Acoustic-to-Phrase Models for Speech Recognition

Yashesh Gaur, Jinyu Li, Zhong Meng, Yifan Gong

Microsoft Corporation, one Microsoft way, Redmond, WA 98052

{yagaur, jinyli, zhme, yifan.gong}@microsoft.com

Abstract

Directly emitting words and sub-words from speech spectrogram has been shown to produce good results using end-to-end (E2E) trained models. Connectionist Temporal Classification (CTC) and Sequence-to-Sequence attention (Seq2Seq) models have both shown better success when directly targeting words or sub-words. In this work, we ask the question: Can an E2E model go beyond words and transcribe directly to phrases (i.e., a group of words)? Directly modeling frequent phrases might be better than modeling its constituent words. Also, emitting multiple words together might speed up inference in models like Seq2Seq where decoding is inherently sequential. To answer this, we undertake a study on a 3400-hour Microsoft Cortana voice assistant task. We present a side-by-side comparison for CTC and Seq2Seq models that have been trained to target a variety of tokens including letters, sub-words, words and phrases. We show that an E2E model can indeed transcribe directly to phrases. We see that while CTC has difficulty in accurately modeling phrases, a more powerful model like Seq2Seq can effortlessly target phrases that are up to 4 words long, with only a reasonable degradation in the final word error rate.

Index Terms: speech recognition, sequence-to-sequence, connectionist temporal classification, end-to-end training

1. Introduction

End-to-end trained (E2E) models for automatic speech recognition (ASR) are no longer just a promising research direction. They have begun to replace conventional hybrid systems [1] and have found their way into production services [2]. Connectionist Temporal Classification (CTC) and Sequence-to-Sequence attention (Seq2Seq) models are among the most popular E2E ASR solutions today. The flexibility afforded by E2E models has inspired people to train these models to emit a variety of tokens, including phones [3, 4], characters [5, 6], sub-words and words [7] [8]. For both CTC and Seq2Seq, better results were obtained as we moved from smaller tokens like characters to larger word-level units like word-pieces [7] and mixed-units [8]. A natural next step might be to see if E2E models can emit even bigger output units, for e.g. a phrase. A system that can emit a group of words as one token might do a better job of jointly learning how the constituent words are spoken together. The reduced output sequence length might also speed up training/inference. Moreover, training these models to target phrases would be a good way to test the limits of their modeling.

In this work, we try to investigate this by constructing 3 different phrase-based vocabularies and training both Seq2Seq and CTC with them. Each vocabulary exposes the E2E models to progressively longer (hence potentially more difficult) phrases. In addition to phrases, the model also has to learn to fall back to emitting words and sub-words in case frequent phrases can not be predicted. The intention is to check if and when E2E models can learn to accurately emit phrases. We present the results for phrase-based vocabularies in context of 4 strong baselines that were trained on a variety of tokens from the literature. The "baseline tokens" consist of single characters, words, word-pieces and mixed units. We present a side-by-side comparison for Seq2Seq and CTC for these 7 tokens. The results show that both CTC and Seq2Seq can learn to emit phrases. However, CTC's final word error rate (WER) increases drastically when trained with phrase-based vocabularies. Seq2Seq, on the other hand, can effectively emit phrase tokens that are as long as 4 words. The final WER for Seq2Seq models only showed a minor degradation. This study works as a proof of concept and allows us to launch future investigations into constructing phrase-based vocabularies that might benefit from potentially better modeling and faster training/decoding speeds. This paper has the following contributions. To the best of our knowledge, this is the first work to target units that are larger than words. Secondly, we present an extensive side-by-side empirical comparison between CTC and Seq2Seq across a variety of output tokens, ranging from single characters to phrases. Lastly, we show that Seq2Seq is strong enough to accurately target phrases as output units. CTC, on the other hand, falls short on this task.

2. End-to-End models for ASR

2.1. Connectionist Temporal Classification

A CTC model is usually a Recurrent Neural Network (RNN) [9] that is trained using the CTC loss function [3], that directly optimizes prediction of the transcription sequence. Lets denote $\mathbf{x}$ as the speech input frame sequence and $\mathbf{y}$ as the original label sequence. As the length of $\mathbf{y}$ is shorter than the length of $\mathbf{x}$, a CTC path $\boldsymbol{\pi}$ is introduced to have the same length as $\mathbf{x}$ by adding the blank symbol as an additional label and allowing repetition of labels. A CTC path can thus be collapsed to give back $\mathbf{y}$ by simply collapsing repeating tokens and removing blank symbols. Let's call this operation B. The CTC loss function is then defined as follows:

$$L_{CTC} = -\ln P(\mathbf{y}|\mathbf{x}) = -\ln \sum_{\boldsymbol{\pi} \in B^{-1}(\mathbf{y})} P(\boldsymbol{\pi}|\mathbf{x}). \quad (1)$$

To calculate $P(\boldsymbol{\pi}|\mathbf{x})$, CTC assumes conditional independence between prediction at various output time-steps and decomposes it into product of posteriors from each frame as:

$$P(\boldsymbol{\pi}|\mathbf{x}) = \prod_{t=1}^{T} P(\pi_t|\mathbf{x}). \quad (2)$$

This makes the computation tractable but also does not allow the RNN to learn any dependencies between the output tokens. To decode CTC models, we can greedily pick the token corresponding to the posterior spikes to form the output sequence. This would neither require a language model nor any complex beam search decoding procedure. For the sake of simplicity, we will be using this greedy decoding for all our CTC models in this work.

2.2. Sequence-to-Sequence models

Seq2Seq models are an efficient way to combine the acoustic, language and pronunciation models into one single neural network. Many Seq2Seq architectures have been proposed, including Recurrent Neural Network Transducer [10], Listen Attend and Spell [5] and Recurrent Neural Aligner [11]. In this work we have to chosen to work with an Encoder-Decoder architecture very similar to [4]. The Encoder in our case is an RNN that takes in the input features, $\mathbf{x}$, and maps them to a higher level feature representation $\mathbf{h^{enc}}$. The decoder is also an RNN, which acts like a conditional LM and tries to compute:

$$P(\mathbf{y}|\mathbf{x}) = P(\mathbf{y}|\mathbf{h^{enc}}) = \prod_{t=1}^{T} P(y_t|y_0, \ldots, y_{t-1}, \mathbf{h^{enc}}) \quad (3)$$

where $\mathbf{y}$ is the output label sequence. The dependence of y_t on $\mathbf{h^{enc}}$ is captured by conditioning the decoder-RNN on the so-called context-vector (c_t), which is just a weighted combination of encoder hidden states. c_t tells the decoder which input frames it should "focus on" to emit the next token. To calculate c_t, the decoder RNN is augmented with a location-aware attention mechanism similar to [4]. The network is optimized by minimizing the Cross-Entropy (CE) loss between the output of the decoder and references at all time-steps. We refer the reader to [4] for further details on our implementation.

3. Output tokens in E2E ASR

3.1. Characters

Some of the earliest work with CTC and Seq2Seq had phones as targets [4, 12]. Later, these models were also shown to be successful with characters as targets [5, 6]. Directly targeting characters allowed us to get rid of the phonetic lexicon, thereby making the decoding process simpler. In addition, the models could also overcome the losses due to imperfect or incomplete pronunciations in the phonetic lexicon, if any.

3.2. Words

CTC suffers from the Conditional Independence (CI) assumption (Section 2.1) which leads to poor results when decoding greedily. One way to partially mitigate the CI assumption is to avoid learning the dependency between characters altogether and directly targeting words. [13, 14] showed that words as output units would be a natural choice for CTC. Seq2Seq models have also been trained to directly target words [15]. However, the word-based models suffer from the out-of-vocabulary (OOV) issue as they can only model a limited number of words in the output layer. Several solutions have been proposed to mitigate the OOV issue [16, 8, 17]. A popular approach to address the OOV issue is to predict the rare and unknown words as sequences of words and/or subwords. Two examples of this approach are word pieces [7] and mixed units [8].

3.3. Word Pieces (Byte-Pair Encoding)

Sub-words as targets in E2E modeling were first proposed in [7]. Inspired by the Byte-Pair Encoding algorithm, which is used for data compression, it described a simple algorithm that starts from characters and iteratively collapses tokens to form subwords. This vocabulary of sub-words can be used to segment words in a deterministic fashion. Word-pieces have already demonstrated superior performance for E2E ASR models [17, 18, 2].

3.4. Mixed units

In contrast to word pieces algorithm, which builds up the vocabulary in a bottom-up manner, mixed-units [8] preparation works in a top-down fashion. We start by selecting all the words that occur greater than a certain number of times, in the training set. Rest of the words make the OOV set. A set of character n-grams (n=1,2,3) is then constructed so that all the OOV words can be decomposed as a sequence of character n-grams and/or frequent words. The final vocabulary is the union of frequent words and character n-grams (sub-words).

3.5. Phrases

Going beyond words and trying to target phrases as output units might be worth exploring for a few reasons. Firstly, in some cases, modeling frequent phrases might be better than modeling its constituent words as it might allow the model to jointly learn how its constituent words are spoken together. Secondly, segmenting the text into phrases will reduce the length of output sequence that needs to be targeted. A much shorter output sequence can lead to quicker loss computation and might result in faster training times for both CTC and Seq2Seq models. Moreover, in Seq2Seq models, where the decoding is inherently sequential, decoding a longer output sequence can be slower. Modeling phrases allows to emit a group of words at the same time, potentially speeding-up the decoding. Additionally, so far, both CTC and Seq2Seq have demonstrated enough modeling prowess by being able to successfully target a variety of output tokens. This motivates us to ask how complex can the output tokens get before these models start to fail. Training these models to target phrases might help us explore the limits of these E2E models.

Having said that, it is important that we demonstrate the feasibility of this approach first. Hence, in this work, we only check if E2E models can accurately emit phrases. We leave the investigation into construction of phrase-based vocabularies that lead to better modeling or speed-ups in training/inference for future studies. To that extent, we construct the 3 vocabularies: $P2$, $P3$ and $P4$. These are vocabularies that contain up to 2/3/4-gram phrases respectively.

To construct $P2$, we first collect all word-bigrams, that occur in the training set with frequency greater than a certain threshold. Then, we simply augment this set of bigrams with the mixed-units vocabulary. Hence $P2$ not only contains frequent word-bigrams, but also frequent words, subwords and letters. $P3$ and $P4$ are constructed in a similar manner, i.e., by augmenting frequent word-trigrams and word-4grams with $P2$ and $P3$ respectively.

To test the limits of E2E modeling, we want the network to prioritize emitting higher-order n-grams first, and then consequently fall back to emitting lower-order ngrams, words, subwords and letters, in that order. To achieve this behavior, we adopt the tokenization strategy below. To tokenize a sentence using Pn ($n = 2, 3, 4$):

- Check if a frequent n-gram from Pn is present in the sentence. If multiple such n-grams are present, select the n-gram with the greatest frequency. Collapse this n-gram into one single token. For example, "Hey Cortana" will collapse into "Hey+Cortana", "on the way" will collapse into "on+the+way" and so on. In this way, the model can emit multiple words at the same time.
- Repeat the above step until there are no n-grams to collapse in the sentence

- The remaining sentence is tokenized exactly like $P(n-1)$. Note that $P1$ is just mixed-units, were frequent unigrams are tokenized first and OOV words are broken into a sequence of words and subwords.

As a side note, please see that this kind of tokenization is not likely to yield the best final WER. For that, the network should prioritize emitting tokens that it can model most accurately (irrespective of their length). Table 1 shows how a particular utterance was tokenized differently for different tokens.

4. Experiment details

4.1. Dataset and preparation

All the experiments were performed on Microsoft's Cortana voice assistant task. The training dataset contains approximately 3.3 million short utterances ($\sim$ 3400 hours) in US-English. The test set contains about 5600 utterances ($\sim$ 6 hours). 0.05% of the utterances were extracted from the training data to form the validation set. The base feature vector for every 10 ms is a 80-dimensional vector containing log filterbank energies. The base feature vectors in three continuous frames are stacked together to form the 240-dimension input feature that is used to train both CTC and Seq2Seq models [19]

4.2. Models

Both Seq2Seq and CTC models were developed independently, with best effort for each model class. The CTC models were developed and trained using CNTK [20] while PyTorch [21] was used for Seq2Seq models. Within a particular model class though, the training schedule and other details remain the same for various output tokens. We evaluate the test set using the checkpoint that produced the best loss on validation set.

Both Seq2Seq and CTC have a 6-layer bidirectional RNN [22] with 512 hidden units in each layer. Seq2Seq adopts Gated Recurrent Unit (GRU) [23] cells, while CTC has LSTM [24] cells. Furthermore, Seq2Seq models also have a decoder which is a 2-layer forward-only GRU-RNN. These models use dropout [25] with $p = 0.1$ and layer normalization [26] in both encoder and decoder. To optimize them, we minimize the label-smoothed cross-entropy [27] loss between the posterior probabilities from the decoder and the ground truth tokens. We also use scheduled sampling [28] during training. The sampling probability starts at 0.0 and gradually ramps up to 0.4 [17].

4.3. Tokens

We train Seq2Seq and CTC models with all the tokens described in section 3. The size of 'character' vocabulary is 30, including the 26 alphabets and a few other miscellaneous tokens like $\langle space \rangle$ and ctc_blank. To construct 'words' vocabulary, we select all the words that have occurred at least 10 times in the dataset. This gives us $\sim$27k words and rest of the words are mapped to the OOV token, yielding an OOV rate of 1.8%. For word-pieces, we run the BPE algorithm [7] to get $\sim$29k units. The mixed-units vocabulary is constructed by augmenting the $\sim$ 27k frequent words with sub-word units produced according to [8]. The size mixed-unit vocabulary totals to $\sim$33k.

To construct the phrase-based vocabularies ($P2$, $P3$, $P4$), we keep cut-off frequency for 2/3/4-gram as 100. Note that the cut-off frequency for 1-gram was 10. If we keep the same cut-off frequency for phrases, the vocabulary size increases drastically. For example, a cut-off frequency of 10 will give us 104k bigrams, while the same cut-off gave us only 27k unigrams. Hence, we decided to set a much higher frequency cut-off for phrases. This gives us 11k 2-grams, 5k 3-grams and 2k 4-grams. The total vocabulary sizes for $P2$, $P3$, $P4$ become 45k, 50k and 52k respectively. Next we apply the tokenization schemes for $P2$, $P3$ and $P4$ as described in section 3.5. Post tokenization, we analyze the % of various token types present in the train set corresponding to each phrase based vocabulary (Table 2). Note that there are significant amount of phrases as target tokens in the training sets.

5. Results

Table 3 shows the greedy 1-best test WERs and model sizes for Seq2Seq and CTC across 7 different tokens. We do not use any beam search or language models during decoding. Note that all Seq2Seq and CTC models share exactly the same architecture within their respective model types. The only difference in test WER and model sizes comes from targeting different vocabularies. Across the baseline tokens (i.e., 'characters', 'words', 'word-pieces' and 'mixed-units'), we see that Seq2Seq outperforms CTC models by a significant margin most of the time. This is expected as Seq2Seq overcomes limitations of CTC, like the conditional independence assumption. This does not allow it to model dependencies between the output tokens. As a result, Seq2Seq vastly outperforms CTC when both are trained to emit characters. This problem with CTC is, however, mitigated to a large extent when it is trained to target word-level units i.e., 'words', 'word-pieces' and 'mixed-units'. Even for these units, CTC's results are still inferior to Seq2Seq which tells us that Seq2Seq is, arguably a better E2E modeling technique. However, the results for 'words' tokens do not follow this trend. We think that Seq2Seq performs worse than CTC in this case because of the OOV token. OOV token has the potential to disturb the modeling of dependencies between output tokens in the Seq2Seq decoder. This would not be a concern for CTC models. Note that despite following similar encoder architectures (6-layer 512-unit RNNs), Seq2Seq models are considerably smaller than CTC. This is because the CTC models use LSTMs while Seq2Seq models use GRUs, which are much more compact.

The results for phrase-based vocabularies ($P2$, $P3$ and $P4$) makes the gap between Seq2Seq and CTC performance much more prominent. While the results for Seq2Seq are pretty encouraging, performance degrades drastically for CTC when compared to the baseline units. To check if these models are indeed emitting phrases, we calculate the distribution of the emitted tokens for both Seq2Seq (Table 4) and CTC (Table 5). We see that both Seq2Seq and CTC do emit phrases in proportions similar (but not exact) to what was present in the training sets. But while Seq2Seq gives a reasonably good final WER, CTC WER suffers significantly. This leads us to believe that CTC is not strong enough to deal with tokens that vary greatly in input lengths. Another interesting observation is that both models emit less number of "mix" tokens (i.e. combinations of words and/or sub-words to deal with OOV) than present in the train set. CTC's mix tokens output, in particular, is very low. This likely tells us that the sub-words approach might not be very effective in dealing with OOVs. As noted earlier, the phrase-based vocabularies were designed only to check the feasibility of the phrase-based approach. Hence, we were not expecting phrase-based vocabularies would outperform the baseline tokens. However, these experiments do show that Seq2Seq can be trained to emit reasonably long phrases, which we think is an important result. We plan to leverage this in our future work to

Characters	Words	Word-pieces	Mixed-units	$P2$	$P3$	$P4$
$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$
h	hey	hey	hey	hey+cortana	hey+cortana+what's	hey+cortana+what's+the
e	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$
y	cortana	cortana	cortana	what's+the	the+traffic+like	traffic
$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$
c	what's	what	what's	traffic+like	on+the+way	like
o	$\langle space \rangle$	'	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$
r	the	s	the	on+the	to	on+the+way+to
t	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$
a	traffic	the	traffic	way+to	clair	clair
n	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	ton	ton
a	like	traffic	like	clair	$\langle space \rangle$	$\langle space \rangle$
$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	ton	.	.
w	on	like	on	$\langle space \rangle$		
h	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$	.		
a	the	on	the			
t	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$			
'	way	the	way			
s	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$			
$\langle space \rangle$	to	way	to			
t	$\langle space \rangle$	$\langle space \rangle$	$\langle space \rangle$			
h	$\langle oov \rangle$	to	clair			
e	$\langle space \rangle$	$\langle space \rangle$	ton			
$\langle space \rangle$	.	clair	$\langle space \rangle$			
t		ton	.			
r		$\langle space \rangle$				
.....		.				

Table 1: *"hey cortana, what's the traffic like on the way to clairton" tokenized for different token types. 'clairton' is an OOV in this case. $\langle space \rangle$ is used as delimiter between words. 'Characters' tokenized output has been clipped for lack of space. Note that the length of output sequence is considerably shorter for phrase-based vocabularies $P2$, $P3$ and $P4$.*

tokens	mix	1-gram	2-gram	3-gram	4-gram
$P2$	4.9	57.2	37.9	0	0
$P3$	5.0	66.8	14.2	13.9	0
$P4$	4.9	73.8	10.0	5.7	5.5

Table 2: *% of various token types in the train-set after applying tokenization schemes for $P2$, $P3$ and $P4$. 'x-gram' means a word-phrase with 'x' words. 'mix' is count of words and sub-words when their combination is used to represent an OOV.*

token Type	token Size	WER(%)		Size (MB)	
		S2S	CTC	S2S	CTC
Characters	30	9.54	17.54	81	139
Words	27k	10.95	9.84	186	242
Word-pieces	29k	7.75	9.73	195	256
Mixed-units	34k	7.51	9.32	213	274
$P2$	45k	8.15	13.76	256	315
$P3$	50k	8.31	21.97	277	339
$P4$	52k	8.21	24.87	286	348

Table 3: *Greedy decoding test-WER and model size for Seq2Seq and CTC models on the 4 baseline and 3 phrase-based vocabularies. Token size is the size of the vocabulary. Tokens with large vocabulary sizes have been rounded off to the nearest 1k.*

design phrase-based vocabularies that can provide faster training/inference and potentially better modeling.

tokens	mix	1-gram	2-gram	3-gram	4-gram
$P2$	2.7	57.0	40.2	0	0
$P3$	2.6	70.3	11.0	16.1	0
$P4$	2.3	78.7	6.7	4.7	7.5

Table 4: *% of various token types emitted by Seq2Seq models.*

tokens	mix	1-gram	2-gram	3-gram	4-gram
$P2$	0.9	57.4	41.5	0	0
$P3$	0.06	72.9	10.5	16.4	0
$P4$	0.01	82.5	5.7	4.1	7.4

Table 5: *% of various token types emitted by CTC models*

6. Conclusions

In this work, we explore if E2E ASR models can be trained to emit phrases. We construct 3 phrase-based vocabularies for a 3400 hour Microsoft Cortana task and train Seq2Seq and CTC models using them. We present the results in context of 4 strong baselines that were trained on a variety of tokens including characters, words, word-pieces and mixed-units. We find that while CTC can not accurately learn to model phrases, Seq2Seq can easily learn to emit phrases that are up-to 4 words long, without much degradation in the final WER. Being able to target phrases as output tokens opens up the possibility of improved modeling and speed-up in training/inference. In future, we plan to explore phrase-based vocabularies that directly optimize these potential benefits.

7. References

[1] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 273–278.

[2] 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. C. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 6381–6385.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[5] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[6] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 173–182. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045410

[7] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://www.aclweb.org/anthology/P16-1162

[8] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing acoustic-to-word ctc model," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5794–5798.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proceedings of the National Academy of Sciences*, 1982.

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

[11] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *INTERSPEECH*, 08 2017, pp. 1298–1302.

[12] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143891

[13] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition," *Proc. Interspeech*, 2016.

[14] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo, "Direct acoustics-to-word models for english conversational speech recognition," in *INTERSPEECH*, 08 2017, pp. 959–963.

[15] S. Palaskar and F. Metze, "Acoustic-to-word recognition with sequence-to-sequence models," *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 397–404, 2018.

[16] J. Li, G. Ye, R. Zhao, J. Droppo, and Y. Gong, "Acoustic-to-word model without oov," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 111–117.

[17] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[18] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 12 2017, pp. 193–199.

[19] H. Sak, A. W. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *CoRR*, vol. abs/1507.06947, 2015. [Online]. Available: http://arxiv.org/abs/1507.06947

[20] F. Seide and A. Agarwal, "Cntk: Microsoft's open-source deep-learning toolkit," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 2135–2135. [Online]. Available: http://doi.acm.org/10.1145/2939672.2945397

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[26] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv e-prints*, Jul. 2016.

[27] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Proc. Interspeech 2017*, 08 2017, pp. 523–527.

[28] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1171–1179.