Improving CTC-based Acoustic Model with Very Deep Residual Time-delay Neural Networks

Sheng Li¹, Xugang Lu¹, Ryoichi Takashima¹, Peng Shen¹, Tatsuya Kawahara¹,², and Hisashi Kawai¹

¹National Institute of Information and Communications Technology, Kyoto, Japan
²Kyoto University, Kyoto, Japan

sheng.li@nict.go.jp

Abstract

Connectionist temporal classification (CTC) has shown great potential in end-to-end (E2E) acoustic modeling. The current state-of-the-art architecture for a CTC-based E2E model is based on a deep bidirectional long short-term memory (BLSTM) network that provides frame-wise outputs estimated from both forward and backward directions (BLSTM-CTC). Since this architecture can lead to a serious time latency problem in decoding, it cannot be applied to real-time speech recognition tasks. Considering that the CTC label of one current frame can only be affected by a few neighboring frames, we argue that using BLSTM traversing on a whole utterance from both directions is not necessary. In this paper, we use a very deep residual time-delay (VResTD) network for the CTC-based E2E acoustic modeling (VResTD-CTC). The VResTD network provides frame-wise outputs with local bidirectional information without needing to wait for the whole utterance. Speech recognition experiments on a Corpus of Spontaneous Japanese were carried out to test our proposed VResTD-CTC and the state-of-the-art BLSTM-CTC model. Comparable performance was obtained while the proposed VResTD-CTC does not suffer from the decoding time latency problem.

Index Terms: Speech recognition, acoustic model, connectionist temporal classification (CTC), very deep residual network

1. Introduction

The connectionist temporal classification (CTC) framework [1] is an effective end-to-end (E2E) framework [2] for speech recognition. The CTC modeling technique greatly simplifies the acoustic modeling pipelines. No frame-level labels or initial GMM-HMM systems are needed anymore. Built on top of the deep bidirectional long short-term memory (BLSTM) recurrent neural networks, CTC models achieve state-of-the-art performance on speech recognition tasks [3]. However, BLSTM traverses a whole utterance from both directions to estimate the frame-wise outputs. This leads to serious time latency and makes the model impractical for online speech recognition (especially on our SprinTra WFST decoder [4]).

The BLSTM model can be improved using a context-sensitive-chunk BLSTM (CSC-BLSTM) [5]. This method chops a whole sentence into several overlapping chunks and controls the delay time by appending several contextual chunks (both past and future). A latency-controlled BLSTM (LC-BLSTM) [6] accelerates the training and decoding speed by optimizing the calculation of each chunk. For the same purpose, Xue et al. [7] proposed other structures: forward approximation and a backward simple recurrent neural network BLSTM (FABSR-BLSTM).

More direct methods focus on replacing the BLSTM network with other no-latency structures. Using a unidirectional LSTM model with a similar parameter size [8] or a state-of-the-art Grid LSTM [9] instead is a very effective strategy. Feed-forward sequential memory neural networks (FSMN) [10], a time-delay neural network (TDNN) [11, 12], a residual memory network (RMN) [13], and a layer-wise context expansion and attention (LACEA) network [14] can learn bidirectional long-term dependency without recurrent feedback and thus have very good potential for integration with CTC training.


When calculating the frame-level CTC label output, the predicted label of a frame is mostly affected by its neighboring short frame trunks (coarticulation effect [18]). For this reason, processing the whole utterance using the BLSTM-CTC is not necessary. In this paper, we propose a very deep residual time-delay CTC neural network (VResTD-CTC). In our method, a very deep residual structure is used to enhance the conventional TDNN during CTC training.

The rest of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes our proposed model’s structure. Section 4 presents experiments. Conclusions and future works are given in Section 5.

2. Related Work

2.1. Connectionist Temporal Classification (CTC)

The CTC training criterion is intensively used in E2E [2] acoustic modeling, which focuses on solving the sequence labeling problem between variable-length speech frame inputs and label outputs (phone character, syllable, etc.).

Output based on a CTC-trained network is a frame-level token sequence called a path (denoted as \(p\)). With multiple hypotheses, many paths are generated where a blank symbol (\(\phi\) as no label output) is included and inserted between CTC labels. These paths are mapped into label sequences by removing the massive blank symbols detected during the decoding and the duplication of identical labels. One label sequence \(Z\) is mapped to multiple CTC label paths (denoted as \(\text{Path}(Z)\)). The likelihood of \(Z\) can be evaluated as a sum of the probabilities of its corresponding CTC paths:

\[
P(Z|X) = \sum_{p \in \text{Path}(Z)} P(p|X),
\]

where \(X\) is the speech frame sequence in the whole utterance and \(p\) is a CTC path corresponding to label sequence \(Z\).

By differentiating Eq. 1, we can backpropagate training errors
and further update the network parameters based on a forward-backward procedure [1]. The weighted finite state transducer (WFST) [19, 3] is used for decoding with CTC models. LFM-MMI training [20] has been applied on various acoustic models, and it has shown promising results [21]. However, these models need frame-level labels for training, which are often estimated using GMM-HMM system, and thus they are not E2E systems.

2.2. Modeling Temporal Context Information in Feedforward Neural Network without Recurrent Feedback

As mentioned in Section 1, TDNN [22] and its various forms [12, 20] can learn long-term dependency without using recurrent feedback and thus might be integrated with CTC training as an alternative to BLSTM.

A feedforward sequential memory network (FSMN) [10, 16], which can also learn long-term dependency in sequential data without recurrent structure, has memory blocks (usually global vectors) to encode the activations for each hidden layer. Since it also uses a time-delay operation to capture the signal at a certain time step, we can consider it a special TDNN.

2.3. From Deep to Very Deep Structure

In the speech recognition field, the very deep residual networks also achieve significant performance [23, 24, 25, 26, 13, 14] over conventional DNNs. The residual structure [27] is a simplified version of the highway structure [28, 6]. The signals can bypass the transformation across multiple layers from residual connections. We can train much deeper time-delay networks equipped with residual connections in a similar way.

3. Proposed CTC Model based on Very Deep Residual Time-Delay Neural Network

In this section, we introduce our proposed VResTD-CTC, which is a very deep residual neural network that enhances the conventional TDNN in CTC training.

Figure 1 is the general architecture of the proposed VResTD-CTC. The network has two kinds of residual blocks stacked together:

1. Conventional residual blocks (ResBlock): Each ResBlock has several layers connected with activations and a residual-skip 1. A time-delay operation is not used. The stacked ResBlocks are located close to the input layer and transform the speech feature to a higher level representation (Table 2).

2. Time-delay residual blocks (TDResBlock): We integrate a time-delay operation (bidirectional) 2 to the TDResBlock. Two global memory blocks (past and future) encode the output signal after every time-delay operation. The residual connection passes the frame of the current time-step across five layers (Table 2).

In the VResTD-CTC network, the entire input sequence of the l-th time-delay hidden layer can be represented as \( H^l = (h^l_1, h^l_2, h^l_3, \ldots, h^l_T) \). Any \( h^l_i \) has the following processing:

\[
\begin{align*}
\text{pipeline:} & \quad h^l_i \xrightarrow{[W^l, b^l]} \tilde{h}^l_i \xrightarrow{\text{time-delay}} \tilde{H}^l_i \xrightarrow{\text{subsampling}} e^l_i \xrightarrow{\text{ReLU}} h^{l+1}_i. \\
\text{First, it is linearly transformed by standard weight matrix} & \quad W^l \text{ and bias } b^l \text{ for layer } l: \\
& \quad \tilde{h}^l_i = W^l(h^l_i) + b^l. \\
\text{Then a time-delay operation captures a} & \quad (N^l_1 + 1 + N^l_2)-\text{length context } \tilde{H}^l_i \text{ at the } l\text{-th time-delay hidden layer for } h^l_i, \\
& \quad \tilde{H}^l_i = (\tilde{h}^l_{i-N^l_1}, \ldots, \tilde{h}^l_{i-1}, \ldots, \tilde{h}^l_{i+N^l_2}). \text{ With subsampling, } \tilde{H}^l_i \approx (\tilde{h}^l_{i-N^l_1}, \tilde{h}^l_{i}, \tilde{h}^l_{i+N^l_2}). \text{ } N^l_1 \text{ and } N^l_2 \text{ are the window sizes of the past and future context (} N^l_1 + 1 + N^l_2 \leq T). \text{ Following [13], we use symmetric context window (} N^l_1 = N^l_2). \text{ We also slightly tuned the past and future contexts. No benefit was identified (Sec. 4.3.2).} \\
\text{The past context} & \quad (\tilde{h}^l_{i-N^l_1}) \text{ and future context } (\tilde{h}^l_{i+N^l_2}) \text{ are encoded with transformations individually and summed up with current feature } (e^l_i) \text{ before activation function.} \\
& \quad e^l_i = a^l_{N^l_1} \odot \tilde{h}^l_{i-N^l_1} + \tilde{h}^l_i + a^l_{N^l_2} \odot \tilde{h}^l_{i+N^l_2}, \\
\text{where the encoding weight for the past context is denoted as} & \quad a^l_{N^l_1} \text{ and the encoding weight for the future context is denoted as } a^l_{N^l_2}. \\
\text{Following vFSMN} & \quad [10, 16], \text{we use vectors instead of scalers here. } \odot \text{ is element-wise multiplication. Our experiments (Sec. 4.3.2) show that the performance would drop dramatically without memory vectors. We also tuned the number of the vectors and used two vectors.} \\
& \quad h^{l+1}_i = \begin{cases} 
\text{the } l\text{-th layer locates inside of a TDResBlock:} & \quad \text{ReLU}(e^l_i) \\
\text{the } l\text{-th layer is the output layer of a TDResBlock:} & \quad \text{ReLU}(e^l_i + h^{l\text{residual}}_i), 
\end{cases}
\end{align*}
\]

\[\text{Figure 1: General network architecture of VResTD-CTC. Memory block encodes output signal after every time-delay and subsampling operation by default.}\]
where \( h_{t+1}^{\text{residual}} \) is the \( t \)-th input frame to the current TDRes-Block. Suppose the block has 5 layers and the output layer is \( t \)-th layer, \( h_{t+1}^{\text{residual}} \) is actually \( h_{t-1}^{\text{residual}} \).

4. Experiments

4.1. Task and Data Descriptions

In this paper, we tested our proposed method on the Corpus of Spontaneous Japanese (CSJ) [29] and used 240-hour lecture recordings as the training set (CSJ-Train) [30, 31]. We used three official evaluation sets (CSJ-Eval01, CSJ-Eval02, and CSJ-Eval03), each of which contained ten lecture recordings [31], to evaluate the speech recognition results. Ten lecture recordings were chosen for validation (CSJ-Dev) during training. We also selected 27.6-hour training data in CSJ (CSJ-Train\text{small}) to train the seed models for warm-start initialization [32] and tuning the parameters.

Table 1: Data Sets of CSJ

<table>
<thead>
<tr>
<th>Component</th>
<th>Structure (26 FC layers)</th>
<th>#Para (35.5M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>72-dim filter-bank features</td>
<td>(24-dim static + ∆ + ∆∆)</td>
</tr>
<tr>
<td>ResBlock 1</td>
<td>3 FC layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[2048 × 2048]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2048 × 2048]</td>
<td></td>
</tr>
<tr>
<td>ResBlock 2</td>
<td>3 FC layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[2048 × 128]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[128 × 2048]</td>
<td></td>
</tr>
<tr>
<td>ResBlock 3</td>
<td>3 FC layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[2048 × 128]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[128 × 2048]</td>
<td></td>
</tr>
<tr>
<td>TDResBlock 1</td>
<td>5 Time-delay layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td>TDResBlock 2</td>
<td>5 Time-delay layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td>TDResBlock 3</td>
<td>5 Time-delay layers</td>
<td>Residual-skip</td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1024 × 2042]</td>
<td></td>
</tr>
<tr>
<td>Fully-connect</td>
<td>2 FC layers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1024 × 2048]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2048 × 2048]</td>
<td></td>
</tr>
</tbody>
</table>

The model, which is only constructed with FC layers connected with ReLU activations, has three ResBlocks (each with three FC layers) and three TDResBlocks (each with five FC layers and five time-delay operations). The time-delay operations are all enhanced with two global [1024 × 1] vectors.

4.3. Model Structure Tuning

4.3.1. Structure of proposed model

We built a set of prototype networks and selected a best 26-layer structure using an evaluation set (CSJ-Dev) shown in Table 2.

We trained the baseline models using CSJ-Train. To start the first baseline model (DNN-HMM-CE), we first trained a GMM-HMM model and then a DNN model with five hidden layers, each of which is composed of 2048 hidden nodes. The output layer has about 8500 nodes that correspond to the tied-triphone states of the GMM-HMM model. We used 72-dim filter-bank features (24-dim static + ∆ + ∆∆), which are mean and variance normalized per speaker, and 11 frames were spliced (5 left, current, 5 right). The DNN model was trained using a standard stochastic gradient descent (SGD) based on the cross-entropy (CE) loss criterion. All were implemented using the Kaldi toolkit (net1) [33, 30].

We trained the baseline CTC models using the EESEN toolkit [3]. The BLSTM-CTC baseline model (BLSTM-CTC) was trained with the same 72-dimensional filter-bank features (24-dim static + ∆ + ∆∆) without splicing. The BLSTM network has five hidden layers, each of which is composed of 320 nodes. We used 28 Japanese context-independent phones (CI-phones) + blank (\( \phi \)) as acoustic units for the CTC training. The third baseline model is a unidirectional LSTM CTC model (ULSTM-CTC) with the same parameter size as the BLSTM-CTC (five hidden layers, each composed of 640 nodes). The right context of the feature is eight frames, and the left context is a zero frame. The subsampling number was set to three.

For decoding, we trained a 4-gram word language model (WLM) from the transcription of 591-hours of CSJ training data. The WLM’s vocabulary size was 98K. We compiled WPOST-based decoding graphs for these models. The CTC-based models were decoded on EESEN decoder [3, 34] with a prior estimated from the training data. The performances are shown in Section 4.4.

4.2. Settings for Training Proposed Network

We trained the proposed CTC models with CNTK [35]. Similar to BLSTM-CTC, its features are a 72-dim non-spliced filter-bank (24-dim static + ∆ + ∆∆). The labels are the same CI-phone label of the two CTC baselines.

We first trained seed models with 27.6-hours of CSJ-Train\text{small} with the CTC loss criterion using a single GPU. Training the seed model costs two days and the WER% on CSJ-Eval01 is 19.3%. Then we used this seed model to train the CTC model using 240-hour training data. The FsAdaGrad algorithm (an implementation of Adam [36]) was used during CTC training. To quicken training with the 240-hour training data (CSJ-Train), we used the block-wise model update filtering (BMUF) distributed training algorithm [37] on 4 Tesla K40m GPUs. The initial learning rate for each frame was 0.00001 and automatically adjusted using validation on CSJ-Dev. The minibatch size was 2048. The number of parallel sequences in the same minibatch was 16. The maximum epoch number was 25. The training took around a week and finished at the 13th epoch. We used the model from the 5th epoch for evaluation.

The model, which is only constructed with FC layers connected with ReLU activations, has three ResBlocks (each with three FC layers) and three TDResBlocks (each with five FC layers and five time-delay operations). The time-delay operations are all enhanced with two global memory blocks.

4.3.2. Number of time-delay layers

The local bidirectional contexts are captured by stacking the TDResBlocks. Following [13], the context-windows of the stacking time-delay operations are shown in Table 3. The local bidirectional context structure is implicitly encoded in the network architecture by stacking the time-delay network structure. The context’s length can be exponentially extended by the
depth of the stacking layers. The width of the captured context is \( \frac{l(l+1)}{2} + 1 \), where \( l \) is the total number (15) of the FC layers in these TDResBlocks.

We trained VResTD-CTC models where the number of stacked time-delay layers ranged from 10 to 20. We also slightly tuned the past and future contexts. No benefit was identified (Table 3). Exhaustively searching for the optimized structure is not the focus of this paper.

Table 3: ASR performance (WER% on CSJ-Dev) by tuning stacked time-delay operations

<table>
<thead>
<tr>
<th>Symmetric</th>
<th>Asymmetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>l=10</td>
<td>13.1</td>
</tr>
<tr>
<td>l=15</td>
<td>11.7</td>
</tr>
<tr>
<td>l=20</td>
<td>12.7</td>
</tr>
<tr>
<td>l=25</td>
<td>12.7</td>
</tr>
<tr>
<td>l=30</td>
<td>13.7</td>
</tr>
</tbody>
</table>

4.3.3. Number of vectors for memory encoding

We tuned the number of vectors for memory encoding purposes, as shown in Table 4.

Table 4: ASR performance (WER% on CSJ-Dev) of VResTD-CTC models with global or local memory encoding

<table>
<thead>
<tr>
<th>Network</th>
<th>Num. MemVecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>WER%</td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td>11.7</td>
</tr>
<tr>
<td></td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>12.0</td>
</tr>
<tr>
<td></td>
<td>13.8</td>
</tr>
</tbody>
</table>

Our experiments showed that the performance dropped dramatically without memory vectors. We trained VResTD-CTC models where the number of memory encoding vectors ranged from 6 to 30, but they cannot outperform the model with two vectors. In our experiment with full training set, two vectors and ten vectors can achieve the same performance. We choose the simpler structure.

4.4. Performance of Speech Recognition

In this subsection, we compared the performance of the proposed acoustic models (VResTD-CTC) on the three CSJ evaluation sets with the baseline systems: DNN-HMM-CE, BLSTM-CTC, and ULSTM-CTC, see Section 4.1. We decoded the CTC model by feeding the network’s scaled log-likelihood output to the EESEN decoder [3]. The decoding setting is the same with BLSTM-CTC and ULSTM-CTC.

Table 5: ASR performance (WER%) of acoustic models

<table>
<thead>
<tr>
<th>Network (CSJ 240 hours)</th>
<th>Eval01</th>
<th>Eval02</th>
<th>Eval03</th>
<th>Ave.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN-HMM-CE</td>
<td>14.4</td>
<td>11.8</td>
<td>13.6</td>
<td>13.9</td>
</tr>
<tr>
<td>BLSTM-CTC [34]</td>
<td>14.4</td>
<td>11.9</td>
<td>15.1</td>
<td>13.8</td>
</tr>
<tr>
<td>ULSTM-CTC</td>
<td>15.9</td>
<td>12.1</td>
<td>16.6</td>
<td>15.2</td>
</tr>
<tr>
<td>VResTD-CTC</td>
<td>15.1</td>
<td>12.0</td>
<td>14.5</td>
<td>13.9</td>
</tr>
</tbody>
</table>

As shown in Table 5, DNN-HMM-CE and BLSTM-CTC have similar performances. Compared to these two baselines, the word error rate (WER%) of VResTD-CTC is about 0.7% higher on CSJ-Eval01. However its performance can still improve on CSJ-Eval02 and CSJ-Eval03 and has comparable averaged performances on these three evaluation sets. Moreover, it outperformed ULSTM-CTC on all three evaluation sets. Its structure difference may explain such performance gaps between TDNN- (including our model) and BLSTM-based models. Every TDNN layer is only fed with the past and future contexts from its lower layers, while BLSTM can also get feedback from its current layer. Our model still has room for improvement.

Concerning the decoding speed, we tested the real-time factors (RTFs) of the above models. The decoding is applied on the same workstation without sharing with other jobs and the feed-forward pass and decoding are all calculated into RTFs. Since BLSTM-CTC cannot be used in the Sprintra WFST decoder [4], we didn’t test its speed. Compared with the DNN-HMM-CE model (RTF=0.70), HCLG decoding graph, the speed of proposed VResTD-CTC is 0.20 on RTF with TLG decoding graph, which matches the 3x faster results reported in [3]. It is slightly slower than the ULSTM-CTC model (RTF=0.15) due to the much deeper layers.

Parameter size is another problem of our proposed model. Although compared to DNN-HMM-CE (38M), the proposed 26-layer model (36M) is not very large. But it remains over three times larger than BLSTM-CTC (11M) and ULSTM-CTC (10M). To compress the model, we applied singular value decomposition (SVD)-based parameter compress to all of the FC-layers [38, 39] and then fine-tuned with CSJ-Trainbase. Empirical evaluations are shown in Table 6. By reducing half of the size of the original very deep model, we have the least performance loss (1.0% of WER%). This loss can be compensated by using [34, 40] or retraining with full training set.

Table 6: ASR performance (Average WER% on CSJ-Eval01/Eval02/Eval03) of acoustic models with different neural network structures

<table>
<thead>
<tr>
<th>Threshold of SVD</th>
<th>1.0 (Full model)</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Para. Size (approx.)</td>
<td>36M</td>
<td>12M</td>
<td>18M</td>
<td>25M</td>
</tr>
<tr>
<td>WER% (Ave.)</td>
<td>13.9</td>
<td>15.7</td>
<td>14.9</td>
<td>15.9</td>
</tr>
</tbody>
</table>

5. Conclusion and Future Work

In this paper, we propose a method that focuses on replacing the BLSTM network with a very deep residual time-delay CTC neural network (VResTD-CTC). It can learn bidirectional long-term dependency without using recurrent feedback and be integrated with CTC training to achieve comparable performance with DNN-HMM-CE and BLSTM-CTC models. In the future, we will further improve this deep structure.

6. Acknowledgements

The authors thank Dr. Naoyuki Kanda for building concrete baseline systems and Mr. Mitsuyoshi Tachimori for his discussions about CTC-based decoding.

7. References


