since we marginalize over all possible alignments $\mathbf{z}$ corresponding to $\mathbf{y}$ and $P(\mathbf{z}|\mathbf{x})$ is calculated using a recurrent decoder network conditioned on labels in the alignment, it is not feasible to calculate this loss exactly. Therefore, we use an approximate forward-backward algorithm to estimate it. After this initial training, we further train the model to minimize expected loss for label predictions using a sampling-based technique.

2.1. Log-likelihood Loss

We can represent all possible alignments between the input sequence $\mathbf{x}$ and the target sequence $\mathbf{y}$ as a lattice as shown in Figure 1. In this work, since we focus on speech recognition, we assume that the length of input sequence is equal to or greater than the length of target sequence, i.e. $N \leq T$. Therefore, unlike RNN transducer model, we do not allow outputting multiple labels without processing one input vector. This reduces the amount of computation during training and makes beam search simpler during inference since RNA model will always process one input vector at each time step. The horizontal axis represents the encoded input sequence $\mathbf{x}$ and shows the input vector x_t for RNA decoder network at each time step t for $1 \leq t \leq T$. The vertical axis represents the target label sequence $\mathbf{y}$ starting at index 1. The diagonal transitions represent the output probabilities for the labels, while the horizontal transitions represent the probabilities for the blank labels (shown as ϵ). The first label y_0 is assumed to be blank label. Each path through this lattice starting from the initial node at $t = 1$ and $n = 0$ to the final node at $t = T + 1$ and $n = N$ represents a possible alignment $\mathbf{z}$.

To calculate the log-likelihood of target label sequences, we need to sum over all the path probabilities represented in the lattice. However, this is not feasible since there are exponential number of paths and we need to run the RNA decoder for each path. Therefore, we use an approximate forward-backward calculation. We assume that each node in the lattice represents a different state of RNA decoder and we do a forward calculation with input pair (x_t, y_{n+1}) and state at (t, n) to calculate the new state at $(t + 1, n + 1)$ and with input pair (x_t, ϵ) to calculate the state at $(t+1, n)$. The output of the decoder is the softmax activations for all the labels and it defines the transition probabilities in the lattice. The horizontal transitions represent blank label predictions and allow the decoder to delay the label predictions in the target sequence while processing more input vectors. The diagonal transitions represent the probability for observing the next label at $n + 1$ in the target sequence. The forward calculation of the RNA decoder results in an updated state of the decoder. We propagate this new state to the node at $(t+1, n+1)$ with the next label prediction and $(t+1, n)$ with the blank label prediction.

However, we can have two distinct RNA states coming to a node since we can have two different paths merging at a node. There has been previous work on merging or combining recurrent neural network states for multi-dimensional RNN [15] and Grid LSTM [16]. However, these approaches are not computationally efficient for inference in some applications such as speech recognition. In inference, we ideally would like to avoid combining RNN states corresponding to different alignments.

We solve this problem by choosing one of the RNA decoder states when two paths merge at a node in the lattice. To decide which RNA state to keep, we calculate the forward variable $\alpha(t, n)$ defined as the probability of outputting labels $(y_1, y_2, ..., y_n)$ up to time t. Let $p(y_n|t-1, n-1)$ be the probability of label y_n calculated by the softmax layer in the RNA decoder using the network state at node $(t-1, n-1)$ and $p(\epsilon|t-1, n-1)$ be the probability of blank label similarly. If $\alpha(t-1, n-1)+p(y_n|t-1, n-1) > \alpha(t-1, n)+p(\epsilon|t-1, n)$, we keep the RNA state coming from the $(t-1, n-1)$ node. Otherwise, we keep the state from the $(t-1, n)$ node. Therefore, the forward variables can be calculated recursively as follows using the RNA decoder state at each node;

$$\alpha(t, n) = \alpha(t-1, n-1)p(y_n|t-1, n-1) + \alpha(t-1, n)p(\epsilon|t-1, n) \quad (2)$$

With this approximation and using the forward-backward algorithm, we can efficiently calculate the log-likelihood loss

for target label sequences. We also define the backward variable $\beta(t,n)$ as the probability of outputting the labels $(y_{n+1}, y_{n+2}, ..., y_N)$ starting from time t;

$$\beta(t,n) = \beta(t+1,n+1)p(y_{n+1}|t,n) + \beta(t+1,n)p(\epsilon|t,n) \quad (3)$$

Then, the probability for the target label sequence $\mathbf{y}$ can be calculated by summing the path probabilities through the lattice which is equal to $\beta(0,0)$ variable:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}) = \beta(0,0) \quad (4)$$

If the negative log-likelihood loss for one training example $(\mathbf{y}, \mathbf{x})$ is $L = -\log(P(\mathbf{y}|\mathbf{x}))$, then it can be shown that the gradient of the loss with respect to sofmax activations from the RNA decoder is as follows:

$$\frac{\partial L}{\partial p(l|t,n)} = -\frac{\alpha(t,n)}{\beta(0,0)} \begin{cases} \beta(t+1,n+1) \text{ if } l = y_{n+1} \\ \beta(t+1,u) \text{ if } l = \epsilon \\ 0 \text{ otherwise} \end{cases} \quad (5)$$

Note that in inference time, we do not need to run RNA decoder model on all possible alignments. We can do beam search decoding using the probability distribution estimated from the model at each time step. In the beam search, each hypothesis will have a different RNA decoder state which will expand to new states with each label prediction chosen in search.

2.2. Expected Loss

The likelihood loss optimizes probability of target label sequences estimated from RNA model. However, when we apply the model to a task we commonly measure the performance of the model with a sequence level loss or metric, such word error rate (WER) for speech recognition. We ideally want the model to assign higher probabilities to label sequences having smaller sequence level loss. RNA model trained with likelihood loss also suffers from the exposure bias [17]. In training time, the probability of target label sequence is calculated by summing over all possible alignments, however, RNA decoder network is always conditioned on the true labels from the target sequence. Therefore, the model is exposed to only correct labels and may not learn to be robust to the errors in its label predictions when used for inference with beam search. Another disadvantage is the approximation that we had to use when calculating the path probabilities in the lattice when pooling RNA decoder states.

These problems can be addressed with a sequence level discriminative loss function – expected loss for the distribution over alignments given input sequence x from RNA model:

$$L = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x})\text{loss}(\mathbf{x}, \mathbf{z}, \mathbf{y}) \quad (6)$$

where $P(\mathbf{z}|\mathbf{x})$ is the probability of an alignment $\mathbf{z}$ as estimated from RNA model and $\text{loss}(\mathbf{x}, \mathbf{z}, \mathbf{y})$ is the calculated sequence level loss, such as the edit distance between the reference label sequence $\mathbf{y}$ and $\mathbf{z}$ after removing the blank labels.

We use Monte-Carlo sampling for approximate calculation of gradient [18, 17] as described in [19].

$$\frac{\partial L}{\partial p(l|t)} \approx \frac{1}{N} \frac{\partial \log(P(\mathbf{z_i}|\mathbf{x}))}{\partial p(l|t)} (\text{loss}(\mathbf{x}, \mathbf{z_i}, \mathbf{y}) - \overline{\text{loss}}(\mathbf{x}, \mathbf{z}_*, \mathbf{y})) \quad (7)$$

$$= \frac{1}{N} \frac{1}{p(l|t)} (\text{loss}(\mathbf{x}, \mathbf{z_i}, \mathbf{y}) - \overline{\text{loss}}(\mathbf{x}, \mathbf{z}_*, \mathbf{y})) \quad (8)$$

where $\mathbf{z_i}$ is an alignment label sequence obtained by sampling the labels from the output distribution of the model at each time step of input, N is the sample size, $p(l,t)$ is the probability of the label l at time step t in the sample z_i, $\overline{\text{loss}}(\mathbf{x}, \mathbf{z}_*, \mathbf{y})$ is the average loss of samples, i.e. $\frac{1}{N}\sum_{i=1}^{N} \text{loss}(\mathbf{x}, \mathbf{z_i}, \mathbf{y})$.

3. Experimental Results

In the speech recognition experiments, we used LSTM RNNs [20] for all encoder and decoder networks. The encoder networks are multi-layer LSTMs. For the decoder networks, we only experimented with a single layer LSTM. We first compare the performance of bidirectional CTC and RNA models on YouTube video transcription task [14]. The training set consists of 125,000 hours of audio data with semi-supervised transcripts from user-uploaded captions [14]. It has 1.2 billion words with a vocabulary size of 1.7 million. The test set is 25 hours and has around 250,000 words. For the experiments where we use a language model for decoding, we use a 5-gram language model with $30M$ n-grams with a vocabulary of 500,000 words.

In Table 1, we compare RNA grapheme model with CTC context-dependent (CD) phone and CTC word models described in this paper [14].. CTC CD phone model is a bidirectional LSTM RNN model with 7 layers of 1000 units, 500 for forward and 500 for backward units in each layer. It has a softmax layer with output size of 6400 units for CD phones. CTC word model has been initialized with CTC CD phone model for LSTM layers while the softmax layer parameters have been initialized randomly. The output layer has close to 100K units for the word vocabulary. RNA grapheme model uses the CTC CD phone LSTM layers as the encoder model. The decoder model is a single layer LSTM with 400 units and a softmax layer of size 50 for graphemes. CTC CD phone model is used with a finite-state transducer decoder and an n-gram language model is used for decoding. CTC word model is an end-to-end trained neural speech recognizer outputting the words and can be used without doing any beam search. However, we can get a small WER improvement from 13.6% to 13.1% when we generate a lattice using the model predictions and rescore with the n-gram language model. RNA grapheme model is used for end-to-end speech recognition without any other external language or pronunciation model. The model was trained using the log-likelihood loss, and sampling based sequence level loss did not improve results for this data. We speculate that eMBR training does not improve model since there are not many errors the model makes for the training data. For this model, we found that beam search did not improve accuracy of the model. We do a greedy search where we pick the most probable label predicted at each time step and use that label as input for the next prediction. RNA model gives about the same accuracy (13.1%) with the CTC word model with language model rescoring. These models are significantly better than the conventional CTC CD phone model.

In Table 2, we compare RNA grapheme models with CTC context-dependent (CD) phone model on mobile dictation task. The training set consists of 15 million anonymized mobile voice

Table 1: *RNA grapheme model compared with CTC CD phone and word models on YouTube video transcription task. We report the WERs before and after sMBR sequence training for the CTC model. We did not get improvements for other models with sequence-level loss.*

Model	Layers (Encoder + Decoder)	Outputs	Params	Vocab	OOV(%)	WER(%) w/ LM	WER(%) w/o LM
CTC CD phone	7x1000	6400	43m	500000	0.24	14.2 / 13.6	—
CTC word	7x1000	97827	137m	97827	0.70	13.0	13.6
RNA grapheme	7x1000 + 1x400	50	42m	unlimited	0	—	13.1

Table 2: *RNA grapheme models compared with CTC CD phone models on mobile dictation task. WER numbers for CTC models are after sMBR sequence training. WER numbers for RNA models are given for greedy and beam search. The first number is after training with log-likelihood loss and the second number is after further training with expected WER loss.*

Model	Bidirectional	Layers (Encoder + Decoder)	Outputs	Params	Vocab	WER(%) Greedy	WER(%) Beam
CTC CD phone	No	5x600	8192	19m	4,000,000	—	— / 6.1
CTC CD phone	Yes	5x700	8192	19m	4,000,000	—	— / 5.4
RNA grapheme	No	8x700 + 1x400	50	32m	unlimited	9.9 / 8.5	9.0 / 8.4
RNA grapheme	Yes	5x700 + 1x500	50	16m	unlimited	7.4 / 6.5	7.0 / 6.4

search and dictation utterances. The test set contains about 13,000 utterances. The unidirectional CTC CD phone model is an LSTM RNN model with 5 layers of 600 units with a softmax output layer of 8192 units for CD phones. The bidirectional CTC CD phone model is an LSTM RNN model with 5 layers of 700 units – 350 for forward and 350 for backward layers. CTC phone models are decoded with a pruned 5-gram language model of 100 million n-grams. The encoder LSTM layers of RNA grapheme models have been initialized with pretrained CTC grapheme models. The unidirectional RNA model has 8 LSTM encoder layers each with 700 units and an LSTM decoder layer of 400 units while the bidirectional one has 5 encoder layers with 700 units and a decoder layer of 500 units. For RNA models, we report the WERs for greedy and beam search (10 best hypothesis) after training with log-likelihood loss and then after additional training with expected WER loss. No external language model is used in beam search. RNA models perform significantly worse than CTC CD phone models on this task. Bidirectional RNA model significantly improves over the unidirectional one, however it is still not as good as the bidirectional CTC model. The relatively smaller training set and larger vocabulary for the dictation task than the YouTube transcription make it more challenging for RNA grapheme models due to data sparsity problem. However, bidirectional RNA grapheme model after training with expected WER loss performs quite well as an end-to-end speech recognition system without even requiring beam search.

4. Conclusions

We presented RNA model which is a recurrent neural network model in the encoder-decoder framework. We applied it to end-to-end speech recognition and showed initial experimental results on YouTube transcription and mobile dictation tasks. We showed a RNA grapheme model with greedy search and no external language model can match the accuracy of a CTC word model which uses a word n-gram language model for rescoring when we have large amount of acoustic training data with transcripts available. We found that RNA grapheme model can not perform as well as CTC CD phone models for mobile dictation task where the vocabulary size is larger and the training data is relatively smaller. This can be expected since the grapheme models suffer more from the data sparsity problem than phone models and word level language model trained on much larger text corpus than available training data transcripts provides significant improvements for conventional models. We also showed RNA models trained with sampling based expected loss after initial training with likelihood loss can improve the model accuracy for dictation task. It also solves the exposure bias problem and accuracy of greedy search almost matches accuracy of beam search. With RNA model and these training techniques, we build an end-to-end trained neural speech recognition model which gives competitive accuracy with no decoding and external model.

As a future work, we would like to compare the performance of RNA model with other neural network models such as recurrent neural network transducer and attention-based encoder-decoder models. We also plan to train RNA phone models and experiment with rescoring with word level language models.

5. References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[2] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[3] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*. IEEE, 2013, pp. 6645–6649.

[4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[6] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[7] L. Lu, X. Zhang, K. Cho, and S. Renals, "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition." in *INTERSPEECH*, 2015, pp. 3249–3253.

[8] N. Jaitly, D. Sussillo, Q. V. Le, O. Vinyals, I. Sutskever, and S. Bengio, "A neural transducer," *arXiv preprint arXiv:1511.04868*, 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] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016.

[11] L. Lu, X.-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.

[12] L. Lu, L.-P. Kong, C. Dyer, N. A. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Proc. Interspeech*, 2016.

[13] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[14] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," *CoRR*, vol. abs/1610.09975, 2016. [Online]. Available: http://arxiv.org/abs/1610.09975

[15] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *Proc. ICANN*, 2007, pp. 549–558.

[16] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," *arXiv preprint arXiv:1507.01526*, 2015.

[17] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[18] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks." in *Proc. ICML*, vol. 14, 2014, pp. 1764–1772.

[19] M. Shannon, "Optimizing expected word error rate via sampling for speech recognition," in *Proc. Interspeech*, 2017, in review.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.