Sequence to Sequence Training of CTC-RNNs with Partial Windowing

Kyuyeon Hwang KYUYEON.HWANG@GMAIL.COM
Wonyong Sung WYSUNG@SNU.AC.KR
Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul, 08826 Korea

Abstract

Connectionist temporal classification (CTC) based supervised sequence training of recurrent neural networks (RNNs) has shown great success in many machine learning areas including end-to-end speech and handwritten character recognition. For the CTC training, however, it is required to unroll (or unfold) the RNN by the length of an input sequence. This unrolling requires a lot of memory and hinders a small footprint implementation of online learning or adaptation. Furthermore, the length of training sequences is usually not uniform, which makes parallel training with multiple sequences inefficient on shared memory models such as graphics processing units (GPUs). In this work, we introduce an expectation-maximization (EM) based online CTC algorithm that enables unidirectional RNNs to learn sequences that are longer than the amount of unrolling. The RNNs can also be trained to process an infinitely long input sequence without pre-segmentation or external reset. Moreover, the proposed approach allows efficient parallel training on GPUs. Our approach achieves 20.7% phoneme error rate (PER) on the very long input sequence that is generated by concatenating all 192 utterances in the TIMIT core test set. In the end-to-end speech recognition task on the Wall Street Journal corpus, a network can be trained with only 64 times of unrolling with little performance loss.

1. Introduction

Supervised sequence learning is a regression task where the objective is to learn a mapping function from the input sequence $\mathbf{x}$ to the corresponding output sequence $\mathbf{z}$ for all $(\mathbf{x}, \mathbf{z}) \in S$ with the given training set S, where $\mathbf{x}$ and $\mathbf{z}$ can have different lengths. When combined with recurrent neural networks (RNNs), supervised sequence learning has shown great success in many applications including machine translation (Bahdanau et al., 2014; Sutskever et al., 2014; Cho et al., 2014), speech recognition (Fernández et al., 2007; Graves et al., 2013; Graves & Jaitly, 2014; Hannun et al., 2014; Bahdanau et al., 2015; Chorowski et al., 2015; Chan et al., 2015; Maas et al., 2015; Miao et al., 2015; Hwang & Sung, 2016), and handwritten character recognition (Graves et al., 2008; Frinken et al., 2012). Although several attention-based models have been introduced recently, connectionist temporal classification (CTC) (Graves et al., 2006) is still one of the most successful techniques in practice, especially for end-to-end speech and character recognition tasks (Graves & Jaitly, 2014; Hannun et al., 2014; Maas et al., 2015; Miao et al., 2015; Hwang & Sung, 2016; Graves et al., 2008; Frinken et al., 2012).

The CTC based sequence training is usually applied to bidirectional RNNs (Graves & Schmidhuber, 2005), where both the past and the future information is considered for generating the output at each frame. However, the output of the bidirectional RNNs is available after all of the frames in the input sequence are fed into the RNNs because the future information is backward propagated from the end of the sequence. Therefore, the bidirectional RNNs cannot be employed for low-latency online applications such as incremental speech recognition (ISR) (Fink et al., 1998; Hwang & Sung, 2016). On the other hand, unidirectional RNNs only make use of the past information and are suitable for the low-latency applications at the cost of a little accuracy loss. Moreover, the CTC-trained unidirectional RNNs do not need to be unrolled (or unfolded) at the test time. It is shown by Graves et al. (2012) that CTC can also be employed for sequence training of unidirectional RNNs on a phoneme recognition task. In this case, the unidirectional RNN also learns the suitable amount of the output delay that is required to accurately process the input sequence. Hwang et al. (2015) reports that when a CTC-trained unidirectional RNN is employed for online spoken term detection, the detection latency becomes around 200 ms, which is similar to human response time to speech stimuli (Fry, 1975).

Proceedings of the 33^{rd} *International Conference on Machine Learning*, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

For the CTC training of both unidirectional and bidirectional RNNs, it is required to unroll the RNNs by the length of the input sequence. By unrolling an RNN N times, every activations of the neurons inside the network are replicated N times, which consumes a huge amount of memory especially when the sequence is very long. This hinders a small footprint implementation of online learning or adaptation. Also, this "full unrolling" makes a parallel training with multiple sequences inefficient on shared memory models such as graphics processing units (GPUs), since the length of training sequences is usually not uniform, and thus a load imbalance problem occurs. This load imbalance problem can be solved by grouping training sequences with similar lengths into buckets (Chan et al., 2015; Sutskever et al., 2014). However, it is difficult to achieve high parallelism with this approach, when the training sequences are very long. For unidirectional RNNs, this problem can be addressed by concatenating sequences to make a very long stream of sequences, and training the RNNs with synchronized fixed-length unroll-windows over multiple training streams (Chen et al., 2014; Hwang & Sung, 2015). However, it is not straightforward to apply this approach to the CTC training, since the standard CTC algorithm requires full unrolling for the backward variable propagation, which starts from the end of the sequence.

In this paper, we propose an expectation-maximization (EM) based online CTC algorithm for sequence training of unidirectional RNNs. The algorithm allows training sequences to be longer than the amount of the network unroll. Moreover, it can be applied to infinitely long input streams with roughly segmented target sequences (e.g. only with the utterance boundaries and the corresponding transcriptions for training an end-to-end speech recognition RNN). It was shown that the resulting RNN can run continuously without pre-segmentation or external reset and useful for the continuous spoken term detection (Hwang et al., 2015) and the low-latency ISR system with tree-based online decoding (Hwang & Sung, 2016), where the input speech is infinitely long. Due to the fixed unroll amount, we expect that the proposed algorithm is suitable for online semi-supervised learning or adaptation systems with constrained hardware resource. Furthermore, the approach can directly be combined with the GPU based parallel RNN training algorithm described in Hwang & Sung (2015). For evaluation, we present examples of end-to-end speech recognition on the Wall Street Journal (WSJ) corpus (Paul & Baker, 1992) with continuously running RNNs.[1] Experimental results show that the proposed online CTC algorithm performs comparable to the almost fully unrolled CTC training even with the small unroll amount that is shorter than the average length of the sequences in the training set. Also, the reduced amount of unroll allows more parallel sequences to be trained concurrently with the same memory use, which results in greatly improved training speed on a GPU.

The paper is organized as follows. In Section 2, the standard CTC algorithm is explained. Section 3 contains the definition of the online sequence training problem and proposes the online CTC algorithm. In Section 4, the algorithm is extended for the continuously running RNNs, which is followed by its parallelization in Section 5. In Section 6, the proposed algorithm is evaluated with speech recognition examples. Concluding remarks follow in Section 7.

2. Connectionist Temporal Classification

The CTC algorithm (Graves et al., 2006; 2012) uses a many-to-one sequence-to-sequence mapping function that converts the sequence of labeling with timing information (e.g. frame-wise output labels from RNNs) into the shorter sequence of labels by removing timing and alignment information. The main idea is to introduce the additional CTC blank label, b, for the sequence that has timing information, and remove the blank labels and merging repeating labels to obtain the unique corresponding sequence.

Specifically, for the set of target labels, L, and its extended set with the additional CTC blank label, $L' = L \cup \{b\}$, the path, π, is defined as a sequence over L', that is, $\pi \in L'^T$, where T is the length of the input sequence, $\mathbf{x}$. Then, the output sequence, $\mathbf{z} \in L^{\leq T}$, is represented by $\mathbf{z} = \mathcal{F}(\pi)$ with the sequence to sequence mapping function $\mathcal{F}$. $\mathcal{F}$ maps any path π with the length T into the shorter sequence of the label, $\mathbf{z}$, by first merging the consecutive same labels into one and then removing the blank labels. Therefore, any sequence of the raw RNN outputs with the length T can be decoded into the shorter labeling sequence, $\mathbf{z}$, with ignoring timings. This enables the RNNs to learn the sequence mapping, $\mathbf{z} = \mathcal{G}(\mathbf{x})$, where $\mathbf{x}$ is the input sequence and $\mathbf{z}$ is the corresponding target labeling for all $(\mathbf{x}, \mathbf{z})$ in the training set, S. More specifically, the gradient of the loss function $\mathcal{L}(\mathbf{x}, \mathbf{z}) = -\ln p(\mathbf{z}|\mathbf{x})$ is computed and fed to the RNN through the softmax layer (Bridle, 1990), of which the size is $|L'|$.

The CTC algorithm employs the forward-backward algorithm for computing the gradient of the loss function, $\mathcal{L}(\mathbf{x}, \mathbf{z})$. Let $\mathbf{z}'$ be the sequence over L' with the length of $2|\mathbf{z}| + 1$, where $z'_u = b$ for odd u and $z'_u = z_{u/2}$ for even u. Then, the forward variable, α, and the backward variable,

[1] Further experiments are performed on TIMIT (Garofolo et al., 1993) in the supplementary material.

β, are initialized by

$$\alpha(1,u) = \begin{cases} y_b^1 & \text{if } u = 1 \\ y_{z'_1}^1 & \text{if } u = 2 \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(T,u) = \begin{cases} 1 & \text{if } u = |\mathbf{z}'|, |\mathbf{z}'| - 1 \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

where y_k^t is the softmax output of the label $k \in L'$ at time t. The variables are forward and backward propagated as

$$\alpha(t,u) = y_{z'_u}^t \sum_{i=f(u)}^{u} \alpha(t-1,i)$$

$$\beta(t,u) = \sum_{i=u}^{g(u)} \beta(t+1,i) y_{z'_i}^{t+i}, \qquad (2)$$

where

$$f(u) = \begin{cases} u-1 & \text{if } z'_u = b \text{ or } z'_{u-2} = z'_u \\ u-2 & \text{otherwise} \end{cases}$$

$$g(u) = \begin{cases} u+1 & \text{if } z'_u = b \text{ or } z'_{u+2} = z'_u \\ u+2 & \text{otherwise} \end{cases} \qquad (3)$$

with the boundary conditions:

$$\alpha(t,0) = 0, \ \forall t \ , \quad \beta(t, |\mathbf{z}'| + 1) = 0, \ \forall t. \qquad (4)$$

Then, the error gradient with respect to the input of the softmax layer at time t, a_k^t, is computed as

$$\frac{\partial \mathcal{L}(\mathbf{x},\mathbf{z})}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{z}|\mathbf{x})} \sum_{u \in B(\mathbf{z},k)} \alpha(t,u)\beta(t,u), \qquad (5)$$

where $B(\mathbf{z},k) = \{u : \mathbf{z}'_u = k\}$ and $p(\mathbf{z}|\mathbf{x}) = \alpha(T, |\mathbf{z}'|) + \alpha(T, |\mathbf{z}'| - 1)$.

3. Online Sequence Training

3.1. Problem Definition

Throughout the paper, the online sequence training problem is defined as follows.

- The training set S consists of pairs of the input sequence $\mathbf{x}$ and the corresponding target sequence $\mathbf{z}$, that is, $(\mathbf{x},\mathbf{z}) \in S$.
- The estimation model $\mathcal{M}$ learns the general mapping $\mathbf{z} = \mathcal{G}(\mathbf{x})$, where the training sequences $(\mathbf{x},\mathbf{z}) \in S$ are sequentially given.
- For each $(\mathbf{x},\mathbf{z}) \in S$ and at time t, only the fraction of the input sequence up to time t, $\mathbf{x}_{1:t}$, and the entire target sequence, $\mathbf{z}$, are given, where $1 \leq t \leq |\mathbf{x}|$. The length of the input sequence, $|\mathbf{x}|$, is unknown except when $t = |\mathbf{x}|$.

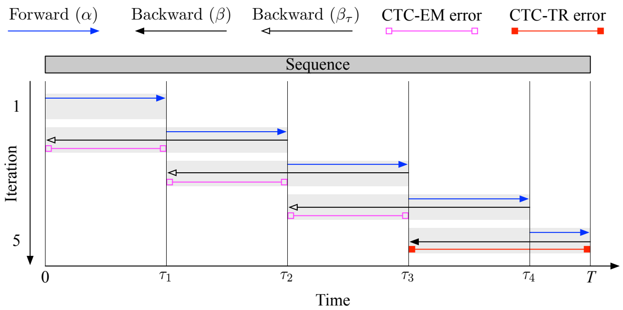

Figure 1. Online CTC($2h'$; h') algorithm depicted for a single sequence that is longer than the RNN unroll amount. The shaded areas indicate the range of the RNN unrolling at each iteration.

- The parameters of the estimation model $\mathcal{M}$ are updated in the manner of online learning, that is, they can be frequently updated even before seeing the entire input sequence $\mathbf{x}$.

This online learning problem usually occurs in real world when a human learns a language from texts and the corresponding audio. For example, when watching movies with subtitles, we are given the entire target sequence (subtitle for the current utterance) and the input sequence (the corresponding audio) up to the current time, t. We cannot access the future audio and even do not know exactly when the utterance will end (at $t = |\mathbf{x}|$).

When RNNs are trained with the standard CTC algorithm, it is difficult to determine how much amount of unrolling is needed before the entire sequence $\mathbf{x}$ is given, since the length of $\mathbf{x}$ is unknown at time $t < |\mathbf{x}|$. Also, it is not easy to learn the sequences that are longer than the unroll amount, which is often constrained by the hardware resources.

3.2. Overview of the Proposed Approach

We propose an online CTC algorithm where the RNN can learn the sequences longer than the unroll amount, h. The algorithm is based on the truncated backpropagation through time (BPTT) algorithm (Werbos, 1990) with the forward step size of h' and the unroll amount of h, which is called BPTT(h; h'), as proposed in Williams & Peng (1990). Algorithm 1 describes the BPTT(h; h') algorithm combined with the CTC loss, where T is the length of the training sequence, $\mathbf{x}$

However, although BPTT(h; h') is designed for online training of RNNs, employing the standard CTC loss function requires full unrolling of the networks. Therefore, we propose the CTC(h; h') algorithm for computing the CTC loss in the online manner as in BPTT(h; h') as in Algorithm 2. The algorithm is also depicted in Figure 1 with the

Algorithm 1 Online CTC training with BPTT(h; h') for a single sequence

```
$\tau_0 \leftarrow 0$
$n \leftarrow 1$
while $\tau_{n-1} < T$ do
    $\tau'_n \leftarrow \max\{1, nh' - h + 1\}$
    $\tau_n \leftarrow \min\{nh', T\}$
    RNN forward activation from $t = \tau_{n-1} + 1$ to $\tau_n$
    CTC($h$; $h'$) error computation on the output layer
    RNN backward error propagation from $t = \tau_n$ to $\tau'_n$
    RNN gradient computation and weight update
    $n \leftarrow n + 1$
end while
```

Algorithm 2 CTC(h; h') at the iteration n

```
$\tau_{n-1} \leftarrow (n-1)h'$
$\tau'_n \leftarrow \max\{1, nh' - h + 1\}$
$\tau'_{n+1} \leftarrow \max\{1, (n+1)h' - h + 1\}$
$\tau_n \leftarrow \min\{nh', T\}$
if $n = 1$ then
    Init. CTC forward variable, $\alpha$, at $t = 1$
end if
CTC forward prop. of $\alpha$ from $t = \tau_{n-1} + 1$ to $\tau_n$
if $\tau_n = T$ then
    Init. CTC-TR backward variable, $\beta$, at $t = T$
    CTC-TR backward prop. of $\beta$ from $t = T$ to $\tau'_n$
    CTC-TR error computation on $t \in [\tau'_n, T]$
else
    Init. CTC-EM backward variable, $\beta_{\tau_n}$, at $t = \tau_n$
    CTC-EM backward prop. of $\beta_{\tau_n}$ from $t = \tau_n$ to $\tau'_n$
    CTC-EM error computation on $t \in [\tau'_n, \tau'_{n+1} - 1]$
    Set error to zero on $t \in [\tau'_{n+1}, \tau_n]$
end if
```

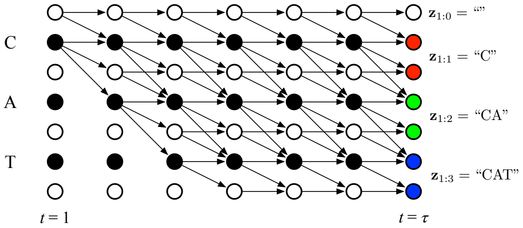

Figure 2. Forward-backward algorithm of CTC-EM for the target sequence “CAT”, where the black and white dots represent the labels and CTC blanks, respectively. The arrows represent the paths with allowed transitions.

example in which the length of the sequence, $T = |\mathbf{x}|$, is 2.5 times as long as the unroll amount.

CTC(h; h') consists of two CTC algorithms. The first one is the truncated CTC (CTC-TR), which is basically the standard CTC algorithm applied at the last iteration with truncation. In the other iterations, the generalized EM based CTC algorithm (CTC-EM) is employed from $t = \max\{1, nh' - h + 1\}$ to $\max\{0, (n+1)h' - h\}$ with the modified backward variable, β_τ. The CTC-TR and CTC-EM algorithms are explained in Section 3.3 and Section 3.4, respectively. Note that simply setting $h = 2h'$ works well in practice. In this setting, we denote the algorithm as CTC($2h'$; h').

3.3. CTC-TR: Standard CTC with Truncation

With the standard CTC algorithm, it is not possible to compute the backward variables when $\tau_n < T$, as the future information beyond τ_n cannot be accessed. Therefore, we only compute the CTC errors at the last iteration, where $\tau_n = T$ as in Algorithm 2. In this case, however, the gradients are only available in the unroll range. Since the backward propagation is truncated at the beginning of the unroll range, we call the CTC algorithm in this range as truncated CTC, or CTC-TR. Also, we call the range that is covered by the CTC-TR algorithm as the CTC-TR coverage.

The RNN can be trained only with CTC-TR if there are sufficient labels that occur within the CTC-TR coverage. However, the CTC-TR coverage decreases by making the unroll amount smaller. Then, the percentage of the effective training frames, which actually generate the output errors, goes down, and the efficiency of training decreases. Also, the effective size of the training set gets smaller, which results in the generalization performance loss of the RNN. Therefore, for maintaining the training performance while reducing the unroll amount, it is critical to make use of the full training frames by employing the CTC-EM algorithm, which is described in Section 3.4.

3.4. CTC-EM: EM-Based Online CTC

Assume that only the fraction of the input sequence, $\mathbf{x}_{1:\tau}$, is available. Then, as shown in the Figure 2, there are $|\mathbf{z}|+1$ possible partial labelings.[2] Let $\mathbf{z}_{1:m}$ be the subsequence of $\mathbf{z}$ with the first m labels. Also we define Z as the set that consists of these labeling sequences:

$$Z = \{\mathbf{z}_{1:m} : 0 \leq m \leq |\mathbf{z}|\}. \tag{6}$$

One of the most simple approach for training the network under this condition is to choose the most likely partial alignment from Z and compute the standard CTC error by regarding the partial alignment as the ground truth labeling. For example, we can select $\mathbf{z}_{1:m'}$ where $m' = \arg\max_m \alpha(\tau, m)$ since $\alpha(\tau, m)$ is a posterior probability $p(\mathbf{z}_{1:m}|\mathbf{x}_{1:\tau}, \mathbf{w}^{(n)})$ with the current network parameter

[2] Although $\mathbf{z}_{1:m}$ is not possible by the standard CTC formulation when $m > \tau$, we can still say that $\mathbf{z}_{1:m}$ is a possible labeling with a probability of zero without loss of generality.

$\mathbf{w}^{(n)}$. This is a well-known hard-EM approach. This simple idea can easily be extended to the more sophisticated soft-EM approach as follows. First, select one of the partial labelings in Z with the probability $p(\mathbf{z}_{1:m}|Z, \mathbf{x}_{1:\tau}, \mathbf{w}^{(n)})$ estimated by the RNN with current parameters (E-step). Then, maximize the probability of that labeling by adjusting the parameters (M-step).

This optimization problem is readily reduced into the generalized EM algorithm. Specifically, the expectation step is represented as

$$\begin{aligned} Q_\tau(\mathbf{w}|\mathbf{x}, \mathbf{z}, \mathbf{w}^{(n)}) &= \mathbb{E}_{\mathbf{z}_{1:m}|Z, \mathbf{x}_{1:\tau}, \mathbf{w}^{(n)}} \left[\ln p(\mathbf{z}_{1:m}|\mathbf{x}_{1:\tau}, \mathbf{w})\right] \\ &= \sum_{m=0}^{|\mathbf{z}|} p(\mathbf{z}_{1:m}|Z, \mathbf{x}_{1:\tau}, \mathbf{w}^{(n)}) \ln p(\mathbf{z}_{1:m}|\mathbf{x}_{1:\tau}, \mathbf{w}), \end{aligned} \tag{7}$$

where $\mathbf{w}^{(n)}$ is the set of the network parameters at the current iteration, n. In the maximization step of the generalized EM approach, we try to maximize Q_τ by finding new parameters $\mathbf{w}^{(n+1)}$ that satisfies $Q_\tau(\mathbf{w}^{(n+1)}|\mathbf{x}, \mathbf{z}, \mathbf{w}^{(n)}) \geq Q_\tau(\mathbf{w}^{(n)}|\mathbf{x}, \mathbf{z}, \mathbf{w}^{(n)})$. As proved in the supplementary material, this is equivalent to the optimization problem where the objective is to minimize the loss function defined as $\mathcal{L}_\tau(\mathbf{x}, \mathbf{z}) = -\ln p(Z|\mathbf{x}_{1:\tau})$. Then, the gradient of the loss function with respect to the input of the softmax layer is

$$\frac{\partial \mathcal{L}_\tau(\mathbf{x}, \mathbf{z})}{\partial a_k^t} = y_k^t - \frac{1}{p(Z|\mathbf{x}_{1:\tau})} \sum_{u \in B(\mathbf{z}, k)} \alpha(t, u)\beta_\tau(t, u), \tag{8}$$

where $p(Z|\mathbf{x}_{1:\tau})$ can be computed by

$$p(Z|\mathbf{x}_{1:\tau}) = \sum_{u=1}^{|\mathbf{z}'|} \alpha(\tau, u) \tag{9}$$

and the backward variable, $\beta_\tau(t, u)$, is initialized as

$$\beta_\tau(\tau, u) = 1, \ \forall u. \tag{10}$$

The new backward variable is propagated using the same recursion in (2), and the error gradients are computed with (5) as in the standard CTC algorithm. See the supplementary material for the derivation of the above equations.

4. Training Continuously Running RNNs

In this section, the proposed online CTC algorithm in Section 3 is extended for training infinitely long streams. The training stream can be naturally very long with the target sequence boundaries, or can be generated by concatenating training sequences in a certain order. When trained on this training stream without external reset of the RNN at the sequence boundaries, the resulting RNN can also continuously process infinitely long input streams without pre-segmentation or external reset. This property has been proved useful for low-latency ISR (Hwang & Sung, 2016) or spoken term detection systems (Hwang et al., 2015) since we can remove the frontend voice activity detector (Sohn et al., 1999) for detecting and pre-segmenting utterances.

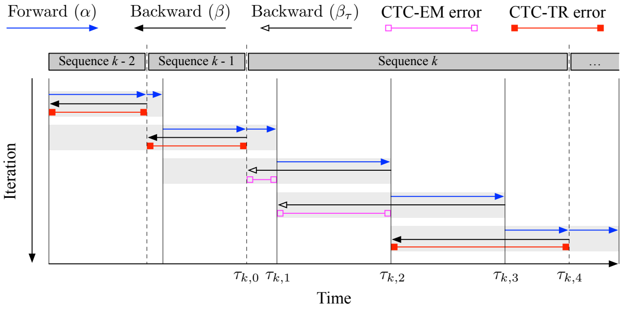

Figure 3. Online CTC($2h'$; h') training with a continuous stream of sequences. The shaded areas indicate the range of RNN unrolling, of which length is $2h'$, at each iteration. The modified segment boundaries for the sequence k, $\tau_{k,n}$, are shown.

The CTC(h; h') algorithm can directly be applied to the infinitely long training streams as shown in Figure 3. When the sequence boundaries are reached during the forward activation, we perform CTC-TR, initialize the forward variable, and process the next sequence with some frame offset. Also, care should be taken on the transition of CTC labels at the boundary. Assume that the last label of the sequence k and the first label of the sequence $k + 1$ are the same. Then, a CTC blank label should be inserted between two sequences since the same labels that occur consecutively in the decoding path are folded into one label. In practice, this folding can easily be prevented by forcing the blank label at the first frame of each sequence by modifying the initialization of the forward variable as follows:

$$\alpha_c(1, u) = \begin{cases} y_b^1 & \text{if } u = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where the subscript c indicates the continuous CTC training.

5. Parallel Training

In a massively parallel shared memory model such as a GPU, efficient parallel training is achieved by making use of the memory hierarchy. For example, computing multiple frames together reduces the number of read operations of the network parameters from the slow off-chip memory by temporarily storing them on the on-chip cache memory and reuse them multiple times. For training RNNs on a GPU, this parallelism can be explored by employing multiple training sequences concurrently (Hwang & Sung, 2015).

The continuous CTC(h; h') algorithm in Section 4 can be directly extended for parallel training with multiple streams. Since the forward step size and the unroll amount is fixed, the RNN forward, backward, gradient computation, and weight update steps can be synchronized over multiple training streams. Thus, the GPU based parallelization approach in Hwang & Sung (2015) can be employed for the RNN training. Although the computations in the CTC(h; h') algorithm are relatively fewer than those of the RNN, further speed-up can be achieved by parallelizing the CTC algorithm similarly.

6. Experiments

6.1. End-to-End Speech Recognition with RNNs

For the evaluation of the proposed approach, we present examples of end-to-end speech recognition with character-level RNN language models (LMs) and tree-based online decoding (Hwang & Sung, 2016). The acoustic RNN is a deep unidirectional long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) with forget gates (Gers et al., 2000) and peephole connections (Gers et al., 2003), which is trained with the online CTC algorithm on the continuous stream of speech. Also, a character-level RNN language model (Sutskever et al., 2011) is employed for tree-based decoding. The system continuously recognizes infinitely-long input speech in realtime without presegmentation.

Specifically, the acoustic RNN has 3 unidirectional LSTM layers, where each layer has 768 LSTM cells and . The output layer is a 31-dimensional softmax layer. Each unit of the softmax layer represents one of the posterior probabilities of 26 alphabet characters, two special characters (. and '), a whitespace character, the end of sentence (EOS) symbol, and the CTC blank label. The input of the network is a 123-dimensional vector that consists of a 40-dimensional log Mel-frequency filterbank feature vector plus energy, and their delta and delta-delta values. The feature vectors are extracted from the speech waveform in every 10 ms with 25 ms Hamming window using HTK (Young et al., 1997). Before being fed into the RNN, feature vectors are element-wisely normalized to the zero mean and the unit standard deviation, where the statistics are extracted from the training set.

The character-level RNN LM consists of 2 unidirectional LSTM layers with 512 cells per layer. The input is a 30-dimensional one-hot encoded vector that represents a current label, and the output is the probabilities of the next labels. The input and output labels are same as the output labels of the acoustic RNN except the CTC blank label. The RNN LM considers the past and current inputs for computing the probabilities of the next labels.

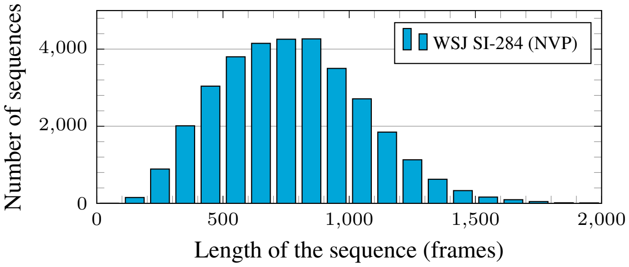

Figure 4. Histogram of the length of the sequences in the WSJ SI-284 training set, where only the utterances with non-verbalized punctuations (NVPs) are considered. The feature frames are extracted with the period of 10 ms.

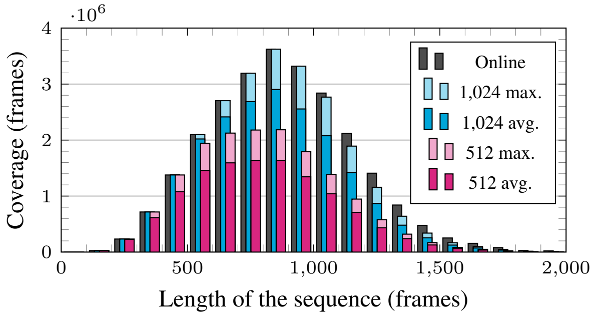

Figure 5. Coverage of the trainable frames with respect to the length of the sequences in the WSJ SI-284 (NVP) training set. The average and maximum coverages of CTC-TR on continuous training streams are visualized for the unroll amount of 512 and 1,024 when CTC($2h'$; h') is applied. Note that the proposed online CTC algorithm (CTC-TR + CTC-EM) covers the entire training frames.

6.2. Wall Street Journal (WSJ) Corpus

The experiments are performed on the Wall Street Journal (WSJ) (Paul & Baker, 1992) corpus. The RNN LM is trained with the text-based language model training data included in the WSJ corpus with the resulting bit-per-character (BPC) of 1.167. For the acoustic RNN training, the subset of the WSJ SI-284 set is used, where only the utterances with non-verbalized punctuations (NVPs) are included, resulting in about 71 hours of utterances. The histogram of the length of the sequences in the training set is shown in Figure 4. Note that the average length of the sequences is 772.5 frames. If we unroll the network over N frames, the sequences longer than N frames will not be fully covered by CTC-TR.

In Figure 5, the CTC-TR coverage is further analyzed with respect to the length of the sequence and the unroll amount.

When the stream of sequences are trained with the continuos CTC algorithm, the CTC-TR coverage varies depending on the frame offsets of CTC(h; h'). The average coverage is calculated assuming that the offset is uniformly distributed. If the probability that a certain frame is included in the coverage is greater than zero, then the frame is included in the maximum coverage. For the experiments, we only consider CTC($2h'$; h'), that is, the unroll amount is twice as much as the forward step size. Then, unrolling the network 1,024 times results in the CTC-TR coverage of 79.48 % on average and 95.69 % at maximum. On the other hand, when the unroll amount is 512, CTC-TR only covers 48.16 % on average and 63.27 % at maximum. Note that the full coverage is achieved when CTC-TR is combined with CTC-EM.

The WSJ Nov'93 20K development set and the WSJ Nov'92 20K evaluation set are used as the development (validation) set and the test (evaluation) set, respectively. For the final evaluation of the network after training, a single test stream is used that is generated by concatenating all of the 333 utterances in the test set.

6.3. Training Procedure

The RNN LM is trained with truncated BPTT(512; 256) on infinite training streams generated by concatenating sentences in the text training data. Note that the EOS symbols are inserted between sentences. The training is performed on a GPU with multiple streams (Hwang & Sung, 2015). We applied ADADELTA (Zeiler, 2012) for annealing and early stopping for preventing overfitting. However, overfitting was not observed in our experiments.

The acoustic RNN are trained on a GPU as in Section 5 with the memory usage constraint. To maintain the memory usage same while changing the unroll amount, we fixed the total amount of unrolling over multiple training streams to 16,384. For example, the number of parallel streams become 8 with the unroll amount of 2,048 and 32 with 512 times of unrolling. The total amount of GPU memory usage is about 9.5 GiB in our implementation.

The performance evaluation of the network is performed at every 10,485,760 training frames (i.e. N continuous training streams with the length of $10,485,760/N$ each) in terms of word error rate (WER) on the 128 parallel development streams of which length is 16,384 each. For this intermediate evaluation, best path decoding (Graves et al., 2006) is employed without the RNN LM for fast computation.

For the online update of the RNN parameters, the stochastic gradient descent (SGD) method is employed and accelerated by the Nesterov momentum of 0.9 (Nesterov, 1983; Bengio et al., 2013). Also, the network is annealed by combining the early stopping technique as follows. If the network performance based on the intermediate evaluation is not improved for 11 consecutive times (10 times of retry), the learning rate is reduced by the factor of 10 and the training is resumed from the second best network. The training starts from the learning rate of 10^{-5} and finishes when the learning rate becomes less than 10^{-7}.

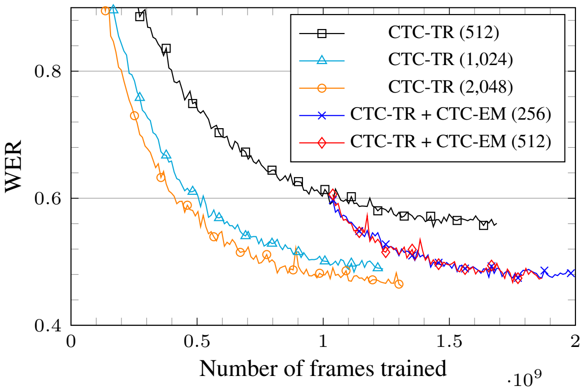

Figure 6. Convergence curves in terms of WER on the development set with the various unroll amounts of 256, 512, 1,024, and 2,048, and the fixed learning rate of 10^{-5}.

The pre-trained network is used for CTC-TR and CTC-EM combined training because the expectation step of CTC-EM requires the RNN to align the target labels in a certain level. The pre-trained networks are obtained by early stopping the CTC-TR training of the networks when the performance is not improved during 6 consecutive intermediate evaluations using the learning rate of 10^{-5}. For the CTC-TR and CTC-EM combined training with the unroll amount of 512, 1,024, and 2,048, the training starts from the pre-trained network that is trained with the same amount of unrolling. Otherwise, for the combined training with the unrolling less than 512 times, we use the pre-trained network with the unroll amount of 512.

6.4. Evaluation

Figure 6 shows the convergence curves in terms of WER on the development set without the RNN LM using various unroll amounts and training algorithms, where the unroll amount is twice the forward step size. The convergence speed of the CTC-TR only training decreases when the unroll amount becomes smaller. This is because the percentage of the effective training frames become smaller due to the reduced CTC-TR coverage. Also, it can be observed that the performance of the CTC-TR only trained network with 512 times of unrolling converges to the worse WER than those of the other networks due to the reduced size of the effective training set. On the other hand, the convergence curves of the CTC-TR and CTC-EM combined

Table 1. Comparison of the CTC-TR coverages, the CER and WERs on the test set, and the training speeds on the GPU with the varying amounts of unrolling.

# Streams	**CTC-TR coverage (%)**		**CER / WER (%)**			**Training speed (frames/s)**	
× **# Unroll**	Average	Maximum	CTC-TR	+ CTC-EM	+ RNN LM	CTC-TR	+ CTC-EM
8 × 2,048	97.84	99.995	-	**10.6 / 38.4**	4.00 / 9.30	3.81 k	3.80 k
16 × 1,024	79.48	95.69	11.2 / 39.1	10.9 / 38.6	4.13 / 9.52	6.79 k	6.60 k
32 × 512	48.16	63.27	13.9 / 47.2	10.9 / 38.8	**3.89 / 8.88**	12.58 k	11.70 k
64 × 256	24.82	33.06	-	11.2 / 39.7	4.08 / 9.53	18.03 k	15.99 k
128 × 128	12.43	16.57	-	11.3 / 40.0	3.89 / 9.20	23.64 k	20.54 k
256 × 64	6.21	8.29	-	11.4 / 40.1	4.41 / 9.85	26.98 k	22.24 k

training with the unroll amounts of 256 and 512 are similar to that of the CTC-TR only training with 2,048 times of unrolling. Considering that the average sequence length of the training set is 772.5 frames, the results are quite encouraging.

The evidence of the similar convergence curves with the different unroll amounts implies that the training can be accelerated under the memory usage constraint by employing more parallel training streams with less unrolling. To examine how much speed-up can be achieved on a GPU, further experiments are performed as in Table 1. The training speed is measured on the system equipped with NVIDIA GeForce Titan X GPU and Intel Xeon E5-2620 CPU. For the final character error rate (CER) and WER report on the test set, the output of the RNN is decoded by the tree-based online CTC beam search (Hwang & Sung, 2016) with and without language models. Note that the experiment with the unroll amount of 2048 is the baseline, where CTC-TR covers most of the training frames and there is little difference from the standard CTC training. As shown in the table, we can achieve a great amount of speedup without sacrificing much WERs. Also, it is possible to train a network with only 64 times of unrolling, which corresponds to 640 ms window, at the cost of 4.5% relative WER when decoded without the RNN LM.

The RNN LM is integrated with a beam width of 512, a beam depth of 50, an LM weight of 2.0, and an insertion bonus of 1.5. When the RNN LM is applied, the baseline network shows 9.30% WER. On the other hand, 8.88% WER is obtained with the acoustic RNN trained with only 512 times of unrolling. However, we consider this improvement is due to the noise in the experimental results. It is observed that the early stopping of training based on the intermediate WER evaluation without the RNN LM does not guarantee the best performance when the decoding is performed with the RNN LM. Nevertheless, it seems there is a slight performance loss when the network is trained with only 64 times of unrolling. Note that 8.9% WER was achieved in Hwang & Sung (2016) with the same network structure. Also, 8.7% and 7.34% WERs were reported in Graves & Jaitly (2014) and Miao et al. (2015), respectively, with bidirectional RNNs for sentence-wise recognition. Our results in Table 1 is reported without any regularization techniques, such as weight noise in Graves & Jaitly (2014) or dropout (Hinton et al., 2012). For fair comparison, we also trained a unidirectional LSTM network with 4 layers, where each layer contains 512 cells, with online CTC(1024; 512) and dropout for RNNs (Zaremba et al., 2014). This model achieves 32.5% WER without LMs, which is comparable to 30.1% WER obtained with the deep bidirectional LSTM network (Graves & Jaitly, 2014).

Further experiments are performed on TIMIT (Garofolo et al., 1993) in the supplementary material, where 20.7% phoneme error rate (PER) is achieved on the very long input speech that is formed by concatenating all the utterances in the core test set.

7. Concluding Remarks

Throughout the paper, the online CTC(h; h') algorithm is proposed for sequence to sequence learning with unidirectional RNNs using partial windows. The algorithm consists of CTC-TR and CTC-EM. CTC-TR is the standard CTC algorithm with truncation and CTC-EM is the generalized EM based algorithm that covers the training frames that CTC-TR cannot be applied. The proposed algorithm allows the unroll amount to be less than the length of the training sequence and is suitable for small footprint online learning systems or massively parallel implementation on a shared memory model such as a GPU. Also, the online CTC algorithm is extended for training continuously running RNNs without external reset, and evaluated in the WSJ and TIMIT experiments. On the WSJ corpus, when the memory capacity is constrained, the proposed approach achieves significant speed-up on a GPU without sacrificing the performance of the resulting RNN much. We expect that further acceleration of training will be possible with lower performance loss when different unroll amounts are used in the pre-training, main training, and annealing stages.

Acknowledgments

We thank the reviewers for the constructive comments. Also, this work was supported in part by the Brain Korea 21 Plus Project and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2015R1A2A1A10056051).

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bahdanau, Dzmitry, Chorowski, Jan, Serdyuk, Dmitriy, Brakel, Philemon, and Bengio, Yoshua. End-to-end attention-based large vocabulary speech recognition. *arXiv preprint arXiv:1508.04395*, 2015.

Bengio, Yoshua, Boulanger-Lewandowski, Nicolas, and Pascanu, Razvan. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8624–8628. IEEE, 2013.

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

Chan, William, Jaitly, Navdeep, Le, Quoc V, and Vinyals, Oriol. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

Chen, Xie, Wang, Yongqiang, Liu, Xunying, Gales, Mark JF, and Woodland, Philip C. Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch. In *Proc. Interspeech*, 2014.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Chorowski, Jan, Bahdanau, Dzmitry, Serdyuk, Dmitriy, Cho, Kyunghyun, and Bengio, Yoshua. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

Fernández, Santiago, Graves, Alex, and Schmidhuber, Jürgen. An application of recurrent neural networks to discriminative keyword spotting. In *Artificial Neural Networks–ICANN 2007*, pp. 220–229. Springer, 2007.

Fink, Gernot, Schillo, Christoph, Kummert, Franz, Sagerer, Gerhard, et al. Incremental speech recognition for multimodal interfaces. In *Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE*, volume 4, pp. 2012–2017. IEEE, 1998.

Frinken, Volkmar, Fischer, Andreas, Manmatha, R, and Bunke, Horst. A novel word spotting method based on recurrent neural networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):211–224, 2012.

Fry, Dennis B. Simple reaction-times to speech and non-speech stimuli. *Cortex*, 11(4):355–360, 1975.

Garofolo, John S, Lamel, Lori F, Fisher, William M, Fiscus, Jonathon G, and Pallett, David S. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93:27403, 1993.

Gers, Felix A, Schmidhuber, Jürgen, and Cummins, Fred. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

Gers, Felix A, Schraudolph, Nicol N, and Schmidhuber, Jürgen. Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Graves, Alex and Schmidhuber, Jürgen. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18 (5):602–610, 2005.

Graves, Alex, Fernández, Santiago, Gomez, Faustino, and Schmidhuber, Jürgen. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376. ACM, 2006.

Graves, Alex, Liwicki, Marcus, Bunke, Horst, Schmidhuber, Jürgen, and Fernández, Santiago. Unconstrained online handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 577–584, 2008.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

Graves, Alex et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

Hannun, Awni, Case, Carl, Casper, Jared, Catanzaro, Bryan, Diamos, Greg, Elsen, Erich, Prenger, Ryan, Satheesh, Sanjeev, Sengupta, Shubho, Coates, Adam, et al. Deepspeech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hwang, Kyuyeon and Sung, Wonyong. Single stream parallelization of generalized LSTM-like RNNs on a GPU. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 1047–1051. IEEE, 2015.

Hwang, Kyuyeon and Sung, Wonyong. Character-level incremental speech recognition with recurrent neural networks. *arXiv preprint arXiv:1601.06581*, 2016.

Hwang, Kyuyeon, Lee, Minjae, and Sung, Wonyong. Online keyword spotting with a character-level recurrent neural network. *arXiv preprint arXiv:1512.08903*, 2015.

Maas, Andrew L, Xie, Ziang, Jurafsky, Dan, and Ng, Andrew Y. Lexicon-free conversational speech recognition with neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

Miao, Yajie, Gowayyed, Mohammad, and Metze, Florian. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *arXiv preprint arXiv:1507.08240*, 2015.

Nesterov, Yurii. A method for unconstrained convex minimization problem with the rate of convergence O (1/k2). In *Doklady AN SSSR*, volume 269, pp. 543–547, 1983.

Paul, Douglas B and Baker, Janet M. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, pp. 357–362. Association for Computational Linguistics, 1992.

Sohn, Jongseo, Kim, Nam Soo, and Sung, Wonyong. A statistical model-based voice activity detection. *Signal Processing Letters, IEEE*, 6(1):1–3, 1999.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Werbos, Paul J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10): 1550–1560, 1990.

Williams, Ronald J and Peng, Jing. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.

Young, Steve, Evermann, Gunnar, Gales, Mark, Hain, Thomas, Kershaw, Dan, Liu, Xunying, Moore, Gareth, Odell, Julian, Ollason, Dave, Povey, Dan, et al. *The HTK book*, volume 2. Entropic Cambridge Research Laboratory Cambridge, 1997.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Zeiler, Matthew D. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.