Advancing Acoustic-to-Word CTC Model with Attention and Mixed-Units

Amit Das, *Student Member, IEEE,* Jinyu Li, *Member, IEEE,* Guoli Ye, *Member, IEEE,* Rui Zhao, *Member, IEEE,* and Yifan Gong, *Senior Member, IEEE*

***Abstract*—The acoustic-to-word model based on the Connectionist Temporal Classification (CTC) criterion is a natural end-to-end (E2E) system directly targeting word as output unit. Two issues exist in the system: first, the current output of the CTC model relies on the current input and does not account for context weighted inputs. This is the hard alignment issue. Second, the word-based CTC model suffers from the out-of-vocabulary (OOV) issue. This means it can model only frequently occurring words while tagging the remaining words as OOV. Hence, such a model is limited in its capacity in recognizing only a fixed set of frequent words. In this study, we propose addressing these problems using a combination of attention mechanism and mixed-units. In particular, we introduce Attention CTC, Self-Attention CTC, Hybrid CTC, and Mixed-unit CTC.**

First, we blend attention modeling capabilities directly into the CTC network using Attention CTC and Self-Attention CTC. Second, to alleviate the OOV issue, we present Hybrid CTC which uses a word and letter CTC with shared hidden layers. The Hybrid CTC consults the letter CTC when the word CTC emits an OOV. Then, we propose a much better solution by training a Mixed-unit CTC which decomposes all the OOV words into sequences of frequent words and multi-letter units. Evaluated on a 3400 hours Microsoft Cortana voice assistant task, our final acoustic-to-word solution using attention and mixed-units achieves a relative reduction in word error rate (WER) over the vanilla word CTC by 12.09%. Such an E2E model without using any language model (LM) or complex decoder also outperforms a traditional context-dependent (CD) phoneme CTC with strong LM and decoder by 6.79% relative.

***Index Terms*—CTC, OOV, acoustic-to-word, attention, end-to-end system, speech recognition**

I. Introduction

In automatic speech recognition (ASR), we are given a sequence of acoustic feature vectors $\mathbf{x}$. The objective is to decode a sequence of words $\mathbf{y}$ from $\mathbf{x}$ with minimum probability of error. With the 0-1 loss function, the optimal solution uses the Bayesian Maximum Aposteriori (MAP) rule

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \; P(\mathbf{y}|\mathbf{x};\Theta_{\text{ASR}}), \tag{1}$$

$$= \arg\max_{\mathbf{y}} \; P(\mathbf{x}|\mathbf{y};\Theta_{\text{AM}})P(\mathbf{y};\Theta_{\text{LM}}). \tag{2}$$

However, to reduce complexity, practical ASR systems often use the sub-optimal solution

$$\hat{\mathbf{y}} \approx \arg\max_{\mathbf{y},\mathbf{l}} \; P(\mathbf{x}|\mathbf{l};\Theta_{\text{AM}})P(\mathbf{l}|\mathbf{y};\Theta_{\text{PM}})P(\mathbf{y};\Theta_{\text{LM}}). \tag{3}$$

The authors are with Microsoft Corporation, USA (email: amitdas@illinois.edu; jinyli@microsoft.com; guoye@microsoft.com; ruzhao@microsoft.com; yifan.gong@microsoft.com).

Here, $\mathbf{l}$ is a sequence of phonemes and $\Theta_{\text{ASR}} = \{\Theta_{\text{AM}}, \Theta_{\text{PM}}, \Theta_{\text{LM}}\}$ is the set of parameters to be estimated during training. The first term $P(\mathbf{x}|\mathbf{l};\Theta_{\text{AM}})$ in Eq. (3) is the likelihood of the features given the phoneme sequence and is obtained from an acoustic model (AM). The second term $P(\mathbf{l}|\mathbf{y};\Theta_{\text{PM}})$ is the likelihood of the phoneme sequence given the word sequence and is obtained from a lexicon or pronunciation model (PM). The third term $P(\mathbf{y};\Theta_{\text{LM}})$ is the prior probability of the word sequence and is obtained from a language model (LM).

In theory, all $\{\Theta_{\text{AM}}, \Theta_{\text{PM}}, \Theta_{\text{LM}}\}$ should be estimated jointly. However, in practice, they are estimated separately and hence training an ASR system becomes a complex disjoint learning problem. Moreover, decoding at test time involves a complex graph search procedure which is intensive both in time and memory. This makes traditional ASR systems often cumbersome for deployment in real-world devices.

In contrast, an end-to-end (E2E) ASR system [1]–[10] circumvents the disjoint learning problem by directly transducing a sequence of features $\mathbf{x}$ to a sequence of words $\mathbf{y}$. Some widely used contemporary neural network based E2E approaches for sequence-to-sequence transduction are: (a) Connectionist Temporal Classification (CTC) [11], [12], (b) Recurrent Neural Network (RNN) Encoder-Decoder (ED) [13]–[16], and (c) RNN Transducer (RNN-T) [17]. These approaches have been successfully applied to large scale ASR [2]–[6], [9], [18]–[24]. In this study, we confine ourselves to the CTC approach.

CTC, first introduced in [11], [12], involves training a stack of underlying RNNs and minimizing the sequence level cross-entropy (CE) loss $-\log P(\mathbf{y}|\mathbf{x})$. In contrast, RNN training minimizes the frame level CE loss. Moreover, CTC networks offer the versatility to model output units of different sizes such as monophones, characters, words, or other sub-word units. Owing to this simplicity in the training structure and versatility of output units, CTC is regarded as one of the most popular E2E methods [1]–[3], [19], [25]–[32].

In ASR, the number of output labels in $\mathbf{y}$ is usually smaller than the number of input speech frames in $\mathbf{x}$. However, since a CTC network is essentially an RNN, it is forced to predict a label for every frame in $\mathbf{x}$. Since some frames may not always be associated with a label (a) CTC introduces a special *blank* label as an additional output label which acts as a filler and, (b) it allows for repetition of labels (for both blank or non-blank). As a result, CTC frame level outputs are usually dominated by blank labels. The outputs corresponding to the non-blank labels usually occur with spikes in their posteriors because of their high confidences. Thus, an easy way to convert intermediate frame level outputs to final ASR outputs using CTC involves a simple two-step procedure. In the first step, generate a sequence of labels corresponding to the highest posteriors and merge consecutive duplicate labels. In the second step, remove the blank labels and concatenate the remaining non-blank labels into words. This is known as greedy decoding. It is a very attractive feature for E2E modeling as there is neither any LM nor any complex decoding involved. This makes it easy for deployment in real-world devices. The E2E ASR developed in this study uses greedy decoding.

As the goal of ASR is to generate a word sequence from speech acoustics, the word is the most natural output unit compared to other output units such as monophones or characters. A big challenge in the word-based CTC model, a.k.a. acoustic-to-word (A2W) CTC or word CTC, is the OOV issue [33]–[36]. In [19], [26], [30], only the most frequent words in the training set were used as output targets whereas the remaining words were lumped together as OOVs. These OOVs can neither be modeled nor recognized correctly. For example, consider an utterance containing the sequence "have you been to newyorkabc" in which "newyorkabc" is an OOV (infrequent) word. For an OOV-based model, a likely output for this utterance would be "have you been to OOV". Despite it being the expected output from the OOV-based model, the presence of the OOV tag in the sentence degrades the end-user experience. Another disadvantage of OOV modeling is that the data related to those infrequent words are wasted, resulting in reduced modeling power. To underscore this issue, [26] trained a word CTC with up to 25 thousand (k) word targets. However, the ASR accuracy of the word CTC was far below the accuracy of a context dependent (CD) phoneme CTC model with LM, partially due to the high OOV rate when using only around 3k hours of training data.

The accuracy gap between a word CTC and CD phoneme CTC can be attributed to multiple reasons. First, training a word CTC requires orders of magnitude of more training data than a CD phoneme CTC because words which qualify as non-OOVs (frequent words) require sufficient number of training examples. Words which do not meet this sufficiency requirement are simply tagged as OOVs. Hence, such words can neither be modeled as valid words during training nor recognized during evaluation. Second, even in the presence of large training data, it is difficult to capture the entire vocabulary of a language. For example, a word CTC cannot handle unfamiliar nouns or emerging hot-words (e.g. selfie, meme, unfriend) which gradually become popular after an acoustic model has been built.

Several studies in the past have attempted to address these issues. In [19], it was shown that by using 100k words as output targets and by training the model with 125k hours of data, a word CTC was able to outperform a CD phoneme CTC. However, easy accessibility to such large databases is rare. Usually, at most a few thousand hours of data are available. In [37], the authors were able to train a word CTC model with only 2k hours of data achieving ASR accuracy comparable to that of a CD phoneme CTC. Their proposed training regime included initializing the word CTC with a well-trained phoneme CTC, curriculum learning [38], Nesterov momentum-based stochastic gradient descent, dropout, and low rank matrix factorization [39]. To address the hot-words issue, [37] also proposed a spell and recognize (SAR) model which has a combination of words and characters as output targets. The SAR model is used to learn to first spell a word as a sequence of characters and then recognize it as a whole word. However, whenever an OOV is detected, the decoder consults the letter sequence from the speller. Thus, the displayed hypothesis to the end-user contains words (for non-OOVs) and characters (for OOVs). Spelling out the characters for OOVs is more meaningful to the users than simply displaying "OOV". However, it was reported that the overall recognition accuracy of the SAR model improved only marginally over a word-only CTC. In [40], the authors proposed training two CTC models separately - an acoustics-to-phoneme model from acoustic data and a phoneme-to-word model using text data respectively. Then, the two models were jointly optimized resulting in an A2W model.

In this study, we propose four solutions to improve the recognition accuracy of the all-neural word CTC using only 3400 hours of training data while also alleviating the OOV issue.

- First, in Section III, we propose *Attention CTC* [41] to address the inherent hard alignment problem in CTC. Since CTC relies on the hidden feature vector at the current time to make predictions, it does not directly attend to feature vectors of the neighboring frames. This is the hard alignment problem which makes CTC's output independent assumption worse. Our proposed solution generates new hidden features that carry attention weighted context information. We achieved this by blending some concepts from RNN-ED into CTC modeling.
- Second, in Section IV, we investigate another attention mechanism called *Self-Attention* [42] in CTC networks.
- Third, we propose *Hybrid CTC* [31] which is a single CTC consisting of a word CTC and a letter CTC trained jointly using multi-task learning (MTL) [43], [44]. We train the word CTC first and then add a letter CTC as an auxiliary task by sharing the hidden layers of the word CTC. During recognition, the word and letter CTCs generate sequences of words and letters respectively. However, the letter CTC is consulted for the letter sequence only when the word CTC emits an OOV token. This makes the Hybrid CTC capable of recognizing OOVs and thereby reducing errors introduced by OOVs.
- Finally, we further improve the word CTC and reduce OOV errors by introducing *Mixed-unit CTC* [45]. Here, during training, the OOV word is decomposed into a sequence of frequent words and letters (which we refer to as *mixed-units*). During testing, we perform greedy decoding for the whole E2E system in a single step without the need of using the two-stage process (OOV-detection and then letter-sequence-consulting) as in Hybrid CTC. We will later show that a CTC with mixed-units outperformed a CTC with wordpieces which have become popular in recent RNN-ED

frameworks [9].

Our final proposed word CTC achieved a relative WER reduction (WERR) of about 12.09% over the vanilla word CTC [11]. Furthermore, the same word CTC outperformed the traditional CD phoneme CTC with a strong LM and decoder by 6.79% relative.

The remainder of the article is organized as follows. In Section II we give a brief overview of CTC and RNN-ED. In Sections III, IV, V, VI, we explain the proposed Attention CTC, Self-Attention CTC, Hybrid CTC, and Mixed-unit CTC respectively. In Section VII, we provide experimental evaluations of our proposed algorithms. Finally, we summarize our study and draw conclusions in Section VIII. The terms letter and character have been interchangeably used in this study.

II. End-to-End Speech Recognition

An E2E ASR system models the posterior distribution $p(\mathbf{y}|\mathbf{x})$ by transducing an input sequence of acoustic feature vectors $\mathbf{x}$ to an output sequence of tokens $\mathbf{y}$ (phonemes, characters, words etc.). More specifically, for an input sequence of feature vectors $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$ of length T with $\mathbf{x}_t \in \mathbb{R}^m$, an E2E ASR system transduces the input sequence to an intermediate sequence of hidden feature vectors $\mathbf{h} = (\mathbf{h}_1, \cdots, \mathbf{h}_L)$ of length L with $\mathbf{h}_l \in \mathbb{R}^n$. The sequence $\mathbf{h}$ undergoes another transduction resulting in an output sequence $\mathbf{y}$ whose posterior probability is $\tilde{p}(\mathbf{y}|\mathbf{x})$. Here $\mathbf{y} = (y_1, \cdots, y_U)$ is of length U with $y_u \in \mathbb{L}$, $\mathbb{L}$ being the label set. Usually $U \leq T$ and $L = T$ in E2E ASR systems. Thus, an E2E neural network, parameterized by $\mathbf{W}$, learns a many-to-one functional $\mathbf{f_W} : \mathbf{x} \mapsto \tilde{p}(\mathbf{y}|\mathbf{x})$ where $\tilde{p}(\mathbf{y}|\mathbf{x})$ closely resembles the true $p(\mathbf{y}|\mathbf{x})$.

A. Connectionist Temporal Classification (CTC)

A CTC network uses a recurrent neural network (RNN) and the CTC error criterion [11], [12] which directly optimizes the prediction of a transcription sequence. As the length of the output labels is shorter than the length of the input speech frames, a CTC path is introduced to make their lengths equal by adding the blank symbol ϕ as an additional label and allowing repetition of labels. Thus, the new label set becomes $\mathbb{L}' = \mathbb{L} \cup \phi$. Let $K = |\mathbb{L}'|$ be the cardinality of the label set $\mathbb{L}'$.

Denote $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_T)$ as the CTC path (or alignment) with $\pi_t \in \mathbb{L}'$, $\mathbf{y}$ as the target label sequence (transcription) we want to recognize, and $B^{-1}(\mathbf{y})$ as the preimage of $\mathbf{y}$ mapping all possible CTC paths $\boldsymbol{\pi}$ that result in $\mathbf{y}$. Then, the CTC loss function is defined as the negative log of sum of the probabilities of all possible CTC paths $\boldsymbol{\pi}$ that result in $\mathbf{y}$. This is given by

$$L_{CTC} = -\ln p(\mathbf{y}|\mathbf{x}) = -\ln \sum_{\boldsymbol{\pi} \in B^{-1}(\mathbf{y})} p(\boldsymbol{\pi}|\mathbf{x}). \tag{4}$$

With the conditional independence assumption ($\pi_t \perp\!\!\!\perp \pi_{\neq t}|\mathbf{x}$), $p(\boldsymbol{\pi}|\mathbf{x})$ can be further decomposed into a product of posteriors of each frame as

$$p(\boldsymbol{\pi}|\mathbf{x}) = \prod_{t=1}^{T} p(\pi_t|\mathbf{x}). \tag{5}$$

During decoding, it is very simple to generate the decoded sequence using greedy decoding: simply concatenate the labels corresponding to the highest posteriors and merge the duplicate labels; then remove the blank labels. Thus, there is neither a language model nor any complex graph search in greedy decoding.

B. RNN Encoder-Decoder (RNN-ED)

An RNN-ED [13]–[16] uses two distinct networks - an RNN encoder network that transforms $\mathbf{x}$ into $\mathbf{h}$ and an RNN decoder network that transforms $\mathbf{h}$ into $\mathbf{y}$. Using these, an RNN-ED models $p(\mathbf{y}|\mathbf{x})$ as

$$p(\mathbf{y}|\mathbf{x}) = \prod_{u=1}^{U} p(y_u|\mathbf{y}_{1:u-1}, \mathbf{c}_u), \tag{6}$$

where $\mathbf{c}_u$ is the context vector at time u and is a function of $\mathbf{x}$. There are two key differences between CTC and RNN-ED. First, $p(\mathbf{y}|\mathbf{x})$ in Eq. (6) is generated using a product of ordered conditionals. Thus, RNN-ED is not impeded by the conditional independence constraint of Eq. (5). Second, The decoder output y_u at time u is dependent on $\mathbf{c}_u$ which is a weighted sum of all its inputs (soft alignment), i.e., $\mathbf{h}_t, t = 1, \cdots, T$. In contrast, CTC generates y_u using only $\mathbf{h}_t$ (hard alignment).

The decoder network of RNN-ED has three components: a multinomial distribution generator Eq. (7), an RNN decoder Eq. (8), and an attention network Eq. (9)-(14) [15], [16] as follows:

$$p(y_u|\mathbf{y}_{1:u-1}, \mathbf{c}_u) = \text{Generate}(y_{u-1}, \mathbf{s}_u, \mathbf{c}_u), \tag{7}$$

$$\mathbf{s}_u = \text{Recurrent}(\mathbf{s}_{u-1}, \mathbf{y}_{u-1}, \mathbf{c}_u), \tag{8}$$

$$\mathbf{c}_u = \text{Annotate}(\boldsymbol{\alpha}_u, \mathbf{h}) = \sum_{t=1}^{T} \alpha_{u,t}\mathbf{h}_t, \tag{9}$$

$$\alpha_{u,t} = \text{Attend}(\mathbf{s}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{h}_t), \quad t = 1, \cdots, T. \tag{10}$$

Here, $\mathbf{h}_t, \mathbf{c}_u \in \mathbb{R}^n$, and $\boldsymbol{\alpha}_u = [\alpha_{u,1} \cdots \alpha_{u,T}]$ is a probability distribution. Hence, $\alpha_{u,t} \in \mathbb{U}$ with $\mathbb{U} = [0, 1]$ such that $\sum_t \alpha_{u,t} = 1$. Also, for simplicity assume $\mathbf{s}_u \in \mathbb{R}^n$. Generate(.) is a feedforward network with a softmax operation generating the ordered conditional $p(y_u|\mathbf{y}_{1:u-1}, \mathbf{c}_u)$. Recurrent(.) is an RNN decoder operating on the output time axis indexed by u and has hidden state $\mathbf{s}_u$. Annotate(.) computes the context vector $\mathbf{c}_u$ (also called the soft alignment) using the attention probability vector $\boldsymbol{\alpha}_u$ and the hidden sequence $\mathbf{h}$. Attend(.) computes the attention weight $\alpha_{u,t}$ using a single layer feedforward network (Score(.) function) followed by softmax normalization as follows:

$$e_{u,t} = \text{Score}(\mathbf{s}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{h}_t), \quad t = 1, \cdots, T, \tag{11}$$

$$\alpha_{u,t} = \frac{\exp(e_{u,t})}{\sum_{t'=1}^{T} \exp(e_{u,t'})}, \quad t = 1, \cdots, T. \tag{12}$$

Here, $e_{u,t} \in \mathbb{R}$ and Score(.) can either be a content-based or hybrid-based function. The latter encodes both content ($\mathbf{s}_{u-1}$)

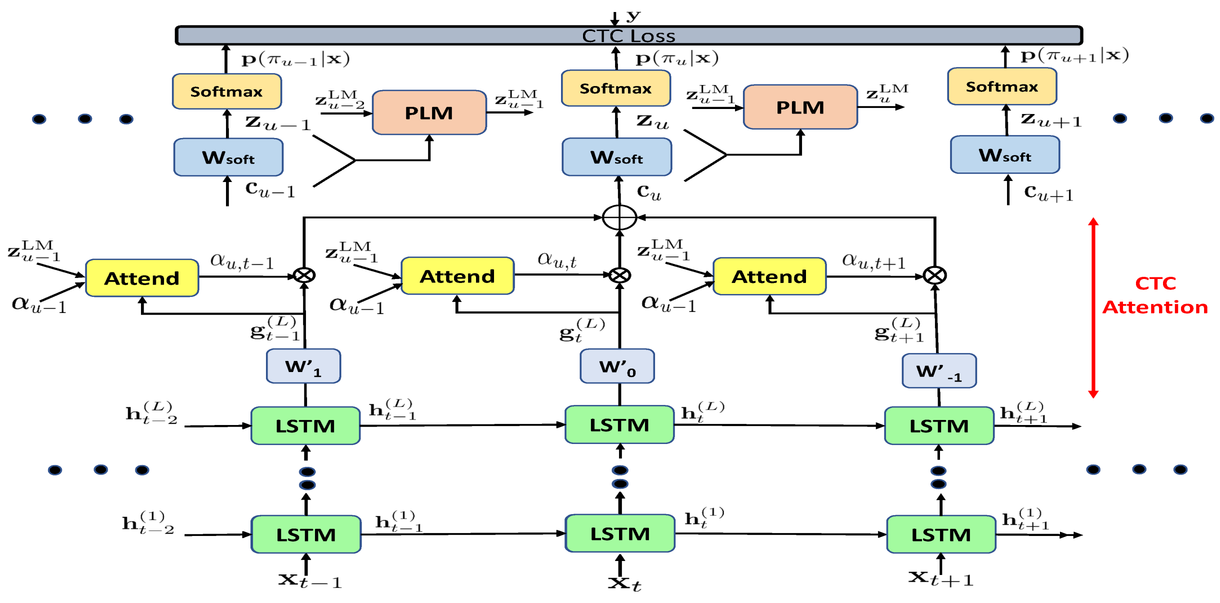


Fig. 1. An example of an Attention CTC network with an attention window of size $C = 3$.

and location ($\boldsymbol{\alpha}_{u-1}$) information. Score(.) is computed using

$$e_{u,t} = \begin{cases} \mathbf{v}^T \tanh\left(\mathbf{U}\mathbf{s}_{u-1} + \mathbf{W}\mathbf{h}_t + \mathbf{b}\right), & \text{(content)} \\ \mathbf{v}^T \tanh\left(\mathbf{U}\mathbf{s}_{u-1} + \mathbf{W}\mathbf{h}_t + \mathbf{V}\mathbf{f}_u + \mathbf{b}\right), & \text{(hybrid)} \end{cases} \tag{13}$$

$$\text{where,} \quad \mathbf{f}_u = \mathbf{F} * \boldsymbol{\alpha}_{u-1}. \tag{14}$$

The operation $*$ denotes convolution. Thus, in the hybrid case, the dependence on $\boldsymbol{\alpha}_{u-1}$ is through $\mathbf{f}_u$. Attention parameters $\mathbf{U}, \mathbf{W}, \mathbf{V}, \mathbf{F}, \mathbf{b}, \mathbf{v}$ are learned while training RNN-ED.

III. Attention CTC

In this section, we outline various steps required to model attention directly within CTC. In the past, several attempts have been made to apply attention on E2E models. For example, attention-based RNN-ED [15], [16] network was used to predict word outputs in [46]. Other studies have investigated using CTC as an auxiliary task to improve attention-based RNN-ED using an MTL framework. For example, CTC was used either at the top layer [47], [48] or at an intermediate layer [49] in the MTL framework. Extensions of CTC such as RNN-T [17], [20] and RNN aligner [7] either change the objective function or the training process to relax the frame independence assumption of CTC. However, none of these approaches used attention directly within the CTC network. The proposed Attention CTC model is different from all these approaches since we use attention mechanism to improve the hidden layer representations with more context information without changing the CTC objective function and the training process. Our primary motivation in this work is to address the hard alignment problem of CTC, as outlined earlier in Section I, by modeling attention directly within the CTC framework.

An example of the proposed Attention CTC network is shown in Figure 1. We propose the following key ideas to blend attention into CTC. (a) First, we derive context vectors using *time convolution features* (Section III-A) and apply attention weights on these context vectors (Section III-B). This makes it possible for CTC to be trained using soft alignments instead of hard. (b) Second, to improve attention modeling, we incorporate a *pseudo language model* (Section III-C) during CTC training. (c) Finally, we improve our attention modeling further by introducing *component attention* (Section III-D) where context vectors are produced as a result of applying attention on hidden features across both time and their individual components. We explain each of these ideas separately with illustrations in the following subsections. We will use the indices t and u to denote the time step for input $\mathbf{h}$ and output $\mathbf{c}$ respectively of the attention block to maintain notational consistency with RNN-ED.

A. Time Convolution (TC) Features

First, we construct TC features from the hidden outputs $\mathbf{h}$ of the last LSTM layer. This is illustrated in Fig. 2. Consider a subsequence of $\mathbf{h}$ rather than the entire sequence. We refer to this subsequence, $(\mathbf{h}_{u-\tau}, \cdots, \mathbf{h}_u, \cdots, \mathbf{h}_{u+\tau})$, as the *attention window*. Each $\mathbf{h}_t \in \mathbb{R}^n$. The attention window is centered around the current time u with τ being the length of the attention window on either side of u. Thus, the total length of the attention window is $C = 2\tau + 1$. Now consider C time convolution kernels $(\mathbf{W}'_{u-\tau}, \cdots, \mathbf{W}'_u, \cdots, \mathbf{W}'_{u+\tau})$ where $\mathbf{W}'_t \in \mathbb{R}^{n \times n}$ and $\mathbf{W}'_{t_1} \neq \mathbf{W}'_{t_2}$ for $t_1 \neq t_2$. Then the context vector $\mathbf{c}_u$ is computed using time convolution as,

$$\begin{aligned} \mathbf{c}_u &= \sum_{t=u-\tau}^{u+\tau} \mathbf{W}'_{u-t} \mathbf{h}_t \\ &\triangleq \sum_{t=u-\tau}^{u+\tau} \mathbf{g}_t \\ &= \gamma \sum_{t=u-\tau}^{u+\tau} \alpha_{u,t} \mathbf{g}_t. \end{aligned} \tag{15}$$

Here, $\mathbf{g}_t, \mathbf{c}_u \in \mathbb{R}^n$ represents the *filtered* signal at time t. The last step Eq. (15) holds when $\alpha_{u,t} = \frac{1}{C}$ and $\gamma = C$. Since Eq. (15) is similar to Eq. (9) in structure, $\mathbf{c}_u$ represents a special case context vector with uniform attention weights $\alpha_{u,t} = \frac{1}{C}$,

$t \in [u-\tau,\ u+\tau]$. Moreover, $\mathbf{c}_u$ is a result of convolving features $\mathbf{h}$ with $\mathbf{W}'$ in time. Thus, $\mathbf{W}'$ and $\mathbf{c}_u$ represent *time convolution kernel* and *time convolution feature* respectively.

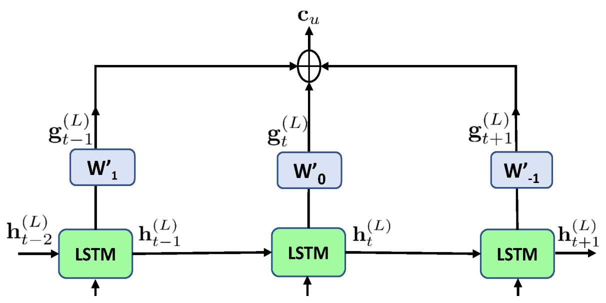


Fig. 2. Time convolution with an attention window of size $C = 3$ (i.e., $\tau = 1$).

B. Content Attention (CA) and Hybrid Attention (HA)

To incorporate non-uniform attention in Eq. (15), we need to compute a non-uniformly distributed $\boldsymbol{\alpha}_u$ where $\boldsymbol{\alpha}_u = (\alpha_{u-\tau}, \cdots, \alpha_u, \cdots, \alpha_{u+\tau})$ using an attention network similar to Eq. (10). However, since there is no explicit decoder like Eq. (8) in CTC, there is no decoder state $\mathbf{s}_u$. Therefore, we use $\mathbf{z}_u$ instead of $\mathbf{s}_u$. The term $\mathbf{z}_u \in \mathbb{R}^K$ is the logit to the softmax and is given by

$$\mathbf{z}_u = \mathbf{W}_{\text{soft}}\mathbf{c}_u + \mathbf{b}_{\text{soft}},$$
$$\mathbf{p}(\pi_u|\mathbf{x}) = \text{Softmax}(\mathbf{z}_u), \tag{16}$$

where $\mathbf{W}_{\text{soft}} \in \mathbb{R}^{K\times n}, \mathbf{b}_{\text{soft}} \in \mathbb{R}^K$. The term $\mathbf{p}(\pi_u|\mathbf{x}) = [p(\pi_u = 1|\mathbf{x})\ p(\pi_u = 2|\mathbf{x}) \cdots p(\pi_u = K|\mathbf{x})]^{\mathrm{T}}$ is the vector of probabilities of labels in the alignment at time u. Thus, Eq. (16) is similar to the Generate(.) function in Eq. (7) but lacks the dependency on $\mathbf{y}_{u-1}$ and $\mathbf{s}_u$. Consequently, the Attend(.) function in Eq. (10) becomes

$$\alpha_{u,t} = \text{Attend}(\mathbf{z}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}_t), \quad t = u-\tau, \cdots, u+\tau \tag{17}$$

where $\mathbf{h}_t$ in Eq. (10) is replaced with $\mathbf{g}_t$. The Attend(.) function is illustrated in Fig. 3 and is simply a single layer neural network with a softmax. A scoring function Score(.), similar to Eq. (11), computes the layer activations. However, here the Score(.) function uses the filtered signal $\mathbf{g}_t$ instead of the raw signal $\mathbf{h}_t$ in Eq. (11). Thus, the new Score(.) function becomes

$$e_{u,t} = \text{Score}(\mathbf{z}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}_t), \quad t = u-\tau, \cdots, u+\tau \tag{18}$$
$$= \begin{cases} \mathbf{v}^T \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{b}), & \text{(content)} \\ \mathbf{v}^T \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{V}\mathbf{f}_u + \mathbf{b}) & \text{(hybrid)} \end{cases} \tag{19}$$

with $\mathbf{f}_u$ a function of $\boldsymbol{\alpha}_{u-1}$ through Eq. (14). The content and location information are encoded in $\mathbf{z}_{u-1}$ and $\boldsymbol{\alpha}_{u-1}$ respectively. Thus, the hybrid function in Eq. (19) includes both content and location information. Scores from Eq. (18) can be normalized using the softmax operation (as in Eq. (12)) to generate non-uniform $\alpha_{u,t}$ for $t \in [u-\tau,\ u+\tau]$. Now, $\boldsymbol{\alpha}_u$ can be plugged into Eq. (15), along with $\mathbf{g}$ to generate the context vector $\mathbf{c}_u$. This completes the attention network. We found that excluding the scale factor γ in Eq. (15), even for non-uniform attention, was detrimental to the final performance. Therefore, we continue to use $\gamma = C$.

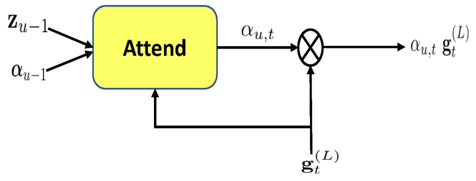
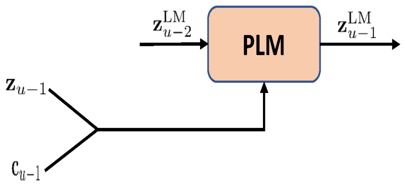


Fig. 3. Content and hybrid attention.



$\mathbf{z}_{u-2}^{\text{LM}}$
PLM
$\mathbf{z}_{u-1}^{\text{LM}}$
$\mathbf{z}_{u-1}$
$\mathbf{c}_{u-1}$

Fig. 4. Pseudo language model.

C. Pseudo Language Model (PLM)

The performance of the attention model can be improved further by providing more reliable content information from the past. This is possible by introducing another recurrent network, which we refer to as PLM, that can utilize content from several time steps in the past instead of just one. This network, in essence, would learn an LM-like model implicitly. This is illustrated in Fig. 4. To build the PLM network, we follow an architecture similar to RNN-LM [50]. As illustrated in the PLM block of Fig. 1, the input to the PLM network is computed by stacking the previous output $\mathbf{z}_{u-1}$ with the context vector $\mathbf{c}_{u-1}$ and feeding it to a recurrent function $\mathcal{H}(.)$. The output of $\mathcal{H}(.)$ is $\mathbf{z}_{u-1}^{\text{LM}}$ which, instead of $\mathbf{z}_{u-1}$, is fed to the Attend(.) block in Eq. (17). This is represented as

$$\mathbf{z}_{u-1}^{\text{LM}} = \mathcal{H}(\mathbf{x}_{u-1}, \mathbf{z}_{u-2}^{\text{LM}}), \quad \mathbf{x}_{u-1} = \begin{bmatrix} \mathbf{z}_{u-1} \\ \mathbf{c}_{u-1} \end{bmatrix}, \tag{20}$$
$$\alpha_{u,t} = \text{Attend}(\mathbf{z}_{u-1}^{\text{LM}}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}_t), \quad t = u-\tau, \cdots, u+\tau. \tag{21}$$

We model $\mathcal{H}(.)$ using a single layer long short-term memory (LSTM) unit [51] with n memory cells and input and output dimensions set to $K+n$ (since $\mathbf{x}_{u-1} \in \mathbb{R}^{K+n}$) and n (since $\mathbf{z}_{u-1}^{\text{LM}} \in \mathbb{R}^n$) respectively. Notice that $\mathbf{z}_{u-1}^{\text{LM}}$ encodes the content of a pseudo LM rather than a true LM since CTC outputs are interspersed with blank symbols by design. Also, $\mathbf{z}_{u-1}^{\text{LM}}$ is a real-valued vector instead of a one-hot vector. Hence, the PLM is not a true LM.

D. Component Attention (COMA)

In the previous sections, $\alpha_{u,t}$ is a scalar term weighting the contribution of the entire n-dimensional vector $\mathbf{g}_t$ to generate the output $\mathbf{p}(\pi_t|\mathbf{x})$. This means all n components (or dimensions) of the vector $\mathbf{g}_t$ are weighted by the same scalar $\alpha_{u,t}$. In this section, we consider weighting each component (dimension) of $\mathbf{g}_t$ using a separate weight. Therefore, we need an n-dimensional weight vector $\boldsymbol{\alpha}_{u,t} \in \mathbb{U}^n$ instead of the scalar $\alpha_{u,t} \in \mathbb{U}$. The vector $\boldsymbol{\alpha}_{u,t}$ can be generated as follows. First, compute an n-dimensional score $\mathbf{e}_{u,t}$ for each t. This is easily achieved using the Score(.) function in Eq. (19) but without taking the inner product with $\mathbf{v}$. For example, in the case of hybrid, the scoring function becomes

$$\mathbf{e}_{u,t} = \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{V}\mathbf{f}_u + \mathbf{b}), \;\; t = u-\tau, \cdots, u+\tau. \quad (22)$$

Now, we have C column vectors $[\mathbf{e}_{u,u-\tau}, \cdots, \mathbf{e}_{u,u+\tau}]$ where each vector is of dimension n. Stacking them column-wise, we have an $n \times C$ scoring matrix $\mathbf{E}$

$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}_{u,u-\tau} & \mathbf{e}_{u,u-\tau+1} & \dots & \mathbf{e}_{u,u+\tau} \\ | & | & & | \end{bmatrix}_{n \times C}. \quad (23)$$

Let $e_{u,t}(j) \in (-1, 1)$ be the j^{th} component of the vector $\mathbf{e}_{u,t}$. To compute $\alpha_{u,t}(j)$ from $e_{u,t}(j)$, we normalize $e_{u,t}(j)$ across t (columns) keeping j (row) fixed. Thus, $\alpha_{u,t}(j)$ is computed as

$$\alpha_{u,t}(j) = \frac{\exp(e_{u,t}(j))}{\sum_{t'=u-\tau}^{u+\tau} \exp(e_{u,t'}(j))}, \quad j = 1, \cdots, n. \quad (24)$$

Since exp(.) and tanh(.) are both one-to-one functions, their composition is also one-to-one. Thus, there is a one-to-one correspondence between the input $g_t(j)$ and output $\alpha_{u,t}(j)$ through the composite function. Consequently, $\alpha_{u,t}(j)$ can be interpreted as the amount of contribution of $g_t(j)$ in computing $c_u(j)$. Now, from Eq. (24), we know the values of the vectors $\boldsymbol{\alpha}_{u,t}$, $t \in [u-\tau, \; u+\tau]$. Hence, under the COMA formulation, the context vector $\mathbf{c}_u$ can be computed from $\boldsymbol{\alpha}_{u,t}$ and $\mathbf{g}_t$ using

$$\mathbf{c}_u = \text{Annotate}(\boldsymbol{\alpha}_u, \mathbf{g}, \gamma) = \gamma \sum_{t=u-\tau}^{u+\tau} \boldsymbol{\alpha}_{u,t} \odot \mathbf{g}_t, \quad (25)$$

where $\odot$ is the Hadamard product. One attractive feature of the COMA formulation is that it does not introduce any additional training parameters.

Finally, we highlight the differences between the attention mechanism in this work and in [5]. First, we apply attention across time (past, present, future) on the time convolution features extracted from the final layer of the recurrent network (encoder). Moreover, we attend only to a small context window. In contrast, [5] attends to the entire output sequence of the encoder in addition to the state of the decoder. There is no time convolution applied on the encoder sequence either. Second, to improve attention modeling, we make use of the logit from the previous time $\mathbf{z}_{u-1}$ (or $\mathbf{z}_{u-1}^{\text{LM}}$) as an additional input to our attention block. The attention mechanism in [5] does not make use of logit due to the presence of an explicit decoder. Finally, our COMA formulation yields additional gains without introducing any additional training parameters. There is no such formulation in [5].

IV. Self-Attention CTC

In this section, we investigate another attention-based paradigm known as Self-Attention (SA) [42] in the context of CTC training. There are some key differences in the way the attention weights are computed between SA-CTC and Attention CTC (Section III). In Attention CTC, the attention weights are computed using the hidden features and the output prediction from the previous time step ($\mathbf{z}_{u-1}$). This is evident from the scoring function in Eq. (18). In contrast, in SA-CTC, the weights are computed from the hidden features only. It does not use any past output predictions. Another difference is that the attention weights are computed using additive operations in Attention CTC whereas multiplicative operations (inner products) are used in SA-CTC. Moreover, matrix-vector multiplications used in Attention CTC are computationally slower than performing inner products in SA-CTC.

We highlight only the most important steps in the formulation of SA-CTC. First the hidden features are converted into input projections using the projection matrix $\mathbf{W}_p$ as

$$\mathbf{b}_t = \mathbf{W}_p \mathbf{h}_t, \quad t = u-\tau, \cdots, u+\tau \quad (26)$$

where u denotes the current time step. The inputs to the attention block of SA-CTC consists of three kinds of vectors - keys, values, and a query. These are derived using

$$\mathbf{q}_t = \mathbf{Q}\mathbf{b}_t, \quad t = u, \quad (27)$$

$$\mathbf{k}_t = \mathbf{K}\mathbf{b}_t, \quad t = u-\tau, \cdots, u+\tau, \quad (28)$$

$$\mathbf{v}_t = \mathbf{V}\mathbf{b}_t, \quad t = u-\tau, \cdots, u+\tau, \quad (29)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are the query, key, and value matrices respectively. Here, the dimensions of $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t$ are d_k, d_k, d_v respectively. Note that while there is a single query vector corresponding to the current time step u, there are multiple key and value vectors corresponding to the context window $[u-\tau, u+\tau]$.

Following this, scores are evaluated between the query and the keys by taking their dot products and scaling them with $\frac{1}{\sqrt{d_k}}$. This is given by

$$e_{u,t} = \frac{\mathbf{q}_u^T \mathbf{k}_t}{\sqrt{d_k}}, \quad t = u-\tau, \cdots, u+\tau. \quad (30)$$

The scores reflect the correlation between the current input and the neighboring inputs. These scores are then converted into probabilities (attention weights) using the softmax operation. Linear combination of the value vectors using these attention weights generates a context vector $\mathbf{c}_u$ as follows:

$$\alpha_{u,t} = \frac{\exp(e_{u,t})}{\sum_{t'=u-\tau}^{u+\tau} \exp(e_{u,t'})}, \quad t = u-\tau, \cdots, u+\tau \quad (31)$$

$$\mathbf{c}_u = \sum_{t=u-\tau}^{t=u+\tau} \alpha_{u,t} \mathbf{v}_t. \quad (32)$$

This is followed by a residual connection [52] and layer normalization, i.e., LayerNorm($\mathbf{c}_u + \mathbf{b}_u$). The output of this is fed to a single layer feed-forward network which is followed by another round of residual connection and layer normalization. This is the uni-head attention architecture of SA-CTC since it computes a scalar weight α_t for the entire value vector $\mathbf{v}_t$. This can be easily extended to multi-head attention where $\mathbf{v}_t$ is fragmented into smaller sub-vectors and each sub-vector is weighted using a distinct scalar weight. For more details on SA architecture, readers may refer [42].

V. Hybrid CTC

In this and the next section, our primary motivation is to mitigate the OOV issue of the A2W model as mentioned in Section I.

First, we describe the Hybrid CTC network. The Hybrid CTC network uses a word CTC as the primary task and a letter CTC as the auxiliary task in an MTL framework. The output units of the word CTC correspond to frequently used words and an OOV token. Infrequent words in the training set are lumped together and tagged as OOV. Given an input sequence of features, the word and letter CTCs emit a word and letter sequence respectively. If the word sequence contains a list of frequent words, then the letter sequence from the letter CTC is completely ignored. However, if the word sequence contains the OOV token, the letter CTC is consulted at the segment that generated the OOV token. In the consultation process, the letter sequence from the letter CTC is merged to form a word. Finally, this newly constructed word from the letter CTC is used to replace the OOV token. Since the word CTC and letter CTC are time synchronized through the shared hidden layers of the MTL network, it is possible to find a correspondence between the outputs of the two CTCs. An illustration of this method is shown in Fig. 5. Here, the word CTC generates the sequence "*play artist OOV*". The word sequence generated after merging the letters from the letter CTC is "*play artist ratatat*". Since the segment containing "*ratatat*" from the letter CTC has the most time overlap with the segment containing "*OOV*" from the word CTC, the OOV token is replaced with "*ratatat*". Thus, the final output of the Hybrid CTC is "*play artist ratatat*".

The detailed steps for building the Hybrid CTC model are described as follows:

- Build an LSTM-CTC model of L layers with its output units mapped to frequently occurring words in the training corpus. Map all the remaining infrequent words (occurring less than N times) as the OOV token. Thus, the output units in this LSTM-CTC model correspond to (a) the frequent words, (b) the OOV token, and (c) blank and silence (two additional tokens).
- Freeze the bottom $L-1$ hidden layers of the word-CTC, add one LSTM hidden layer and one softmax layer to build a new LSTM-CTC model with letters as its output units.
- During testing, generate the word output sequence using greedy decoding. If the output word sequence contains an OOV token, replace the OOV token with the word generated from the letter CTC that has the largest time overlap with the OOV token.

VI. Mixed-unit CTC

In this section, we briefly explain multi-letter CTC and compare the past implementation of multi-letter CTC with ours. Based on this foundation, we then explain our proposed Mixed-unit CTC.

Although single-letter units in CTCs perform well, they are prone to high degree of variability across training examples due to their short temporal context. As we will see later in Table IV, multi-letter units tend to perform better than single-letter units since they have low degree of variability by capturing context information and thereby offer more stability during training. Improving letter CTCs can help improve the accuracy of word CTCs. For example, a stronger letter CTC can lower the WER of the Hybrid CTC since the OOV token

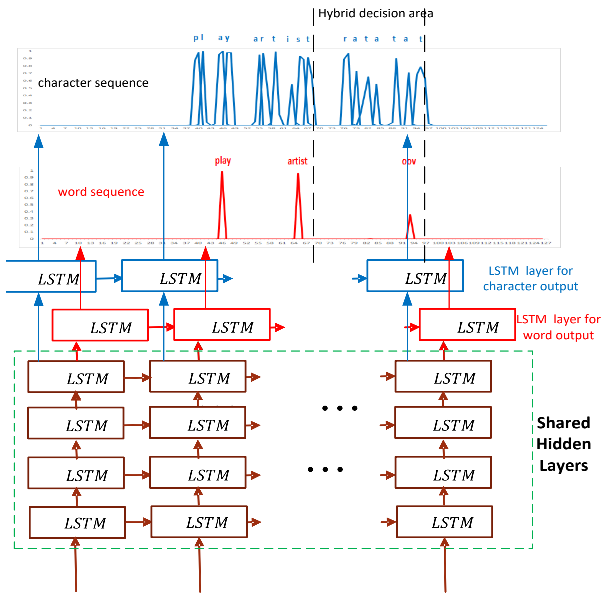


Fig. 5. An example of how the Hybrid CTC solves the OOV issue of the word CTC. The words "play", "artist", "OOV" are obtained from the word CTC. The words "play", "artist", "ratatat" are obtained from the letter CTC. Hence, the final output of Hybrid CTC is "play, artist, ratatat" with the first two words obtained from the word CTC and the last word obtained from the letter CTC.

may be replaced by more precise words generated by the letter CTC.

Gram CTC [29] and multi-phone CTC [53] are multi-letter CTCs based on letters and phonemes respectively. They allow variable number of letters (or grams) and phonemes to be output at each time step. The size of the units in gram CTC and multi-phone CTC are learned automatically with the modified forward-backward algorithm accounting for all decompositions. However, in the test phase, their decoding procedure is more complex than the simple greedy decoding procedure used in single-letter CTC models. To reduce the decoding complexity, the authors in [54] proposed phone synchronous decoding. In contrast, we offer a facile implementation of our multi-letter CTC. We simply decompose every word (which includes both frequent and OOV words) into a sequence of one or more letter units. Examples are shown in the first three rows of Table I where each word, frequent or OOV, is decomposed into single-letter or double-letter or triple-letter units. The advantages of doing this are three-fold. First, our decomposition is straightforward. Second, it does not change the CTC forward-backward algorithm. Finally, during the test phase, our method is able to retain the same greedy decoding procedure used in single-letter CTC models.

In Hybrid CTC, the shared-hidden-layer constraint is used to aid the time synchronization of word outputs between the word and letter CTCs. However, the blank symbol dominates most of the frames. The unit boundaries from CTC is also notorious for being arbitrary. Therefore, time synchronization may not be very reliable with the two CTCs running in parallel. A direct solution is to forgo the MTL framework and train a single CTC model comprising of a mixture of frequent words and letters. The letters arise as a result of decomposing the

TABLE I
EXAMPLES OF HOW WORDS ARE DECOMPOSED INTO DIFFERENT OUTPUT UNITS. "NEWYORK" IS A FREQUENT WORD WHILE "NEWYORKABC" IS AN OOV (INFREQUENT WORD).

Decomposition Type	newyork	newyorkabc
All words → single-letter	n e w y o r k	n e w y o r k a b c
All words → double-letter	ne wy or k	ne wy or ka bc
All words → triple-letter	new yor k	new yor kab c
All words → word	newyork	OOV
OOVs only → single-letter	newyork	n e w y o r k a b c
OOVs only → word+single-letter	newyork	newyork a b c
OOVs only → word+triple-letter	newyork	newyork abc

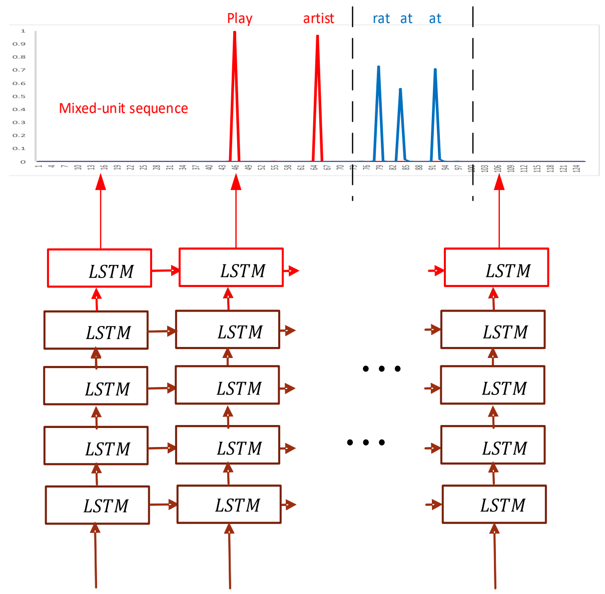


Fig. 6. An example of how a single CTC trained with a mixture of words and letters solves the OOV issue. The final output of this CTC is "play, artist, rat at at".

infrequent words in the training set into letters before CTC training begins. The working of this CTC is illustrated in Fig. 6. If the word is a frequent word, then we just keep it in the output token list. If it is an OOV, then we decompose it into a letter sequence. As shown in the fifth row of Table I, the OOV "newyorkabc" is decomposed into "n e w y o r k a b c" for single letter decompositions. However, the word "newyork" is not decomposed any further because it is a frequent word. Therefore, the output units of the CTC are both words (for frequent words) and letters (for OOVs).

However, we note that artificially decomposing OOVs into sequences of single-letters only may confuse CTC training because the network output modeling units are frequent words and letters. To solve such a potential issue, we decompose the OOVs into a combination of frequent words and letters. We refer to this combination as *mixed units*. For example, in the last two rows of Table I, the OOV "newyorkabc" is decomposed into "newyork a b c" if we use words and single-letter units or "newyork abc" if we use words and triple-letter units. In addition, for mixed units, we use "$" to separate each word in the sentence. For example, the sentence "have you been to newyorkabc" is decomposed into "$ have $ you $ been $ to $ newyork abc $". The "$" symbol acts as a word separator (like the space symbol) and is essential for finding word boundaries of the mixed-units. During training, since the OOVs are decomposed into mixed units, there is no "OOV" output node in the Mixed-unit CTC model. Consequently, during testing, the model emits mixed units instead of "OOV" while still emitting frequent words.

VII. EXPERIMENTS

In this section, we compare the performance of the proposed CTCs with the baseline CTC. We evaluated the proposed methods using Microsoft's Cortana voice assistant task. The training and test sets consist of approximately 3400 hours ($\sim$ 3.3 million utterances) and 6 hours ($\sim$ 5600 utterances) of audio spoken in American English respectively.

All CTC models were trained using either unidirectional LSTMs (ULSTM) or bidirectional LSTMs (BLSTM). The ULSTM is a 5-layer LSTM with 1024 memory cells in each layer. Similarly, the BLSTM is a 6-layer LSTM with 512 memory cells in each direction (therefore resulting in 1024 output dimensions when combining outputs from both directions). The cell outputs are linearly projected to 512 dimensions. The base feature vector is a 80-dimensional vector containing log filterbank energies computed every 10 ms. Eight frames of base features were stacked together ($m = 80 \times 8 = 640$) as the input to the unidirectional CTC, while three frames were stacked together ($m = 80 \times 3 = 240$) as the input to the bidirectional CTC. The skip rate for both unidirectional and bidirectional CTCs was three frames as in [26]. The dimension n of vectors $\mathbf{h}_t, \mathbf{g}_t, \mathbf{c}_u$ was set to 512. For decoding, the greedy decoding procedure (no complex beam search decoder or external LM) was used. This makes our E2E ASR systems purely all-neural.

We focus on letter CTC first and then move on to word CTC. This is because improvements in the letter CTC increase the accuracy of the word CTC especially when encountering an OOV word during test time. Thus, we evaluated the performance of Attention CTC (Section III), SA-CTC (Section IV), and multi-letter CTC using letter units. Then, we evaluated the performance of our proposed Hybrid CTC (Section V) and Mixed-unit CTC (Section VI) using both word and letter units.

A. Experiments With Letter-Based CTCs

We experimented with different sizes of letter units. The sizes are represented by the cardinality K of the label set (defined in Section II-A). For single-letter units, K was set to 30. This corresponds to 26 English letters [a-z], ', *, $, and a blank symbol. For double and triple-letter units, K was set to 763 and 8939 respectively covering all double-letter and triple-letter occurrences in the training set.

1) Attention CTC (Section III): In the first set of experiments, we evaluated Vanilla CTC [11] and the proposed Attention CTC models trained using our 5-layer ULSTM with single-letter units. We experimented with $\tau = 4$ (length of one-sided attention window, defined in Section III-A) considering the training efficiency with this setting. The results are tabulated in the second column of Table II. The top row

TABLE II
WERs of letter-based Vanilla CTC [11] and Attention CTC for $\tau = 4$ trained with 5-layer ULSTM or 6-layer BLTSM and single-letter output units. Relative WER reduction and the number of model parameters are in parentheses. TC = Time Convolution, CA = Content Attention, HA = Hybrid Attention, PLM = Pseudo Language Model, COMA = Component Attention, M = million.

E2E Model	WER (%)	
	ULSTM	BLSTM
Vanilla CTC	24.03 (0.00, 24.12 M)	17.84 (0.00, 35.13 M)
Attention CTC		
+TC (Sec III-A)	21.89 (08.91, 26.48 M)	17.67 (0.95, 37.49 M)
+CA (Sec III-B)	20.45 (14.90, 26.74 M)	16.13 (09.59, 37.75 M)
+HA (Sec III-B)	20.27 (15.65, 27.00 M)	16.01 (10.26, 38.01 M)
+PLM (Sec III-C)	19.78 (17.69, 29.62 M)	15.34 (14.01, 40.63 M)
+COMA (Sec III-D)	**18.57 (22.72, 29.62 M)**	**14.47 (18.89, 40.63 M)**

TABLE III
WERs of letter-based Vanilla CTC [11] and SA-CTC (1, 4, 8 heads) for $\tau = 4$ trained with ULSTM/BLSTM and single-letter units. Relative WER reduction and the number of model parameters are in parentheses. M = million.

E2E Model	WER (%)	
	ULSTM	BLSTM
Vanilla CTC	24.03 (0.00, 24.12 M)	17.84 (0.00, 35.13 M)
SA-CTC (1 head)	20.06 (16.52, 30.70 M)	15.69 (12.05, 41.91 M)
SA-CTC (4 heads)	18.90 (21.35, 30.71 M)	14.98 (16.03, 41.92 M)
SA-CTC (8 heads)	**18.85 (21.56, 30.72 M)**	**14.88 (16.59, 41.93 M)**

TABLE IV
WERs of letter-based CTC models, trained using 6-layer BLSTMs, with Multi-letter output units. Three structures are evaluated: Vanilla CTC [11], Attention CTC, and Attention CTC sharing 5 hidden layers with word CTC. One-sided attention window size (τ) set to 4.

E2E Model	Total Units	WER (%)		
		Vanilla	Attention	Attention + 5 layers sharing
single-letter	30	17.84	14.47	16.74
double-letter	763	15.37	12.16	14.00
triple-letter	8939	**13.28**	**11.36**	**12.81**

presents the WER for Vanilla CTC. All subsequent rows under “Attention CTC” present the WER for the proposed Attention CTC models when attention modeling capabilities were gradually added in a stage-wise fashion. The best proposed model is in the last row. It includes component attention (COMA) along with all the other enhancements above it (i.e., TC, HA, PLM). It may be recalled, from Eq. (19), that hybrid attention (HA) is a combination of both content and location attention. The best proposed model outperformed Vanilla CTC by 22.72% relative. We found that the gains are marginal when going from CA to HA. Our conjecture is that the benefits of adding location information in HA could become more pronounced with smaller frame sizes and larger attention windows. However, smaller frame sizes lead to an exponential increase in the number of CTC paths resulting in instability during CTC training.

Next we evaluated Attention CTC models trained with our 6-layer BLSTM. The results are tabulated in the third column of Table II. Similar to the unidirectional case, the best proposed model outperformed Vanilla CTC by 18.89% relative. This shows that the proposed Attention CTC models continue to perform well even with stronger baselines like BLSTMs.

As an additional experiment, we compared RNN-T models trained with 5-layer ULSTM or 6-layer BLSTM transcription networks along with 1-layer ULSTM prediction network and 30 letters as output units. The transcription networks have the same structure as our baseline CTC models. We observed 21.07% and 16.96% WER for ULSTM and BLSTM transcription networks respectively. While this outperforms the baseline CTC error rates reported in Table II, it could not outperform our final Attention CTC model (last row in Table II).

2) Self-Attention CTC (Section IV): In the next set of experiments, we evaluated the performance of SA-CTC models using our ULSTM and BLSTM with attention window size $\tau = 4$. We used 1024-dimensional vectors for both key/query and value vectors. Thus, $d_k = 1024, d_v = 1024$. This is in accordance with the number of memory cells used in Attention CTC. We experimented with other dimensions but they performed worse. Furthermore, we experimented with both single and multi-head attention (4 and 8 heads). The results are tabulated in Table III. SA-CTC with 8 heads performed the best for each case. The relative WERR over Vanilla CTC are 21.56% and 16.59% using ULSTM and BLSTM respectively. Comparing the best models from Attention CTC and SA-CTC, we find that Attention CTC performed slightly better than SA-CTC by about 1.2% (22.72-21.56) and 2.3% (18.89-16.59) for ULSTM and BLSTM respectively.

3) Multi-letter CTC: In the next set of experiments, we evaluated the performance of various CTC models trained using our 6-layer BLSTM with multi-letter units as outputs. We evaluated three kinds of CTC models: Vanilla CTC, Attention CTC, and Attention CTC sharing 5 hidden layers with a word CTC. In the third CTC model, we applied attention only to the letter CTC.

As shown in the third column of Table IV, the WER of Vanilla CTC drops significantly when the output units become larger (and hence more stable). The letter CTC using triple-letter units achieved 13.28% WER which is a relative WERR of 25.56% compared to the letter CTC using single-letter units.

As shown in the fourth column of Table IV, Attention CTC improves hugely over the Vanilla CTC. It achieves about 18.89%, 20.88%, and 14.46% relative WERR over Vanilla CTC using single-letter, double-letter, and triple-letter units respectively.

In the last column of Table IV, the shared Attention CTC performed better than the Vanilla CTC but worse than its non-sharing counterpart. This indicates one shortcoming of the shared Attention CTC – it sacrifices the accuracy of the letter CTC because of the shared-hidden-layer constraint with the word CTC.

B. Experiments With Word-Based CTCs

In this section, we evaluate the performance of the Hybrid CTC (Section V) and the Mixed-unit CTC (Section VI) using both words and letters as targets. We refer to these CTCs as word CTCs since a majority of the output nodes in these CTCs directly correspond to words. We are primarily interested in

TABLE V
WERs of word-based Vanilla CTC [11] and Hybrid CTC models trained with 6-layer BLSTMs. Hybrid CTC uses a word CTC and an Attention CTC outputting multi-letter units. Attention models are based on Section III.

E2E Model	WER (%)
Vanilla CTC	9.84
Hybrid CTC: word + double-letter Attention CTC	9.66
Hybrid CTC: word + triple-letter Attention CTC	9.66

TABLE VI
WERs of word-based CTCs with different kinds of output units. All CTC models were trained with 6-layer BLSTMs. Attention models are based on Section III.

E2E Model	WER (%)
Vanilla CTC	9.84
Mixed (OOV $\rightarrow$ single-letter)	20.10
Mixed (OOV $\rightarrow$ word + single-letter)	10.17
WPM	9.73
Mixed (OOV $\rightarrow$ word + double-letter)	9.58
Mixed (OOV $\rightarrow$ word + triple-letter)	9.32
Mixed (OOV $\rightarrow$ word + triple-letter) + Attention	8.65

TABLE VII
Summary of WERs of CD phoneme CTC, Vanilla CTC [11], and Mixed-unit CTC + Attention. All CTC models were trained with 6-layer BLSTMs. Attention models are based on Section III.

Model	LM	WER(%)
1. Conventional: CD phoneme CTC	✓	9.28
2. E2E: Vanilla CTC	✗	9.84
3. E2E: Mixed-unit + Attention CTC	✗	8.65
	#3 vs #1	#3 vs #2
Relative WERR	6.79%	12.09%

recognizing the OOVs as accurately as possible while also boosting the accuracy of recognizing non-OOVs. All attention models in this section are based on Attention CTC (Section III) instead of SA-CTC (Section IV) owing to the superior results of the former (Section VII-A2).

Our Vanilla CTC [11] is a 6-layer BLSTM with approximately 27k output nodes consisting of frequent words and the OOV token. We defined frequent words as those which occurred at least 10 times in the training corpus. All the remaining words were tagged as OOV. This is the mapping scheme described in the fourth row of Table I. Thus, within the family of word CTCs, the Vanilla CTC is a CTC with 6-layer BLSTM whose output units model words and the OOV token. The Vanilla CTC achieved 9.84% WER (Table V) among which the OOVs contributed to 1.87% WER.

1) Hybrid CTC (Section V): Our Hybrid CTC model has both word and letter CTCs operating in parallel in an MTL framework. They share 5 hidden BLSTM layers. An additional LSTM layer was added for each task (word and letter CTC) and fine tuned. Thus, the underlying structure of Hybrid CTC is still a 6-layer BLSTM which has the same number of hidden layers as that of the Vanilla CTC. Results are tabulated in Table V. Both hybrid models achieved 9.66% WER which is a marginal improvement over the Vanilla CTC. Several factors contribute to such a small improvement. First, the shared-hidden-layer constraint degrades the performance of the letter CTC, potentially affecting the final hybrid system performance. Second, although the shared-hidden-layer constraint helps to synchronize the word outputs from the word and letter CTC, we still observed that the time synchronization can fail at times. In such cases, the OOV token was replaced with its neighboring word because of word segment misalignments. Because of these factors, the triple-letter CTC did not improve over the double-letter CTC.

2) Mixed-unit CTC (Section VI): In the next set of experiments, we compared the performance of CTCs by changing their output units to mixed-letter units or wordpieces [55], [56]. Wordpieces are commonly occurring sub-word units that can be merged to form whole words. Similar to mixed-units, wordpieces offer the flexibility to generate open-vocabulary words. Previous studies [20], [57] have explored using wordpieces. To build a wordpiece model (WPM), each word in a training corpus is first segmented into a sequence of individual characters and an end-of-word symbol. Following this, the most frequently occurring character pair is merged to form a new symbol or wordpiece. This process is iterated until a predefined number of wordpieces are generated. The outcome of this is that the corpus is now redefined using those wordpieces which result in minimal number of whole word segmentations. However, our approach of building mixed-units is different from building wordpieces since we decompose *only* OOVs while still retaining the high frequency words as whole word units.

Results are tabulated in Table VI. As before, the Vanilla CTC achieved a WER of 9.84%. In the next experiment, we decomposed only the OOVs in the training set into single-letters. Thus, the output nodes consist of both single-letters and 27k frequent words. There indeed is no such clear boundary of decomposition with 2 distinct sets of basic units. As mentioned in Section VI, having a mixture of word and single-letters confuses CTC training as the network does not know why the frequent words cannot be decomposed into letters. Therefore, this model achieved 20.10% WER which is far worse than Vanilla CTC. Analyzing the posterior spikes of this model, we observed that the word spikes and letter spikes are interspersed with each other which proves our hypothesis.

However, when we decomposed OOVs into mixed-units (frequent word + single-letters), the WER dropped sharply to 10.17% but still a little worse than the Vanilla CTC. This is again because of the mixture of words with single-letters. Next, we decomposed the OOVs into a combination of frequent words and double-letters. The WER dropped further to 9.58%. When triple-letters and frequent words were used (totally 33k outputs), the WER dropped even more to 9.32%. This is a 5.28% relative WERR over Vanilla CTC. Then we applied attention on this model. To save computational costs, because of large number of output units, we excluded the PLM network in Eq. (20). This model achieved a WER of 8.65%, which is about 12.09% relative WERR over the Vanilla CTC. This is our final word CTC model (mixed-units with triple-letters + attention). As an additional experiment, instead of mixed-units, we used wordpieces as targets. This model achieved a WER of 9.73% which is a little better than that achieved with Vanilla CTC but worse than the results obtained with mixed-unit CTC. This indicates that building A2W models using mixed-units or WPMs is a better choice than simply using words and OOV (as in Vanilla CTC).

Finally, we compared our final word CTC model with a traditional CD phoneme CTC in Table VII. We trained a CD phoneme 6-layer BLSTM with the CTC criterion, modeling around 9000 tied CD phonemes. It has the same structure as other CTC models except that it uses different output units (phonemes instead of mixed-units or words). This CD phoneme CTC model achieved 9.28% WER when decoding with a well-trained 5-gram LM with totally around 100 million (M) N-grams. Despite a strong CD phoneme CTC model and LM, the mixed-unit + Attention CTC model (without any LM or complex decoder) was still able to outperform it by about 6.79% relative.

Note that the proposed model not only reduces the WER of the word CTC but also improves the end-user experience. The proposed model provides more meaningful outputs without outputting any OOV token which can be distracting to users. Moreover, we observed that even when the proposed model failed to recognize the OOVs accurately, it still came out with words which were a close match with the ground truth words. For example, the proposed method recognizes “text fabine” as “text fabian” and “call zubiate” as “call zubiat”. However, the Vanilla CTC recognized these words as “text OOV” and “call OOV” respectively.

VIII. Conclusions

We proposed improving letter and word CTC models using Attention CTC, Self-Attention CTC, Hybrid CTC, and Mixed-unit CTC. In attention-based CTCs, we generated new hidden features that carry attention weighted context information which are more useful than hidden features without context information. To solve the OOV issue in word CTC, we presented Hybrid CTC which uses a word and letter CTC as primary and auxiliary tasks in an MTL framework. Finally, to boost the performance of Hybrid CTC, we introduced Mixed-unit CTC whose output units contain both words and multi-letters. While the frequent words are treated as whole word units, the OOVs are decomposed into a sequence of frequent words and multi-letters. We evaluated all these methods on a 3400 hours Microsoft Cortana voice assistant task. The proposed word-based Mixed-unit CTC model with triple letters when combined with attention improved over the word-based Vanilla CTC model by 12.09% relative. Such an acoustic-to-word CTC model is a pure end-to-end model without using any LM and complex decoder. It also outperformed a traditional CD phoneme CTC model equipped with strong LM and complex decoder by 6.79% relative.

IX. Code

The CNTK script for Attention CTC described in Section III is available online at: https://github.com/microsoft/CNTK/tree/vadimma/CTC/Examples/Speech/AttentionCTC.

Acknowledgment

The authors would like to thank Kastubh Kalgaonkar, during his time at Microsoft, for his help in building WPMs.

References

[1] D. Yu and J. Li, “Recent Progresses in Deep Learning Based Acoustic Models,” *IEEE/CAA J. of Autom. Sinica.*, vol. 4, no. 3, pp. 399–412, Jul. 2017. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7974889

[2] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, “Learning Acoustic Frame Labeling for Speech Recognition with Recurrent Neural Networks,” in *Proc. ICASSP*. IEEE, 2015, pp. 4280–4284.

[3] Y. Miao, M. Gowayyed, and F. Metze, “EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding,” in *Proc. ASRU*. IEEE, 2015, pp. 167–174.

[4] 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, pp. 4960–4964.

[5] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, “A Comparison of Sequence-to-Sequence Models for Speech Recognition,” in *Proc. Interspeech*, 2017, pp. 939–943.

[6] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, D. Seetapun, A. Sriram *et al.*, “Exploring Neural Transducers for End-to-End Speech Recognition,” in *Proc. ASRU*, 2017, pp. 206–213.

[7] H. Sak, M. Shannon, K. Rao, and F. Beaufays, “Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence To Sequence Mapping,” in *Proc. Interspeech*, 2017.

[8] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, “Flat-Start Single-Stage Discriminatively Trained HMM-Based Models for ASR,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 11, pp. 1949–1961, Jun. 2018.

[9] C.-C. Chiu, T. N. Sainath *et al.*, “State-of-the-art Speech Recognition with Sequence-to-Sequence Models,” in *Proc. ICASSP*, 2018.

[10] T. N. Sainath, C.-C. Chiu, R. Prabhavalkar, A. Kannan, Y. Wu, P. Nguyen, and Z. Chen, “Improving the Performance of Online Neural Transducer Models,” in *Proc. ICASSP*, 2018, pp. 5864–5868.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” in *Proc. Int. Conf. in Learning Representations*, 2006, pp. 369–376.

[12] A. Graves and N. Jaitley, “Towards End-to-End Speech Recognition with Recurrent Neural Networks,” in *Proc. of Machine Learning Research*, 2014, pp. 1764–1772.

[13] 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,” in *Proc. Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[14] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” in *ICLR*, 2015.

[15] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brake, and Y. Bengio, “End-to-End Attention-Based Large Vocabulary Speech Recognition,” in *Proc. ICASSP*, 2016, pp. 4945–4949.

[16] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-Based Models for Speech Recognition,” in *Conf. on Neural Information Processing Systems*, 2015.

[17] A. Graves, “Sequence Transduction with Recurrent Neural Networks,” in *Proc. ICML*, 2012.

[18] K. H. C. L. Lu, X. Zhang and S. Renals, “A Study of the Recurrent Neural Network Encoder-Decoder for Large Vocabulary Speech Recognition,” in *Proc. Interspeech*, 2015, pp. 3249–3253.

[19] H. Soltau, H. Liao, and H. Sak, “Neural Speech Recognizer: Acoustic-to-word LSTM Model for Large Vocabulary Speech Recognition,” in *Proc. Interspeech*, 2017, pp. 3707–3711.

[20] 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.

[21] R. Masumura, T. Tanaka, T. Moriya, Y. Shinohara, T. Oba, and Y. Aono, “Large Context End-to-End Automatic Speech Recognition via Extension of Hierarchical Recurrent Encoder-Decoder Models,” in *Proc. ICASSP*, 2019.

[22] N. Moritz, T. Hori, and J. L. Roux, “Triggered Attention for End-to-End Speech Recognition,” in *Proc. ICASSP*, 2019.

[23] P. Bahar, A. Zeyer, R. Schluter, and H. Ney, “On Using 2D Sequence-to-Sequence Models for Speech Recognition,” in *Proc. ICASSP*, 2019.

[24] H. Xiang and Z. Ou, “CRF-Based Single-Stage Acoustic Modeling with CTC Topology,” in *Proc. ICASSP*, 2019.

[25] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, “Deep Speech: Scaling up End-to-End Speech Recognition,” *arXiv:1412.5567*, 2014.

[26] H. Sak, A. Senior, K. Rao, and F. Beaufays, “Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition,” in *Proc. Interspeech*, 2015, pp. 1468–1472.

[27] N. Kanda, X. Lu, and H. Kawai, “Maximum a posteriori Based Decoding for CTC Acoustic Models,” in *Proc. Interspeech*, 2016, pp. 1868–1872.

[28] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, “Advances in All-Neural Speech Recognition,” in *Proc. ICASSP*, 2017, pp. 4805–4809. [Online]. Available: https://www.microsoft.com/en-us/research/publication/advances-neural-speech-recognition/

[29] H. Liu, Z. Zhu, X. Li, and S. Satheesh, “Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling,” in *Proc. ICML*, 2017, pp. 2188–2197.

[30] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo, “Direct Acoustics-to-Word Models for English Conversational Speech Recognition,” in *Proc. Interspeech*, 2017, pp. 959–963.

[31] J. Li, G. Ye, R. Zhao, J. Droppo, and Y. Gong, “Acoustic-to-Word Model Without OOV,” in *Proc. ASRU*. IEEE, 2017.

[32] J. Li, R. Zhao *et al.*, “Developing Far-Field Speaker System via Teacher-Student Learning,” in *Proc. ICASSP*, 2018.

[33] I. Bazzi, “Modelling Out-Of-Vocabulary Words for Robust Speech Recognition,” Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[34] B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq, “Transcription of Out-Of-Vocabulary Words in Large Vocabulary Speech Recognition Based on Phoneme-to-Grapheme Conversion,” in *Proc. ICASSP*, vol. 1, 2002, pp. I–861.

[35] A. Yazgan and M. Saraclar, “Hybrid Language Models for Out of Vocabulary Word Detection in Large Vocabulary Conversational Speech Recognition,” in *Proc. ICASSP*, vol. 1, 2004, pp. I–745.

[36] M. Bisani and H. Ney, “Open Vocabulary Speech Recognition with Flat Hybrid Models,” in *Proc. Interspeech*, 2005, pp. 725–728.

[37] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny, “Building Competitive Direct Acoustics-to-Word Models for English Conversational Speech Recognition,” in *Proc. ICASSP*, 2018.

[38] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum Learning,” in *Proc. International Conference on Machine Learning*, 2009.

[39] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, “Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets,” in *Proc. ICASSP*, 2013.

[40] Z. Chen, Q. Liu, H. Li, and K. Yu, “On Modular Training of Neural Acoustics-to-word Model for LVCSR,” in *Proc. ICASSP*, 2018, pp. 4754–4758.

[41] A. Das, J. Li, R. Zhao, and Y. Gong, “Advancing Connectionist Temporal Classification with Attention Modeling,” in *Proc. ICASSP*, 2018, pp. 4769–4773.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Proc. NIPS*, 2017.

[43] R. Caruana, “Multitask Learning,” *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul 1997.

[44] M. L. Seltzer and J. Droppo, “Multi-Task Learning in Deep Neural Networks for Improved Phoneme Recognition,” in *Proc. ICASSP*, 2013, pp. 6965–6969.

[45] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, “Advancing Acoustic-to-Word CTC Model,” in *Proc. ICASSP*, 2018, pp. 5794–5798.

[46] L. Lu, X. Zhang, and S. Renais, “On Training the Recurrent Neural Network Encoder-Decoder for Large Vocabulary End-to-End Speech Recognition,” in *Proc. ICASSP*, 2016, pp. 5060–5064.

[47] 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.

[48] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, “Advances in Joint CTC-Attention Based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM,” in *Proc. Interspeech*, 2017, pp. 949–953.

[49] S. Toshniwal, H. Tang, L. Liu, and K. Livescu, “Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition,” in *Proc. Interspeech*, 2017, pp. 3532–3536.

[50] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent Neural Networks Based Language Model,” in *Proc. Interspeech*, 2010, pp. 1045–1048.

[51] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[52] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proc. IEEE CVPR*, 2016.

[53] O. Siohan, “CTC Training of Multi-Phone Acoustic Models for Speech Recognition,” in *Proc. Interspeech*, 2017, pp. 709–713.

[54] Z. Chen, Y. Zhuang, Y. Qian, and K. Yu, “Phone Synchronous Speech Recognition with CTC Lattices,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 1, pp. 86–97, Jan. 2017.

[55] R. Senrich, B. Haddow, and A. Birch, “Neural Machine Translation of Rare Words with Subword Units,” in *Proc. ACL*, 2016, pp. 1715–1725.

[56] Y. Wu, M. Schuster, Z. Chen, , Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,” *arXiv:1609.08144*, 2016.

[57] W. Chan, Y. Zhang, Q. Le, and N. Jaitly, “Latent Sequence Decompositions,” in *Proc. ICLR*, 2017.