IMPROVING VERY DEEP TIME-DELAY NEURAL NETWORK WITH VERTICAL-ATTENTION FOR EFFECTIVELY TRAINING CTC-BASED ASR SYSTEMS

Sheng Li[1]*, Xugang Lu*[1]*, Ryoichi Takashima*[1]*, Peng Shen*[1]*, Tatsuya Kawahara*[1,2]*, and Hisashi Kawai*[1]

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

sheng.li@nict.go.jp

ABSTRACT

The very deep neural network has recently been proposed for speech recognition and achieves significant performance. It has excellent potential for integration with end-to-end (E2E) training. Connectionist temporal classification (CTC) has shown great potential in E2E acoustic modeling. In this study, we investigate deep architectures and techniques which are suitable for CTC-based acoustic modeling. We propose a very deep residual time-delay CTC neural network (VResTD-CTC). How to select a suitable deep architecture optimized with the CTC objective function is crucial for obtaining the state of the art performance. Excellent performances can be obtained by selecting deep architecture for non-E2E ASR systems modeling with tied-triphone states. However, these optimized structures do not guarantee to achieve better or comparable performances on E2E (e.g., CTC-based) systems modeling with dynamic acoustic units. For solving this problem and further leveraging the system performance, we introduce the vertical-attention mechanism to reweight the residual blocks at each time step. Speech recognition experiments show our proposed model significantly outperforms the DNN and LSTM-based (both bidirectional and unidirectional) CTC baseline models.

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. For training with large-scale data, this is very convenient. Built on top of the deep bidirectional long short-term memory (BLSTM) recurrent neural networks, CTC models achieve promising performance on speech recognition tasks [3].

In the speech recognition field, very deep convolutional networks can significantly outperform [4, 5, 6, 7] conventional DNNs. Recently, very deep residual time-delay neural networks [8, 9, 10] have been proposed for speech recognition. Compared to conventional shallow time-delay neural networks (TDNN) [11, 12] and feedforward sequential memory networks (FSMN) [13, 14], they can learn longer context dependency without recurrent feedback and thus have no-latency problem that is associated with bidirectional long short-term memory (BLSTM) recurrent neural networks. For this reason, they have excellent potential to be integrated with end-to-end (E2E) training.

In this study, we integrate a very deep residual time-delay neural network (VResTD) with CTC training. The very deep residual structure is used to enhance the conventional TDNN during CTC training. How to select a suitable deep architecture optimized with the CTC objective function is crucial for obtaining the state of the art performance. We observed excellent performances can be obtained by selecting deep architecture for non-E2E (e.g., cross-entropy based DNN-HMM) ASR systems. However, these optimized structures do not guarantee to achieve better or comparable performances on E2E (e.g., CTC-based) systems. In very deep model like ResNet [15], the skipping connection is proposed only for avoiding "vanishing gradient problem" when training, while the importance of each residual block (ResBlock) for the recognition is not considered. Use of skip connection in ResNet is effective for static pattern recognition such as image recognition and speech segment recognition as in DNN-HMM. In E2E modeling, we need to incorporate a dynamic mechanism to consider the importance of the skip connection at each residual block. We propose vertical-attention mechanism to evaluate the importance of each ResBlock in which, importance is expressed as attention weights for each frame when recognition.

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

2. VERY DEEP RESIDUAL TIME-DELAY NEURAL NETWORK

In this section, we introduce our proposed very deep residual time-delay neural network (**VResTD-CTC**). Very deep residual neural networks were built to enhance the conventional TDNN in CTC training (Fig. 1). Compared to the existing cross-entropy trained very deep time-delay neural networks [8, 9, 10], our proposed model is based on the end-to-end (E2E) scheme.

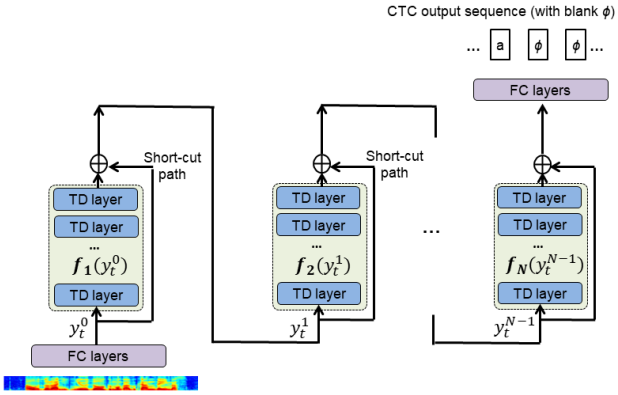

Fig. 1. General network architecture of **VResTD-CTC**

2.1. Operations in a Single Time-delay (TD) Layer

In the VResTD-CTC network, the entire input sequence of the l-th time-delay hidden layer can be represented as (h_1^l, h_2^l, h_3^l, ... , h_T^l). Any h_t^l at time step t has the following processing pipeline:

$$h_t^l \xrightarrow{[W^l:b^l]} \tilde{h}_t^l \xrightarrow[subsampling]{time-delay} \tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}} \xrightarrow[encoding]{memory} e_t^l \xrightarrow{ReLU} h_t^{l+1}.$$

First, it is linearly transformed by standard weight matrix W^l and bias b^l for layer l:

$$\tilde{h}_t^l = W^l(h_t^l) + b^l, \tag{1}$$

Then a time-delay operation captures a (N_1^l+1+N_2^l)-length context $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}}$ at the l-th time-delay hidden layer for $\tilde{h}_t^l$. $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}} = (\tilde{h}_{t-N_1^l}^l, ..., \tilde{h}_t^l, ..., \tilde{h}_{t+N_2^l}^l)$. With subsampling, $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}} \approx (\tilde{h}_{t-N_1^l}^l, \tilde{h}_t^l, \tilde{h}_{t+N_2^l}^l)$. N_1^l and N_2^l are the window sizes of the past and future context (N_1^l+1+$N_2^l \leq T$). Following [10], we use symmetric context window ($N_1^l = N_2^l$). We also slightly tuned the past and future contexts.

The past context ($\tilde{h}_{t-N_1^l}^l$) and future context ($\tilde{h}_{t+N_2^l}^l$) are encoded with transformations individually and summed up with current feature ($\tilde{h}_t^l$) before activation function.

$$e_t^l = \underbrace{a_{N_1^l}^l \odot \tilde{h}_{t-N_1^l}^l}_{subsampled-past} + \tilde{h}_t^l + \underbrace{c_{N_2^l}^l \odot \tilde{h}_{t+N_2^l}^l}_{subsampled-future}, \tag{2}$$

where the encoding weight for the past context is denoted as $a_{N_1^l}^l$ and the encoding weight for the future context is denoted as $c_{N_2^l}^l$. Following vFSMN [13, 14], we use vectors instead of scalers here. $\odot$ is element-wise multiplication. Our experiments [16, 17] show that the performance would drop dramatically without memory vectors. We also tuned the number of the vectors and used two vectors, and no benefit was identified.

$$h_t^{l+1} = \begin{cases} \text{the } l\text{-th layer locates inside of a TDResBlock:} \\ \quad ReLU(e_t^l) \\ \text{the } l\text{-th layer is the output layer of a TDResBlock:} \\ \quad ReLU(e_t^l + h_t^{residual}), \end{cases} \tag{3}$$

where $h_t^{residual}$ is the t-th input frame to the current TDResBlock. Suppose the block has 5 layers and the output layer is l-th layer, $h_t^{residual}$ is actually h_t^{l-4}.

2.2. Stacked Time-delay (TD) Layers in Residual Network

In the VResTD-CTC network, the TD layers are grouped into residual blocks that consist of a multi-layer transformation $\boldsymbol{f}_i$ and a short-cut connection bypassing $\boldsymbol{f}_i$. With y_t^{i-1} as input, the output of the i-th block is recursively defined:

$$y_t^i = \boldsymbol{f}_i(y_t^{i-1}) + y_t^{i-1}, \tag{4}$$

where $\boldsymbol{f}_i$ is a sequence of TD layers and Rectified Linear Units (ReLUs).

These residual blocks that are stacked together behave like ensemble networks [18]. For the proposed network, every output of the last residual block also dynamically includes the outputs of other residual blocks over time. Taking a three-block VResTD-CTC model, for example, the output of the last residual block at time-step t (y_t^3) can be expressed:

$$\begin{aligned} y_t^3 = y_t^0 + \underbrace{\boldsymbol{f}_1(y_t^0)}_{1^{st}-block} + \underbrace{\boldsymbol{f}_2(y_t^0 + \boldsymbol{f}_1(y_t^0))}_{2^{nd}-block} \\ + \underbrace{\boldsymbol{f}_3(y_t^0 + \boldsymbol{f}_1(y_t^0) + \boldsymbol{f}_2(y_t^0 + \boldsymbol{f}_1(y_t^0)))}_{3^{rd}-block} \end{aligned} \tag{5}$$

.

This dynamic nature inspired our proposed method for improving the current model in the next section.

2.3. Reweighting the Paths of a Residual Block with Attention Module

The attention-based end-to-end (E2E) framework has been successfully applied to the speech recognition field [19, 20, 21]. For more generalized network structures (non-recurrent

networks), feedforward attention [22] and self-attention [23] have been proposed. These techniques have also enhanced time-delay neural networks [14, 24].

Compared to these previous works, we propose an attention mechanism to improve the structure of the proposed very deep model. We insert the attention modules into an already trained very deep network and retrain the whole network. The skipping connection was proposed only for avoiding "vanishing gradient problem", and does not consider the importance of each ResBlock for the recognition. On the other hand, the attention mechanism can evaluate the importance of each ResBlock for the recognition and can express the importance as attention weights.

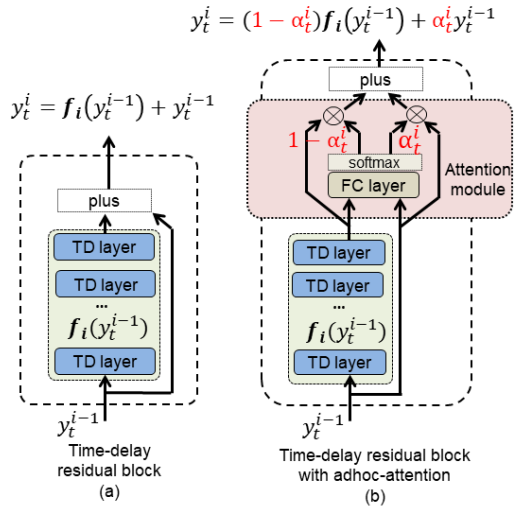

Fig. 2. Attention modules inserted into i-th time-delay residual block

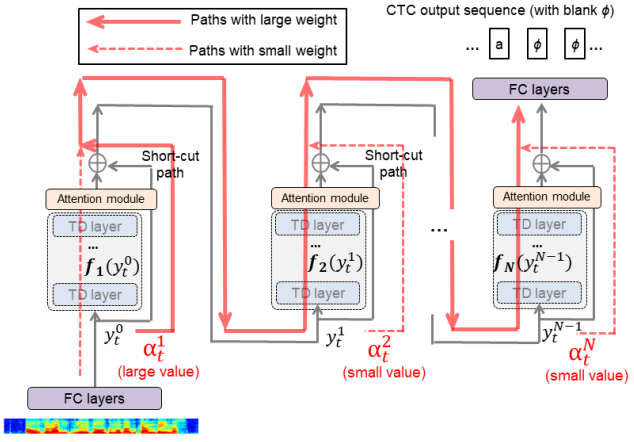

Fig. 3. Attention scores reweights the path inside a very deep residual time-delay network

As shown in Fig. 2 and Fig. 3, the attention module is set after the output layer of every residual block. We use the attention scores $\boldsymbol{\alpha}_t^i$ to reweight the paths of a residual block.

$$\boldsymbol{\alpha}_t^i = \begin{bmatrix} 1-\alpha_t^i \\ \\ \alpha_t^i \end{bmatrix}^T, \tag{6}$$

where α_t^i is a scale factor for the data at time-step t passing through the short-cut path of the i-th block, and the $1-\alpha_t^i$ is the scale factor for the TD layer stackings correspondingly.

The TD layer stackings $\boldsymbol{f}_i(y_t^{i-1})$ and short-cut path y_t^{i-1} in Eq. 4 is reweighted:

$$y_t^i = (1-\alpha_t^i)\cdot \boldsymbol{f}_i(y_t^{i-1}) + \alpha_t^i \cdot y_t^{i-1}, \tag{7}$$

The reweight factors estimated by the block-wise attention modules are defined as follows:

$$\boldsymbol{\alpha}_t^i = Softmax\left(\begin{bmatrix} \boldsymbol{U}_i \cdot \boldsymbol{f}_i(y_t^{i-1}) \\ \\ \boldsymbol{V}_i \cdot y_t^{i-1} \end{bmatrix} + \boldsymbol{b}_i\right), \tag{8}$$

where $\boldsymbol{U}_i$, $\boldsymbol{V}_i$, and $\boldsymbol{b}_i$ are trainable parameters.

We can fix the parameters of the trained model and only retrain the attention modules with the small dataset. The attention scale factors over an evaluation set can show the information of the inner paths of this residual block. Figure 4 illustrates the attention scores of the short-cut paths in a residual block (the first block from the seed model in Section 3.3) on a single utterance in one evaluation set (Section 3.1). The voice activities show different behaviors when they pass the first block in these two systems. For the syllable-based system, the voice segments tended to go through the short-cut, while the phone-based system had no such obvious inclinations. The averaged scores over this evaluation set are 0.6 for the voice segments and 0.36 for the blanks, while a phone-based system has much smaller scores for them.

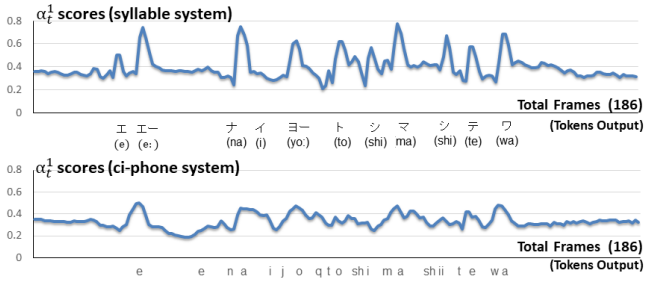

Fig. 4. Attention scores of short-cut paths from a residual block (verified by two systems)

3. EXPERIMENTS

3.1. Data and Task Descriptions

In this paper, we tested our proposed method on the "Corpus of Spontaneous Japanese" (CSJ) [25] with its 240-hour lecture recordings as the training set (**CSJ-Train**), based on the benchmarks [26, 27]. We used three official evaluation sets, (**CSJ-Eval01**, **CSJ-Eval02**, and **CSJ-Eval03**), each of which contained ten lecture recordings [27], 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 from CSJ (**CSJ-Train$_{\text{small}}$**) to train the seed models for warm-start initialization [28] and tuning parameters.

Table 1. CSJ Data Sets

		#Lectures	Hours
Training set	**CSJ-Train$_{\text{small}}$**	155	27.6
	CSJ-Train	957	240
Development set (**CSJ-Dev**)		10	2.0
Testing set	**CSJ-Eval01**	10	2.0
	CSJ-Eval02	10	2.1
	CSJ-Eval03	10	1.4

3.2. Baseline Descriptions

We trained the baseline models using CSJ-Train. To start the first baseline model (**DNN-HMM-CE**), we first trained a GMM-HMM model, followed by a DNN model with five hidden layers, each of which is comprised 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 the following 72-dim filter-bank features (24-dim static $+\Delta$ $+\Delta\Delta$): mean and variance normalized per speaker and 11 spliced frames (five left, current, five right). The DNN model was trained using the standard stochastic gradient descent (SGD) based on the cross-entropy (CE) loss criterion. All were implemented using the Kaldi toolkit (nnet1) [29, 26].

We trained the baseline CTC models with similar parameter sizes of DNN-HMM-CE 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 $+\Delta$ $+\Delta\Delta$). The right context of the feature is one frame, and the left context is also one frame. Since the syllable-based system performs better than the context independent (CI) phone-based system as reported in [30], we use 263 Japanese syllables (Kana, the basic unit of the Japanese language writing system), one non-spoken noise, one spoken noise and one blank (ϕ) as the basic acoustic modeling unit. The BLSTM network has six hidden layers, each of which is composed of 512 nodes. 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 1024 nodes). The right context of the feature is eight frames, and the left context is a zero frame. The subsampling numbers of both systems are set to three. We choose this setting for making the real-time factor (RTF) around 0.3. We did not follow the settings of syllable-based BLSTM-CTC systems in [30], because these previously trained systems are too slow and cannot be used in our Sprintra WFST decoder [31].

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 WFST-based decoding graphs for these models. The CTC-based models were decoded on EESEN decoder [3] and the WFST-based decoding graph normalized using a subword FST [32]. The performances are shown in Section 3.5.

3.3. Settings for Training Proposed VResTD-CTC

We built a set of prototype networks (tuned using a 29 context independent (CI) phone-based seed system trained on CSJ-Train$_{\text{small}}$) and selected a best 26-layer structure using an evaluation set (CSJ-Dev) shown in Table 2.

Table 2. Network structure: all layers are connected with ReLU activations, and past and future memories are stored in two global [1024×1] vectors. The residual-skips use linear projections for dimensional matching.

Component (ordered in seq.)	Structure (26 FC layers)		#Para. (35.9M)
Input	72-dim filter-bank features (24-dim static+Δ+ΔΔ)		
ResBlock 1	3 FC layers [72×2048] [2048×2048] [2048×2048]	Residual-skip [72×2048]	8.3M
ResBlock 2	3 FC layers [2048×128] [128×128] [128×2048]	Residual-skip [2048×2048]	4.5M
ResBlock 3	3 FC layers [2048×128] [128×128] [128×1024]	Residual-skip [2048×1024]	2.6M
TDResBlock 1 (⋆)	5 Time-delay layers [1024×1024]×5	Residual-skip [1024×1024]	6.0M
TDResBlock 2 (⋆)	5 Time-delay layers [1024×1024]×5	Residual-skip [1024×1024]	6.0M
TDResBlock 3 (⋆)	5 Time-delay layers [1024×1024]×5	Residual-skip [1024×1024]	6.0M
Fully-connect	2 FC layers [1024×2048] [2048×output-dim]		2.5M
Output	softmax		

(⋆) means the residual block where we insert vertical-attention module.

The network has two parts as shown in Table 2.

1. Three residual blocks ResBlocks (**ResBlock**): Each ResBlock has three FC layers connected with activations and a residual-skip. No time-delay operation is used. The stacked ResBlocks are located close to the

input layer and transform the speech feature to a higher level representation (Table 2).

2. Three time-delay residual blocks (**TDResBlock**): Each TDResBlock with five FC layers and five time-delay operations (bidirectional). Two global memory blocks (past and future) encode the output signal after every time-delay operation (see Eq.2). The residual connection passes the frame of the current time-step across five layers (Table 2). The windows for stacked time-delay operations are shown in Table 3.

Table 3. Stacked time-delay operations in residual blocks

1st Block	2nd Block	3rd Block
$\{-1, 0, +1\}$	$\{-6, 0, +6\}$	$\{-11, 0, +11\}$
$\{-2, 0, +2\}$	$\{-7, 0, +7\}$	$\{-12, 0, +12\}$
$\{-3, 0, +3\}$	$\{-8, 0, +8\}$	$\{-13, 0, +13\}$
$\{-4, 0, +4\}$	$\{-9, 0, +9\}$	$\{-14, 0, +14\}$
$\{-5, 0, +5\}$	$\{-10, 0, +10\}$	$\{-15, 0, +15\}$

The proposed CTC model (**VResTD-CTC**) is trained with CNTK [33] and 72-dim non-spliced filter-bank features (24-dim static +Δ +$\Delta\Delta$). We trained the seed model with a 27.6-hour CSJ-Train$_{\text{small}}$ with the CTC loss criterion and used the model parameters to initialize the CTC models. The FsAdaGrad algorithm (an implementation of Adam [34]) was used during the CTC training. To speed up the training with the 240-hour training data (CSJ-Train), we used the block-wise model update filtering (BMUF) distributed training algorithm [35] on four Tesla K40m GPUs. The initial learning rate for each frame was 0.00001 and was automatically adjusted using validation on CSJ-Dev. The mini-batch size was 2048. The number of parallel sequences in the same mini-batch was 16, and the maximum epoch number was 25. The CTC model was decoded by feeding the scaled log-likelihood output of the network to the EESEN decoder [3]. We modified the decoding graph normalized using a subword FST [32] (We select 1-gram and 0.9 as the weight factor by grid search). The detailed settings were also described in our previous work [16, 17].

We also trained a cross-entropy model (**VResTD-CE**) with the same structure and feature settings using CNTK. The label is the same with the **DNN-HMM-CE** system.

3.4. Adding Attention Modules to the Network Structure

We inserted the proposed attention module into every time-delay residual block (TDResBlock) of the previously trained **VResTD-CTC**. Then we trained the whole network with CSJ-Train. The initial learning rate is 0.00001. The mini-batch size is set to 2048. We checked the performance on CSJ-Dev after every epoch and stopped the training at the 17-th epoch before the performance started to drop.

We also apply self-attention mechanism [24] to reweight the $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{j}} \approx (\tilde{h}_{t-N_1^l}^j, \tilde{h}_t^j, \tilde{h}_{t+N_2^l}^j)$ of the j-th TD layer (see Eq.2 in last subsection). The $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{j}}$ is reweighted using the attention scores $\boldsymbol{\alpha}_{\boldsymbol{t}}^{\boldsymbol{j}} = (\alpha_{t-N_1^l}^j, \alpha_t^j, \alpha_{t+N_2^l}^j)$.

$$\tilde{\boldsymbol{H}'}_{\boldsymbol{t}}^{\boldsymbol{j}} = (\alpha_{t-N_1^l}^j \cdot \tilde{h}_{t-N_1^j}^j, \alpha_t^j \cdot \tilde{h}_t^j, \alpha_{t+N_2^l}^j \cdot \tilde{h}_{t+N_2^j}^j) \tag{9}$$

where the $\tilde{\boldsymbol{H}'}_{\boldsymbol{t}}^{\boldsymbol{j}}$ is the reweighted result. It will be processed in the same way as Eq.3. The $\boldsymbol{\alpha}_{\boldsymbol{t}}^{\boldsymbol{j}}$ is generated as follows:

$$\boldsymbol{\alpha}_{\boldsymbol{t}}^{\boldsymbol{j}} = Attention(\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{j}}, ..., \tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}}) \tag{10}$$

where $Attention(\cdot)$ is a single layer MLP network with a softmax output. We use the cached feedforward outputs of a set of layers ($\tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{j}}, ..., \tilde{\boldsymbol{H}}_{\boldsymbol{t}}^{\boldsymbol{l}}$) to calculate the attention scores similar to the multi-head attention [24]. We found using output of two successive layers at the begining of the first TDResBlock can save the training time and achieve the best performance.

We call this "**horizontal-attention (hAtt)**" and name our proposed attention "**vertical-attention (vAtt)**". The training setting is same with the vertical-attention.

3.5. Performances of Speech Recognition

We evaluated the performance of the proposed CTC-based models (the **VResTD-CTC** with either kind of attention and without attention) on three CSJ evaluation sets and compared the results with the baseline systems introduced in Section 3.1 (**DNN-HMM-CE**, **BLSTM/ULSTM-CTC** and **VResTD-CE**).

Table 4. ASR performance (WER%) of acoustic models with proposed neural network structures compared with baselines

Network (#Para. approx.)	WER%			
	Eval 01	Eval 02	Eval 03	Ave.
DNN-HMM-CE (38M)	14.4	11.8	15.6	13.9
BLSTM-CTC (33M)	**13.9**	11.6	15.8	13.8
ULSTM-CTC (38M)	14.5	12.1	16.7	14.4
VResTD-CE (51M)	**13.9**	**11.2**	15.2	13.4
VResTD-CTC w/o Att (36M)	18.0	16.2	17.8	17.3
VResTD-CTC w/ hAtt (36M)	**14.2**	12.5	15.1	14.1
VResTD-CTC w/ vAtt (36M)	**13.8**	**11.0**	**14.2**	**13.0**

The results compared to the proposed method (VResTD-CTC w/ vAtt) without statistical significance (from two-tailed t-test at significant level of p-value < 0.05) are shown in bold fonts.

From Table 4, the VResTD-CE outperforms the DNN baseline (DNN-HMM-CE) verified the effectiveness of the proposed very deep structure.

However, the syllable-based VResTD-CTC model cannot work well, compared to the baseline systems in Table 4 and the phone-based VResTD-CTC model in Table 5. We look

into the problem by breaking down the insertion rate from the WER% as shown in Table 5. The WER% increases are mainly associated with the increased insertion errors. Since syllable unit is more dynamic and longer than the phone unit, blank tokens will be wrongly inserted inside the syllable unit. The structure difference caused such performance gaps between TDNN (including our model) and BLSTM 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. This result also suggests the VResTD-CTC model still has room for improvement.

Table 5. Word Error Rate (WER%) and Insertion Rate (Ins%) of VResTD-CTC models with different acoustic units

System	WER% (Ins%)			
	Eval 01	Eval 02	Eval 03	Ave.
phone-based	15.1 (1.4)	12.0 (1.4)	14.5 (1.4)	13.9 (1.4)
syllable-based	18.0 (3.6)	16.2 (3.9)	17.8 (3.5)	17.3 (3.7)

From Table 4, we find both of the attention methods (vAtt and hAtt) can improve the VResTD-CTC model. This result is reasonable, because the attention mechanisms (vAtt and hAtt) can capture the feedbacks from current layer and improves the TDNN layer. Compared with the hAtt, the VResTD-CTC with vAtt can significantly outperform the DNN-HMM-CE baseline. The VResTD-CTC with vAtt can also outperform BLSTM/ULSTM-CTC and VResTD-CE on averaged word error rate (WER%), while the improvement from the hAtt is very limited. The VResTD-CTC with hAtt only works better than the ULSTM-CTC.

Concerning the decoding speed, we tested the real-time factors (RTFs) of the above models. The parameter size of the proposed very deep model is similar with the baseline models (except VResTD-CE, which is much larger). The decoding is applied to the same workstation without sharing with other jobs, and the feedforward pass and decoding are all calculated into RTFs. Compared with the DNN-HMM-CE (RTF=0.70) and VResTD-CE models (RTF=0.74), the speed of proposed VResTD-CTC (w/ vAtt) is 0.22 on RTF, which matches the 3x faster results reported in [3]. It is comparable to the ULSTM-CTC model (RTF=0.28) and faster than the BLSTM-CTC model (RTF=0.4) even with much deeper layers.

4. CONCLUSIONS AND FUTURE WORKS

In this study, we investigate deep architectures and techniques which is suitable for CTC-based acoustic model training. The proposed very deep residual time-delay neural network (VResTD) with attention mechanism can be integrated with CTC training and significantly outperform the DNN-HMM-CE and LSTM-based CTC models (both bidirectional and unidirectional). In the future, we will further improve this deep structure.

5. ACKNOWLEDGEMENTS

The authors thank Naoyuki Kanda who developed the concrete baseline systems (especially the decoding method) and Mitsuyoshi Tachimori for discussing about CTC-based decoding.

6. REFERENCES

[1] A. Graves, S. Fernandez, F. Gomez, and J. Shmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[2] A. Graves and N. Jaitly, "Towards End-to-End speech recognition with recurrent neural networks," in *Proc. ICML*, 2014.

[3] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-End speech recognition using deep RNN models and WFST-based decoding," in *Proc. IEEE-ASRU*, 2015, pp. 167–174.

[4] M. Bi, Y. Qian, and K. Yu, "Very deep convolutional neural networks for LVCSR," in *Proc. INTERSPEECH*, 2015.

[5] Y. Qian and et al., "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Trans. ASLP*, vol. 24, no. 12, pp. 2263–2276, 2016.

[6] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. IEEE-ICASSP*, 2017.

[7] D. Yu, W. Xiong, J. Droppo, A. Stolcke, G. Ye, J. Li, and G. Zweig, "Deep convolutional neural networks with layer-wise context expansion and attention," in *Proc. INTERSPEECH*, 2016.

[8] S. Zhang, M. Li, Z. Yan, and L. Dai, "Deep-FSMN for large vocaburary continuous speech recognition," in *arXiv preprint (accepted for ICASSP2018) arxiv:1803.05030*, 2018.

[9] F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Improved $tdnn$s using deep kernels and frequency dependent grid-rnns," in *arXiv preprint (accepted for ICASSP2018) arxiv:1802.06412*, 2018.

[10] M. Baskar and et al., "Residual memory networks: Feed-forward approach to learn long-term temporal dependencies," in *Proc. IEEE-ICASSP*, 2017.

[11] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE/ACM Trans. ASLP*, vol. 37, no. 3, pp. 328–339, 1989.

[12] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. INTERSPEECH*, 2015.

[13] S. Zhang, C. Liu, H. Jiang, S. Wei, L. Dai, and Y. Hu, "Feed-forward sequential memory networks: A new structure to learn long-term dependency," in *arXiv preprint arxiv:1512.08301*, 2015.

[14] J. Tang, S. Zhang, S. Wei, and L. Dai, "Future context attention for unidirectional LSTM based acoustic model," in *Proc. INTERSPEECH*, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[16] S. Li, X. Lu, R.Takashima, P. Shen, and H. Kawai, "Improving CTC-based acoustic model with very deep residual neural network," in *Meeting of Acoustical Society of Japan*, Spring, 2018.

[17] S. Li, X. Lu, R.Takashima, P. Shen, and H. Kawai, "Improving CTC-based acoustic model with very deep residual neural network," in *Proc. INTERSPEECH*, 2018.

[18] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. NIPS*, 2016, pp. 550–558.

[19] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[20] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE-ICASSP*, 2016.

[21] 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.

[22] C. Raffel and D.P.W. Ellis, "Feed-forward networks with attention can solve some long-term memory problems," in *arXiv preprint arXiv:1512.08756*, 2016.

[23] 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.

[24] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur, "A time-restricted self-attention layer for asr," in *IEEE-ICASSP (accepted)*, 2018.

[25] K. Maekawa, "Corpus of spontaneous japanese: Its design and evaluation," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[26] T. Moriya, T. Shinozaki, and S. Watanabe, "Kaldi recipe for Japanese spontaneous speech recognition and its evaluation," in *Autumn Meeting of ASJ*, 2015, number 3-Q-7.

[27] T. Kawahara, H. Nanjo, T. Shinozaki, and S. Furui, "Benchmark test for speech recognition using the corpus of spontaneous japanese," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[28] J. Dean and et al., "Large scale distributed deep networks," in *Proc. NIPS*, 2012.

[29] D. Povey and et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE-ASRU*, 2011.

[30] N. Kanda, X. Lu, and H. Kawai, "Maximum-a-Posteriori-based decoding for End-to-End acoustic models," *IEEE/ACM Trans. ASLP*, vol. 25, no. 5, pp. 1023–1034, 2017.

[31] P.R. Dixon, C. Hori, and H. Kashioka, "Development of the SprinTra WFST speech decoder," *NICT Research Journal*, pp. 15–20, 2012.

[32] N. Kanda, X. Lu, and H. Kawai, "Maximum a posteriori based decoding for CTC acoustic models," in *Proc. INTERSPEECH*, 2016, pp. 1868–1872.

[33] A. Agarwal and et al., "An introduction to computational networks and the computational network toolkit," in *Microsoft Technical Report MSR-TR-2014-112*, 2014.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[35] K. Chen and Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," in *Proc. IEEE-ICASSP*, 2016.