Training Deep Bidirectional LSTM Acoustic Model for LVCSR by a Context-Sensitive-Chunk BPTT Approach

Kai Chen\textsuperscript{1,2}, Zhi-Jie Yan\textsuperscript{2}, Qiang Huo\textsuperscript{2}

\textsuperscript{1}University of Science and Technology of China, Hefei, China
\textsuperscript{2}Microsoft Research Asia, Beijing, China
\{v-kachen, zhijley, qianghuo@microsoft.com

Abstract

This paper presents a study of using deep bidirectional long short-term memory (DBLSTM) as acoustic model for DBLSTM-HMM based large vocabulary continuous speech recognition (LVCSR), where a context-sensitive-chunk (CSC) backpropagation through time (BPTT) approach is used to train DBLSTM by splitting each training sequence into chunks with appended contextual observations, and a (possibly overlapped) CSCs based decoding method is used for recognition. Our approach makes mini-batch based training on GPU more efficient and reduces the latency of DBLSTM-based LVCSR from a whole utterance to a short chunk. Evaluations have been made on Switchboard-I benchmark task. In comparison with epochwise BPTT training, our method can achieve about three times speed-up on a single GPU card. In comparison with a highly optimized DNN-HMM system trained by a frame-level cross entropy (CE) criterion, our CE-trained DBLSTM-HMM system achieves relative word error rate reductions of 9% and 5% on Eval2000 and RT03S testing sets, respectively.

Index Terms: Long short-term memory, DBLSTM, DNN, LVCSR, Context sensitive chunk, BPTT.

1. Introduction

As a special type of artificial neural networks, recurrent neural networks (RNNs) can model contextual information of a sequence (e.g., [1, 2]), and its bidirectional version (BRNN) [3] provides a framework to utilize future and past information simultaneously at each time instance. These advantages led to its early success in small-scale automatic speech recognition (ASR) tasks (e.g., [1, 2]), but it was difficult to scale up for larger ASR tasks due to the intricacy in training. Recently, deep neural networks (DNNs), which stack multiple feed-forward layers on top of each other, have been successfully used as acoustic models for large vocabulary continuous speech recognition (LVCSR) (e.g., [4, 5] and the references therein). Because DNNs can only provide limited temporal modeling power by feeding a fixed-size sliding window of feature vectors, more powerful model for sequence signal such as RNNs, especially a long short-term memory (LSTM) version (e.g., [6–8]), have attracted the attention of many speech research groups again.

LSTM replaces the neurons in recurrent layers of standard RNN with carefully designed memory blocks to ease training [6–8]. LSTM and its bidirectional version (BLSTM) [9] have been applied to ASR (e.g., [9, 10]) and handwriting recognition (HWR) (e.g., [11]). Combined with connectionist temporal classification (CTC) output layer and trained from unsegmented sequence data by using CTC training [12], it was demonstrated in [11] that BLSTM-based system outperforms a state-of-the-art hidden Markov model (HMM) based system for both offline and online HWR. The same technique has also been applied to ASR and achieved promising results on TIMIT task [12]. Following the success of DNNs for acoustic modeling, (BL)STM layers can also be stacked on top of each other to build deep (BL)STMs for ASR [13]. More recently, DBLSTM with CTC training has been used to build an end-to-end ASR system without leveraging any Gaussian mixture model HMM (GMM-HMM) system [14]. Another way to use DBLSTM for ASR is to combine it with HMM in a hybrid mode [15]. In [16], a hybrid DBLSTM-HMM system gives state-of-the-art results on TIMIT task and outperforms a DNN-HMM system on Wall Street Journal (WSJ) task. However, both TIMIT and WSJ are small to medium scale ASR tasks. In [17, 18], hybrid DBLSTM-HMM systems achieve state-of-the-art results with both frame-level cross entropy (CE) training and sequence-level discriminative training on the LVCSR tasks of voice search and short message dictation. In this paper, we study how DBLSTM-HMM system works for large-scale LVCSR tasks.

Applying DBLSTM to LVCSR faces several challenges. Firstly, DBLSTMs are often trained with an epochwise backpropagation through time (BPTT) algorithm (e.g., [9, 19]), where network states of all time steps of a sequence need to be stored. Nowadays, GPUs are widely used in deep learning by leveraging massive parallel computations via mini-batch based training. When applied to DBLSTM (e.g., [20]), GPU’s limited memory restricts the number of sequences that can be used in a mini-batch, especially for LVCSR tasks with long training sequences and large model sizes. Secondly, full sequence dependence at each time step makes DBLSTM unsuitable for low-latency recognition, because a delay of a whole utterance will be incurred. In [21], a context-sensitive-chunk (CSC) BPTT algorithm, which splits each sequence into chunks with appended contextual observations, is proposed to deal with similar challenges for offline HWR. Since CSC-BPTT can parallelize more chunks and recognition delay is only a short chunk, this technique can also be applied to LVCSR. Another motivation to use CSC-BPTT to train speech DBLSTM is similar to HWR, namely speech feature vectors of a phone are mostly influenced by several phones before and after it. Inspired by ensemble method, we also propose a decoding method with overlapped CSCs to improve recognition accuracy further.

The remainder of this paper is organized as follows. In Section 2, we present our approach. In Section 3, we report experimental results and analyze the effect of different factors of our approach. Finally, we conclude the paper in Section 4.
2. Our Approach

2.1. Hybrid DBLSTM-HMM LVCSR System

Our LVCSR system is based on a DBLSTM-HMM framework as in a neural-network/HMM hybrid system [15], where DBLSTM acts as acoustic model. Frame-level state targets are provided by a forced alignment given by a GMM-HMM system. The activation function of DBLSTM’s output layer is softmax, whose unit number is the total number of HMM states. Different from a DNN-HMM hybrid system, which appends an acoustic context window of frames to either side of the one being classified at each time step, DBLSTM only feeds a single frame at a time since it is able to store past and future states internally. In training, a frame-level cross-entropy objective function is minimized. In recognition, the state-dependent scores derived from the DBLSTM are combined with HMM state transition probabilities and language model (LM) scores to determine the recognition result by using an in-house decoder for LVCSR.

2.2. DBLSTM Training

2.2.1. Context-sensitive-chunk BPTT

Due to its bidirectional interdependence among frames in a sequence, DBLSTM is often trained with epochwise BPTT (e.g., [9, 19, 20]). Given a sequence, this method must accumulate the history of activations in the network over the entire sequence, along with the history of errors, then back-propagation is carried out to calculate gradients. After that, weights are updated accordingly. In order to accelerate training by GPU, mini-batch technique can be used as in DNN training, but the mini-batch here has to be defined over sequences (e.g., [20]). Therefore, for long sequences and large networks, the memory size of GPU restricts the number of parallel sequences in a mini-batch so the acceleration is quite limited.

A simple but effective solution to the limited acceleration problem is to use chunk BPTT [22], which splits sequences into chunks of particular length, and treats these chunks as isolated sequences. However, the lost interdependence among chunks results in performance degradation when the chunk size is small. In [21], a CSC-BPTT algorithm was proposed to address this issue. As shown in Fig. 1, given a sequence, it is firstly split into (possibly overlapped) chunks of fixed length \( N_c \), then \( N_l \) previous frames are appended before each chunk as left context and \( N_r \) future frames after it as right context. For the first/last chunk of each sequence, no left/right contextual frames are appended. This kind of chunk is called context-sensitive-chunk (CSC). The appended frames only act as context and gives no output, so no error signals will be generated during training, as illustrated in Fig. 2. CSCs from all the sequences are pooled together and randomized before every sweep of training. Because CSCs are treated as isolated sequences, this technique increases the number of parallel chunks in a mini-batch, leading to faster training. Moreover, if the length of CSC is short enough, such trained DBLSTM can be applied to low-latency decoding, while it is impossible for DBLSTM trained by traditional epochwise BPTT because of full sequence dependence at each time step.

A CSC with \( N_c \) chunk frames, \( N_l \) left and \( N_r \) right contextual frames is denoted as \( "N_l\cdot N_c\cdot N_r" \) for simplicity. So, for chunk BPTT, its chunk of length \( N_c \) can be represented as \( \text{"0-}N_l\cdot N_c\cdot N_r\text{"} \), and for epochwise BPTT, the chunk configuration is denoted as \( \text{"0-Full+0"} \) because a chunk here is the full sequence. It is noted that CSC-BPTT can also be used to train DLSTM as we demonstrated in [21]. In our implementation, the per-time-step input could be a single frame or a concatenation of a local window of feature vector sequence. The unfolded RNN in [23] is a special case of CSC, whose configuration is \( \text{"N_c-1+0"} \) and per-time-step input consists of current frame and several future frames, but is much more expensive for training and decoding than other configurations with \( N_c >> 1 \).

2.2.2. Learning rate scheduling

Tuning learning rate, especially the scheduling of reducing the learning rates during training, is critical to the performance of deep learning (e.g., [24]). In this study, we have used a semi-automatic learning rate scheduling mechanism similar to the “Newbob+/Train” method in [25]. Our learning rate scheduling approach works as follows: Firstly, a sub-set of the full training data set is reserved as a validation set. It differs from the conventional definition of validation because those data would still be used in training. Secondly, DBLSTM training starts with an empirically chosen initial learning rate. The frame error rate (FER) on the validation set is evaluated once after the model has been updated a fixed number of times. Lastly, the learning rate is adjusted according to the results of adjacent evaluations. If the FER improvement between current and previous evaluations is above a threshold \( \tau \), the learning rate stays unchanged; else if the error rate improvement between previous and antepenultimate evaluations is not above \( \tau \), the learning rate stays unchanged; otherwise, the learning rate will be multiplied by a fixed factor \( \rho \) (\( 0 < \rho < 1 \)).
of learning when shrinking the learning rate does not lead to
efficient FER improvement. In this stage, conventional scheduling
rules would continually cut the learning rate and force the
learning to converge prematurely. Our heuristic essentially pre-
vents the learning rate from shrinking so the network is still able
to learn something in the final training stage.

2.3. CSC-based Decoding

Given a CSC-BPTT trained DBLSTM, an unknown utterance
for recognition should also be split into CSCs of the same con-
figuration as used in training. In a preliminary study, we found
that no difference was made for recognition accuracy whether
the training sequences are split into CSCs with or without over-
lap, therefore we use no-overlap scheme in this paper. For a
testing utterance, we could have options. If computational cost
is a concern, then the testing utterance can be split into non-
overlapping CSCs as in [21]; otherwise, the testing utterance can
be split into overlapped CSCs to achieve higher recognition
accuracy. If overlapped CSCs are used, the t-th frame of observation
may be evaluated \(N_t\) times, whose scores are denoted as
\(\{y_1^{(t)}, \ldots, y_t^{(N_t)}\}\). The final score could be obtained by taking
an arithmetic mean \(y_{t} = \frac{1}{N_t} \sum_{i=1}^{N_t} y_{i}^{(t)}\) or a geometric mean
\(y_{t} = \sqrt[\log(N_t)]{\prod_{i=1}^{N_t} y_{i}^{(t)}}\). The effect of these two averaging methods
will be compared in Section 3.2.3. Because different scores of a
given frame are predicted from several neighboring chunks, the
averaged score leverages more frames as context, which leads to
better recognition accuracy as we will show in next section.

3. Experiments

3.1. Experimental Setup

Switchboard-I conversational telephone speech transcription
task [26] is used for evaluation. About 300 hours of speech from
520 speakers are used in training. About 2 hours of speech
from 2000 Hub5 evaluation (Eval2000) and about 6.3 hours of
speech from Spring 2003 NIST rich transcription set (RT03S)
are used in testing. For front-end spectral feature extraction, 13-
dimensional PLP features along with their time derivatives up
to third order are extracted every 10 ms to form a 52-dimensional
raw feature vector. Windowed mean and variance normalization
is then performed, and a 39 × 52 HLDA transform is esti-
ated afterwards to reduce the feature dimension to 39 such that
GMM-HMM acoustic models are trained with 39-dimensional
feature vectors. A set of speaker independent GMM-HMMs
which contain 9,304 decision-tree tied triphone HMM states
are estimated using maximum likelihood criterion. This GMM-
HMM set is used to perform forced alignment on both training
and testing sets to get the frame-level state labels. In DNN and
DBLSTM training, these labels will be used as ground truth in
training and state classification targets in testing.

A 7-hidden-layer DNN is chosen as the baseline system.
Each hidden layer contains 2,048 rectified linear units (ReLU’s),
resulting in approximately 45 million parameters. Input dimen-
sion of this network is 572, and the input is a concatenation of
11 frames of 52-dimensional raw feature vectors. The frame-
level cross-entropy objective function is minimized via mini-
batch SGD with L2 constraint regularization [27], and 7 sweeps of
training data are conducted with a carefully tuned learning rate
schedule.

All DBLSTMs have 5 hidden layers. Each hidden layer has
512 memory cells (256 for forward and 256 for backward states) with forget gate and peephole connections [16], result-
ing in approximately 11 million parameters. These models will be
optimized by epochwise, chunk or CSC BPTT. Considering
the memory size of a single Nvidia Tesla K20Xm GPU card,
mini-batch sizes are set to be 8 sequences for epochwise BPTT
and 64 chunks for chunk and CSC BPTT. We implement CSC-
BPTT for DBLSTM training based on open-source Current toolkit [20]. During training process, 30hr data is sampled from
training set as validation set. The FER improvement threshold \(r\)
is set to be 2% and learning rate tuning factor \(\beta\) is set to be 0.5;
Evaluation on validation set will be conducted after a sweep of
data and 7 sweeps are used for all the models.

Experiments are conducted for various CSC configurations. For each configuration, initial learning rate is carefully tuned
and the one leading to the best validation set FER is selected
to decode the test sets. Both FER and Word Error Rate (WER)
are used to evaluate different models. The vocabulary, pronun-
ciation lexicon, trigram language models are the same as that
in [28, 29].

3.2. Experimental Results

3.2.1. Comparison with DNN-HMM

Table 1: Performance (in %) comparison on testing sets of
DBLSTM-HMM systems trained by epochwise and CSC BPTT
methods with DNN-HMM system.

<table>
<thead>
<tr>
<th>Config.</th>
<th>Eval2000 FER</th>
<th>Eval2000 WER</th>
<th>RT03S FER</th>
<th>RT03S WER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>FSH</td>
<td>LITERAL</td>
</tr>
<tr>
<td>0-Pail+0</td>
<td>29.7</td>
<td>14.8</td>
<td>44.7</td>
<td>22.7</td>
</tr>
<tr>
<td>21-64+21</td>
<td>29.6</td>
<td>14.7</td>
<td>44.5</td>
<td>22.8</td>
</tr>
<tr>
<td>DNN-HMM</td>
<td>39.9</td>
<td>16.2</td>
<td>55.0</td>
<td>24.0</td>
</tr>
</tbody>
</table>

Experiments are conducted to identify suitable CSC configura-
tions by varying the chunk size and the number of contextual
frames. In recognition, CSC-based decoding without overlap is
used. The effect of CSC configurations on testing performance
(in %) of DBLSTM-HMM systems is compared in Table 2. It
is observed that for CSCs with \(N_c = 64\), a size of contextual frames ranging from 16 to 32 works well, while chunk size \(N_c\)
has a relatively big influence. All the DBLSTM-HMM systems
trained by CSC-BPTT perform better than the DNN-HMM sys-
tem. When no contextual frames are appended to chunks (i.e.,
configuration, the time is shortened to 21.4 hr, namely a 2.8x
BPTT is about 59.9 hr, while for CSC-BPTT with “21-64+21”
In our experiments, the elapsed time per sweep for epochwise
BPTT performs worse than its CSC-BPTT counterpart.
formance significantly, but overall, DBLSTM trained by chunk-
decoding on testing performance (in %) of DBLSTM-HMM
the effect of the number of overlapped frames in chunk-based
decoding with overlapped CSCs. No difference is observed,
CSCs. Table 4 compares two averaging methods for CSC-based decoding with overlapped CSCs. The number of overlapped frames is 48. The DBLSTM is trained by
CSC-BPTT with the configuration of “21-64+21”.
Table 4: Testing performance (in %) comparison of averaging methods for CSC-based decoding with overlapped CSCs. The number of overlapped frames is 48. The DBLSTM is trained by

<table>
<thead>
<tr>
<th>Averaging methods</th>
<th>Eval2000</th>
<th>RT03S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FER</td>
<td>WER</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td>FSH</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td>FSH</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>29.6</td>
<td>14.7</td>
</tr>
<tr>
<td>Geometric</td>
<td>29.6</td>
<td>14.7</td>
</tr>
</tbody>
</table>

Table 5: Effect of the number of overlapped frames in chunk-based decoding on testing performance (in %) of DBLSTM-HMM systems trained by chunk-BPTT with the configuration of “0-64+40”.

<table>
<thead>
<tr>
<th># of overlap frames</th>
<th>Eval2000</th>
<th>RT03S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FER</td>
<td>WER</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td>FSH</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td>FSH</td>
</tr>
<tr>
<td>0</td>
<td>33.5</td>
<td>16.2</td>
</tr>
<tr>
<td>16</td>
<td>31.1</td>
<td>15.3</td>
</tr>
<tr>
<td>32</td>
<td>30.7</td>
<td>15.1</td>
</tr>
<tr>
<td>48</td>
<td>29.8</td>
<td>15.0</td>
</tr>
</tbody>
</table>

4. Conclusion and Discussion
We have demonstrated that DBLSTM-HMM can outperform
DNN-HMM on the Switchboard-I benchmark task if both
DBLSTM and DNN are trained by minimizing a frame-level
CE criterion. It is well-known that bidirectional contextual in-
formation plays an important role in acoustic modeling [30],
and DBLSTM offers an elegant way of modeling and lever-
aging the bidirectional contextual information. Given the co-
articulation effect in a speech utterance, speech features of each phone are mostly influenced by several phones before and after it, therefore there is no need to model the whole utter-
ance by a DBLSTM directly. This insight motivates us to
propose to use a DBLSTM to model a short chunk, a CSC-
BPTT method to train the DBLSTM, and a CSC-based decod-
ing method for DBLSTM-HMM based LVCSR. Compared with
DNN with a fixed-size window of a feature vector sequence as input, DBLSTM can learn automatically the effective length
of input feature vector sequence for acoustic modeling. Com-
pared with standard epochwise BPTT method, our CSC-BPTT
method makes mini-batch based training on GPU more effi-
cient, which can achieve about three times speed-up on a sin-
gle GPU card in our experiments. Furthermore, CSC-based decoding makes low-latency DBLSTM-based LVCSR possi-
ble by incurring only a delay of a short chunk rather than a
whole utterance. Taking a CSC configuration of “21-64+21”
for example, if no frame overlap is used in CSC-based decod-
ing, the delay will be 85 frames or 850 ms, while the computa-
tional complexity will only be about 1.27 times of that using
the DBLSTM trained by epochwise BPTT, which can only be used
for decoding until the whole utterance is observed. In compar-
ison with a highly optimized DNN-HMM system trained by a
frame-level CE criterion, our CE-trained DBLSTM-HMM sys-
tem achieves relative WERRs of 7.4% and 4.2% on Eval2000 and
RT03S testing sets, respectively. If a CSC-based decoding with 48 overlapped frames is used, the above relative WERRs
will become 9.3% and 5.0% respectively with a cost of in-
creased computational complexity. As future work, a compar-
ison of DBLSTM-HMM and DNN-HMM with sequence-level
discriminative training must be made.
5. References


