INVESTIGATION OF SEQUENCE-LEVEL KNOWLEDGE DISTILLATION METHODS FOR CTC ACOUSTIC MODELS

Ryoichi Takashima, Li Sheng, and Hisashi Kawai

National Institute of Information and Communications Technology (NICT), Japan

ABSTRACT

This paper presents knowledge distillation (KD) methods for training connectionist temporal classification (CTC) acoustic models. In a previous study, we proposed a KD method based on the sequence-level cross-entropy, and showed that the conventional KD method based on the frame-level cross-entropy did not work effectively for CTC acoustic models, whereas the proposed method improved the performance of the models. In this paper, we investigate the implementation of sequence-level KD for CTC models and propose a lattice-based sequence-level KD method. Experiments investigating model compression and the training of a noise-robust model using the Wall Street Journal (WSJ) and CHiME4 datasets demonstrate that the sequence-level KD methods improve the performance of CTC acoustic models on both two tasks, and show that the lattice-based method can compute the sequence-level KD more efficiently than the N-best-based method proposed in our previous work.

***Index Terms*—** Speech recognition, acoustic model, connectionist temporal classification, knowledge distillation

1. INTRODUCTION

Acoustic models based on connectionist temporal classification (CTC) [1, 2] have been widely studied [3, 4, 5, 6, 7, 8, 9, 10] as alternatives to the conventional deep neural network (DNN)-hidden Markov model (HMM) hybrids for automatic speech recognition (ASR) systems. As CTC does not assume the presence of frame-level labels (i.e., alignment information) and trains a recurrent neural network (RNN) that directly estimates the label sequence from the input sequence, it does not require either the alignment information for training or the conversion of frame-level output into a label sequence for decoding. In addition, DNN-HMM uses temporally short labels (e.g., state of a triphone HMM), whereas CTC can deal with temporally long labels (e.g., monophone, character, and word). Based on the above features, CTC has been used for tasks such as end-to-end ASR, which does not use any dictionaries or language models [9, 10], and acoustic event detection [11, 12].

Knowledge distillation (KD) [13, 14], also known as teacher-student training, is a method for training neural networks. KD trains a model (called the *student model*) using the outputs of another model (called the *teacher model*) as training labels, with the aim of training the student model to have similar performance to the teacher's. KD is often used for model compression, where a larger and higher-performance model and a smaller and lower-performance model are used as the teacher and student models, respectively [15, 16, 17, 18]. There have also been studies on the training of noise-robust acoustic models based on KD [19, 20, 21]. These methods use parallel training data, which consist of clean speech and noisy speech data, and train a student model using the noisy speech and a teacher model using the clean speech (details are described in Sec. 4). It has been reported that the student model trained using KD exhibit better performance than the model trained in the conventional multi-condition framework.

KD is known to be effective for ASR tasks using DNN-HMM hybrid models. However, [22] reported that KD degrades the performance of CTC acoustic models. In our previous study [23], therefore, we investigated KD methods for CTC acoustic models, and found that the conventional KD method based on frame-level cross-entropy (CE) worsened the performance of student CTC models, whereas the proposed sequence-level KD method based on sequence-level CE improved the performance.

In our previous work, we proposed an N-best-based approach to implement the sequence-level KD method. As that approach incurs N times the computational load for training (where N is the number of hypotheses to be used) compared with conventional CTC training, we only evaluated the proposed method on one model compression task under one condition (N=10). The advances reported in this paper over our previous work are as follows:

- We evaluate the N-best-based approach with more hypotheses (N=50) by implementing it such that N computations are performed in parallel on a general-purpose GPU.
- We propose a lattice-based approach for implementing the sequence-level KD more efficiently.
- We evaluate our methods on not only a model compression task but also a noise-robust model training task.

In the remainder of this paper, we explain the conventional methods and our proposed sequence-level KD methods in Sec. 2 and Sec. 3, respectively, and present the experimental results and conclude our work in Sec. 4 and Sec. 5, respectively.

2. RELATED WORK

2.1. Connectionist temporal classification

In the CTC framework [1], a sequence of labels estimated for each frame (called a 'path' and denoted as π) is converted into a label sequence (denoted as $\mathbf{l}$) by deleting repeated labels and blank labels (i.e., "no label"). We call this conversion "CTC mapping" under the function $\mathcal{B}$, where $\mathbf{l} = \mathcal{B}(\pi)$. The conditional probability of the label sequence $\mathbf{l}$ given the input sequence $\mathbf{x}$ is defined as the sum of the probabilities of all possible corresponding paths:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \quad (1)$$

The CTC model is trained by maximizing the likelihood, that is, minimizing the loss function $\mathcal{L}_{\mathrm{CTC}}$, which is defined as

$$\mathcal{L}_{\mathrm{CTC}} = -\sum_{(\mathbf{x},\mathbf{l}) \in Z} \ln p(\mathbf{l}|\mathbf{x}) = \sum_{(\mathbf{x},\mathbf{l}) \in Z} \mathcal{F}_{\mathrm{CTC}}(\mathbf{l}|\mathbf{x}), \quad (2)$$

The work was performed while Ryoichi Takashima was at NICT. He is currently with Hitachi Ltd. (E-mail: ryoichi.takashima.dh@hitachi.com)

where Z denotes the training dataset and $\mathcal{F}_{\mathrm{CTC}}(\mathbf{l}|\mathbf{x}) = -\ln p(\mathbf{l}|\mathbf{x})$ is the local loss, defined for explanation in Sec. 3. The derivative of $\mathcal{L}_{\mathrm{CTC}}$ with respect to frame-level output y_k^t for a label k at frame t, which is required for backpropagation, is computed as follows:

$$\frac{\partial \mathcal{L}_{\mathrm{CTC}}}{\partial y_k^t} = -\frac{1}{p(\mathbf{l}|\mathbf{x})} \frac{1}{y_k^t} \sum_{\pi:(\mathcal{B}(\pi)=\mathbf{l}, \pi_t=k)} p(\pi|\mathbf{x}), \tag{3}$$

where $\sum_{\pi:(\mathcal{B}(\pi)=\mathbf{l}, \pi_t=k)} p(\pi|\mathbf{x})$ expresses the sum of the probabilities of all paths satisfying $\mathcal{B}(\pi) = \mathbf{l}$ and $\pi_t = k$. This sum of probabilities is efficiently computed using the forward-backward algorithm:

$$\sum_{\pi:(\mathcal{B}(\pi)=\mathbf{l}, \pi_t=k)} p(\pi|\mathbf{x}) = \sum_{s:(l_s=k)} \frac{\alpha_t(s)\beta_t(s)}{y_s^t}, \tag{4}$$

where $\alpha_t(s)$ and $\beta_t(s)$ denote the forward and backward probabilities, respectively, of the path passing through the s-th label in $\mathbf{l}$ at the t-th frame. $s : (l_s = k)$ denotes the position of label k in $\mathbf{l}$. For details of the forward-backward algorithm, see [1].

2.2. Knowledge distillation

KD [14] is a method for training neural networks. The main idea of KD is to train a student model with the outputs of a teacher model used as training labels (often called soft labels) instead of the correct binary labels. In the KD framework, a teacher model is first trained using the correct labels. Using the outputs (i.e., probability distribution) from the teacher model corresponding to the training data, the student model is then trained under the CE criteria as follows:

$$\mathcal{L}_{\mathrm{KD}} = -\sum_l p_{\mathrm{tea}}(l|x) \ln p_{\mathrm{stu}}(l|x), \tag{5}$$

where $p_{\mathrm{tea}}(l|x)$ denotes the posterior probability of label l given input x estimated by the teacher model (i.e., the output of the teacher model) and $p_{\mathrm{stu}}(l|x)$ is that estimated by the student model.

In previous studies [15, 16, 17, 18], Eq. (5) is calculated for each frame, and the KD is applied to DNN-HMM hybrid models. We refer to this original KD version as '*frame-level KD*'. There are also studies in which the KD technique is used for sequence training, such as attention model [24, 25] and maximum mutual information (MMI)-based training [26, 27]. In these methods, the student model is trained by minimizing the CE between the probability distributions of the label sequences on a teacher model and on a student model. We refer to this KD approach as '*sequence-level KD*'.

3. SEQUENCE-LEVEL KD FOR CTC ACOUSTIC MODELS

3.1. Overview

Figure 1 shows an overview of sequence-level KD for CTC acoustic models. Following some relevant studies [24, 26, 27], we extract the hypotheses of the label sequence and their posterior probabilities for each training sample as estimated by an already-trained teacher CTC model. Using the hypotheses and posterior probabilities, we then train a student CTC model under sequence-level CE criteria:

$$\mathcal{L}_{\mathrm{CTC-KD_{seq}}} = -\sum_{\mathbf{x}\in Z} \sum_{\mathbf{h}\in\mathcal{H}} p_{\mathrm{tea}}(\mathbf{h}|\mathbf{x}) \ln p_{\mathrm{stu}}(\mathbf{h}|\mathbf{x}). \tag{6}$$

Here, $\mathbf{h}$ denotes a hypothesis of the label sequence in the set of all possible hypotheses $\mathcal{H}$. $p_{\mathrm{tea}}(\mathbf{h}|\mathbf{x})$ and $p_{\mathrm{stu}}(\mathbf{h}|\mathbf{x})$ are the posterior probabilities of hypothesis $\mathbf{h}$ estimated by the teacher CTC model and the student CTC model, respectively. As Eq. (6) can be expressed as

$$\mathcal{L}_{\mathrm{CTC-KD_{seq}}} = \sum_{\mathbf{x}\in Z} \sum_{\mathbf{h}\in\mathcal{H}} p_{\mathrm{tea}}(\mathbf{h}|\mathbf{x}) \mathcal{F}_{\mathrm{CTC}}(\mathbf{h}|\mathbf{x}), \tag{7}$$

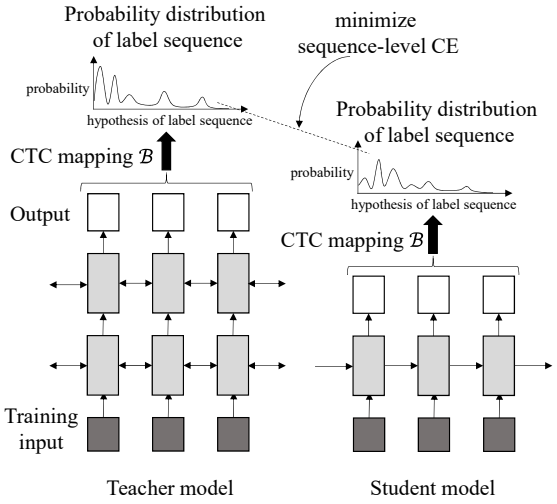

Fig. 1. Overview of sequence-level KD for CTC acoustic models

the sequence-level KD criteria for the CTC model can be summarized as the weighted mean of the original CTC loss regarding each hypothesis of the label sequence. Because it is unrealistic to extract $p_{\mathrm{tea}}(\mathbf{h}|\mathbf{x})$ for all possible hypotheses, we must then approximate the hypothesis space $\mathcal{H}$ as a limited hypothesis space. In this study, we use N-best-based approximation and lattice-based approximation.

3.2. N-best-based sequence-level KD

N-best-based sequence-level KD was first proposed in [24], and was applied to the CTC training framework in our previous study [23]. In N-best-based KD, we approximate Eq. (7) using the N-best hypotheses, that is, the N hypotheses having the highest posterior probabilities:

$$\tilde{\mathcal{L}}_{\mathrm{CTC-KD_{Nbest}}} = \sum_{\mathbf{x}\in Z} \sum_{n=1}^{N} \tilde{p}_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x}) \mathcal{F}_{\mathrm{CTC}}(\mathbf{h}_n|\mathbf{x}). \tag{8}$$

Here, $\mathbf{h}_n$ denotes the n-th hypothesis in the N-best hypotheses, and $\tilde{p}_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x})$ denotes its posterior probability, normalized so that the sum equals 1:

$$\tilde{p}_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x}) = \frac{p_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x})}{\sum_{n=1}^{N} p_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x})}. \tag{9}$$

The derivative with respect to y_k^t, which is required for backpropagation, is computed as follows:

$$\frac{\partial \tilde{\mathcal{L}}_{\mathrm{CTC-KD_{Nbest}}}}{\partial y_k^t} = -\sum_{n=1}^{N} \tilde{p}_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x}) \frac{1}{p_{\mathrm{stu}}(\mathbf{h}_n|\mathbf{x})} \frac{1}{y_k^t} \sum_{\substack{\pi:(\mathcal{B}(\pi)=\mathbf{h}_n, \\ \pi_t=k)}} p(\pi|\mathbf{x}). \tag{10}$$

From Eqs. (2), (3), (8), and (10), we can implement the N-best-based KD in the following way: (i) compute $\mathcal{F}_{\mathrm{CTC}}(\mathbf{h}_n|\mathbf{x})$ or $\sum_{\pi:(\mathcal{B}(\pi)=\mathbf{h}_n, \pi_t=k)} p(\pi|\mathbf{x})$ for each of the N-best hypotheses using the forward-backward algorithm from conventional CTC training, (ii) compute their weighted mean according to the probabilities $\tilde{p}_{\mathrm{tea}}(\mathbf{h}_n|\mathbf{x})$. Although it is relatively easy to implement the N-best-based KD using any tools with a CTC-training function (e.g., Tensorflow [28] or CNTK [29]), this approach incurs N times the computational cost of conventional CTC training. As we can compute $\mathcal{F}_{\mathrm{CTC}}$ or its derivative for each hypothesis independently, we can implement N computations in parallel on a general-purpose GPU. However, using too many hypotheses may overwhelm the GPU cores or memory.

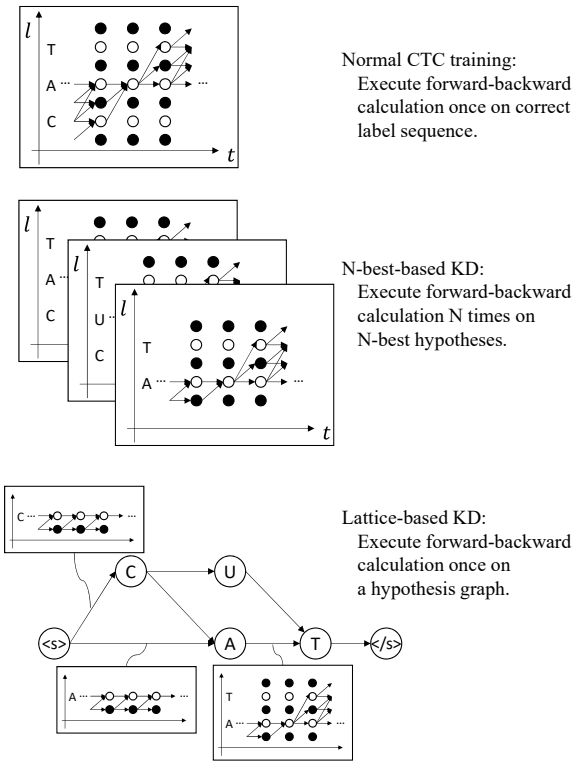

Fig. 2. Forward-backward computations of normal CTC training, N-best-based KD, and lattice-based KD. Black and white circles denote blank and non-blank labels, respectively.

3.3. Lattice-based sequence-level KD

Because the N-best hypotheses tend to have similar label sequences, performing the forward-backward computation independently for each hypothesis often involves redundant computations. In the lattice-based KD framework, for more efficient computations, we form a graph of the hypotheses (i.e., a lattice), and perform the forward-backward computation on the lattice. Figure 2 illustrates the forward-backward computations of normal CTC training, N-best-based KD, and lattice-based KD. In the figure, the three hypotheses "CAT," "CUT," and "AT" are estimated from a sample uttered "CAT." Whereas N-best-based KD performs the forward-backward computation for each hypothesis, lattice-based KD performs one computation for common paths (e.g., $\langle s \rangle \rightarrow$ 'C' and 'T' $\rightarrow \langle /s \rangle$, where $\langle s \rangle$ and $\langle /s \rangle$ denote the start state and end state, respectively).

The forward-backward algorithm on a lattice has been used in [27]. In our proposed approach, we expand the lattice-space forward-backward algorithm to consider the insertion of blank labels between non-blank labels for application to CTC models. First, for computational efficiency, we set the index to each state such that the lattice has no transition from a state having a larger index to that having a smaller index. For the example in Figure 2, we set the index as $[0, 1, 2, 3, 4, 5] = \mathrm{index}([\langle s \rangle, \mathrm{C}, \mathrm{U}, \mathrm{A}, \mathrm{T}, \langle /s \rangle])$, where $\mathrm{index}(*)$ denotes the state indexes. Similar to the forward-backward algorithm in the conventional CTC training, we then expand the lattice by inserting blanks between states such as $[0, 1, \ldots, 9, 10] = \mathrm{index}([\langle s \rangle, -, \mathrm{C}, -, \mathrm{U}, -, \mathrm{A}, -, \mathrm{T}, -, \langle /s \rangle])$, where '-' denotes a blank label. When we denote the state index of the original lattice by $n = 0, \ldots, N$, in the expanded lattice, indexes $2n$ and $2n + 1$ are assigned to non-blank labels and blank labels, respectively.

Computation of forward probability

We now describe the computation of the forward probability $\alpha_t(s)$. Initially, we define the forward probability at $t = 0$ as follows:

$$\begin{aligned} &\alpha_t(0) = 0, && \alpha_t(1) = y_t^{blk} \\ &\alpha_t(2n) = y_t^{lab(n)} p_{tea}(n|0), && \alpha_t(2n+1) = 0, \end{aligned} \tag{11}$$

where y_t^{blk} denotes the network output corresponding to the blank, and $y_t^{lab(n)}$ denotes that corresponding to the label of state n. State 0 is the start state $\langle s \rangle$, and its forward probability is defined as $\alpha_t(0) = 0$ because it does not have a label. $p_{tea}(n|m)$ denotes the transition probability from state m to state n, that is, the arc weight in the lattice estimated by the teacher model. In the cases of $m = 0$ (i.e., transition from start state) and $n = N$ (i.e., transition to end state $\langle /s \rangle$), $p_{tea}(n|m)$ takes binary values.

Next, we define $\alpha_t(s)$ for $t > 0$ as follows:

$$\alpha_t(0) = 0, \quad \alpha_t(1) = y_t^{blk} \alpha_{t-1}(1)$$

$$\begin{aligned} \alpha_t(2n) &= y_t^{lab(n)} \Big(\alpha_{t-1}(2n) \\ &\quad + \sum_{\substack{m \in 1:n-1 \\ lab(n) \neq lab(m)}} p_{tea}(n|m)(\alpha_{t-1}(2m) + \alpha_{t-1}(2m+1)) \\ &\quad + \sum_{\substack{m \in 1:n-1 \\ lab(n) = lab(m)}} p_{tea}(n|m) \alpha_{t-1}(2m+1) \Big) \\ \alpha_t(2n+1) &= y_t^{blk} (\alpha_{t-1}(2n) + \alpha_{t-1}(2n+1)). \end{aligned} \tag{12}$$

The differences from normal CTC training are as follows: (i) there are multiple states transitioning to state $2n$ (corresponding to arcs of lattice), and so we must sum the previous forward probabilities over $m \in 1 : n-1$, (ii) the forward probabilities corresponding to the transition to $2n$ from $2m$ and $2m+1$ are weighted by transition probability $p_{tea}(n|m)$, and this means that we perform the calculation of the weighted mean in the sequence-level KD framework partially for each state transition.

Computation of backward probability

We now describe the computation of the backward probability $\beta_t(s)$. Initially, we define the backward probability for the last frame $t = T - 1$ as follows:

$$\begin{aligned} \beta_t(2n) &= y_t^{lab(n)} p_{tea}(N|n) \\ \beta_t(2n+1) &= y_t^{blk} p_{tea}(N|n), \end{aligned} \tag{13}$$

where $p_{tea}(N|n)$ denotes the transition probability to the end state and has a binary value ($p_{tea}(N|N) = 0$).

For $t < T - 1$, we define $\beta_t(s)$ as follows:

$$\begin{aligned} \beta_t(2n) &= y_t^{lab(n)} \Big(\beta_{t+1}(2n) + \beta_{t+1}(2n+1) \\ &\quad + \sum_{\substack{m \in n+1:N-1 \\ lab(n) \neq lab(m)}} p_{tea}(m|n) \beta_{t+1}(2m) \Big) \\ \beta_t(2n+1) &= y_t^{blk} \Big(\beta_{t+1}(2n+1) \\ &\quad + \sum_{m \in n+1:N-1} p_{tea}(m|n) \beta_{t+1}(2m) \Big). \end{aligned} \tag{14}$$

Table 1. Results of the model compression task using WSJ dataset

Acoustic Model	KD method	WER w/ LM		PER w/o LM		fps. in training
		dev93	eval92	dev93	eval92	
teacher CTC	none	17.03	10.79	21.19	15.38	1306.3
student CTC	none	20.31	13.59	29.76	24.16	2318.7
	1-best	21.86	14.85	29.87	24.01	1982.0
	10-best	20.46	13.47	28.22	22.46	2324.6
	50-best	**19.99**	**12.94**	28.17	22.13	1288.9
	lattice	20.59	13.52	**28.04**	**21.94**	1879.9
	frame	26.60	17.54	37.94	33.06	4350.1

4. EXPERIMENTS

4.1. Experiments on a model compression task

In the experiment on a model compression task, a smaller student model was trained based on KD with a larger teacher model. We used a bidirectional long short-term memory (LSTM) [30] having five hidden layers and 320 memory cells in each layer as the teacher CTC model, and a unidirectional LSTM having three hidden layers and 160 memory cells as the student CTC model. The experiment was conducted using the Wall Street Journal (WSJ) dataset [31], with "WSJ0" (known as "train_si84" in the Kaldi recipe [32]) taken as the training set and "dev93" and "eval92" used for evaluation. We extracted 40-dimensional mel-filterbank features with their first- and second-order derivatives as acoustic features, and defined the target labels to include 69 phonemes, two noise marks, and a blank. We set the initial and final learning rates to 0.0004 and 0.000004, respectively, and decreased the learning rate exponentially on each of 15 epochs. The training was executed on an NVIDIA Tesla P100 PCIe 16 GB GPU. To train and evaluate the CTC models, we used the EESEN toolkit [5]; this was also used to implement both the frame-level and sequence-level KD methods. In the sequence-level KD methods, we extracted the N-best hypotheses and lattices using the WFST beam-search [33]. In this process, we used only the token WFST (defined as T.fst in EESEN), which maps a sequence of frame-level CTC labels to a label sequence (i.e., $\mathcal{B}$ in Sec. 2.1). Note that we did not use a lexicon or language model in this process.

Table 1 presents the experimental results. When evaluating the word error rate (WER [%]), we used a lexicon and a language model. We used the CMU dictionary as the lexicon and the 20,000-word-vocabulary WSJ pruned language model, known as "lm_tgpr" in the Kaldi recipe. When evaluating phone error rate (PER [%]), we did not use either a lexicon or language model. The results in this table indicate that the conventional frame-level KD worsened the performance of the student CTC model compared with the CTC model trained without KD. N-best-based sequence-level KD improved the performance when N was greater than 10, with more hypotheses generating better performance. Lattice-based KD achieved the best performance in evaluating PER without a lexicon or language model; however, it performed almost the same as the CTC model without KD in evaluating WER with a lexicon and language model.

Frames per second (fps) denotes the number of frames processed in one second in training, with a higher fps representing faster training. With $N = 50$, N-best-based KD increased the training time by a factor of two compared with the normal CTC training without KD; this was caused by a lack of GPU cores. However, the lattice-based KD broadly suppressed this increase in training time. This result means that lattice-based KD can compute the sequence-level KD more efficiently than N-best-based KD.

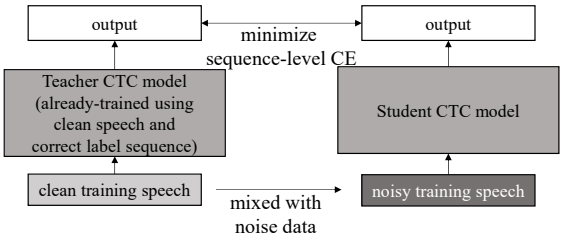

Fig. 3. Training of noise-robust acoustic model using KD

Table 2. Results of training the noise-robust model using CHiME4 dataset

Acoustic Model	KD method	WER w/ LM		PER w/o LM	
		dt05_simu	et05_real	dt05_simu	et05_real
student CTC	none	24.99	54.93	37.65	59.39
	50-best	**22.13**	**52.45**	**33.63**	**55.82**
	lattice	22.99	54.05	33.98	56.27

4.2. Experiments on training noise-robust acoustic model

Figure 3 shows an overview of the experiment conducted to train a noise-robust acoustic model. Following [21], we used parallel training data consisting of clean speech and noise-added data. First, we trained the teacher model using only the clean speech and the correct label sequences. We then input the clean speech into the teacher model and extracted the hypotheses and their probabilities. To train the student model with KD, instead of clean speech, we used the noise-added data and hypotheses extracted from teacher model.

We used a bidirectional LSTM having five hidden layers and 160 memory cells in each layer for both the teacher CTC and student CTC. We used the CHiME4 dataset [34] in this experiment. The training dataset "tr05_simu_noisy" in CHiME4 contains noisy data simulated by adding noise (recorded on the bus, in a cafe, in pedestrian areas, and at street junctions) to "WSJ0." Therefore, we constructed parallel training data by using "WSJ0" as clean data and "tr05_simu_noisy" as noisy data. For the evaluation, we used the noisy datasets "dt05_simu_noisy" and "et05_real_isolated_1ch_track" from CHiME4. The other experimental conditions were the same as described in Sec. 4.1.

Table 2 presents the results. "KD method is none" denotes conventional multi-condition training, that is, we trained the CTC model using "tr05_simu_noisy" and the correct label sequences. As shown in this table, both the N-best-based KD and lattice-based KD methods achieved lower PER and WER than the conventional multi-condition training.

5. CONCLUSION

We have confirmed the effectiveness of both the N-best-based and lattice-based KD methods on the experiments we conducted. Furthermore, the lattice-based KD can compute the sequence-level KD more efficiently than the N-best-based KD. However, there were some cases for which sequence-based KD made no significant improvement in terms of WER with language models, even though it significantly improved the PER without language models. Therefore, we will investigate how to integrate a language model into the CTC acoustic model trained using KD. In addition, we will further study the application of related techniques (e.g., temperature) to the sequence-level KD.

6. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML2006*. ACM, 2006, pp. 369–376.

[2] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks.," in *ICML2014*, 2014, vol. 14, pp. 1764–1772.

[3] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Interspeech*. ISCA, 2015, pp. 1468–1472.

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML2016*, 2016, pp. 173–182.

[5] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *ASRU2015*. IEEE, 2015, pp. 167–174.

[6] Naoyuki Kanda, Xugang Lu, and Hisashi Kawai, "Maximum-a-posteriori-based decoding for end-to-end acoustic models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 5, pp. 1023–1034, 2017.

[7] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *ICASSP2017*. IEEE, 2017, pp. 4835–4839.

[8] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," in *Interspeech*. ISCA, 2017, pp. 949–953.

[9] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *ICASSP2018*. IEEE, 2018, pp. 4759–4763.

[10] Amit Das, Jinyu Li, Rui Zhao, and Yifan Gong, "Advancing connectionist temporal classification with attention modeling," in *ICASSP2018*. IEEE, 2018, pp. 4769–4773.

[11] Yun Wang and Florian Metze, "A first attempt at polyphonic sound event detection using connectionist temporal classification," in *ICASSP2017*, 2017, pp. 2986–2990.

[12] Hiroshi Fujimura, Manabu Nagao, and Takashi Masuko, "Simultaneous speech recognition and acoustic event detection using an lstm-ctc acoustic model and a wfst decoder," in *ICASSP2018*. IEEE, 2018, pp. 5834–5838.

[13] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong, "Learning small-size dnn with output-distribution-based criteria," in *Interspeech*, 2014, pp. 1911–1914.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2014.

[15] Yevgen Chebotar and Austin Waters, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Interspeech*, 2016, pp. 3439–3443.

[16] Liang Lu, Michelle Guo, and Steve Renals, "Knowledge distillation for small-footprint highway networks," in *ICASSP2017*, 2017, pp. 4820–4824.

[17] Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R. Hershey, "Student-teacher network learning with enhanced features," in *ICASSP2017*, 2017, pp. 5275–5279.

[18] Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers," in *Interspeech*, 2017, pp. 3697–3701.

[19] Jinyu Li, Rui Zhao, Zhuo Chen, Changliang Liu, Xiong Xiao, Guoli Ye, and Yifan Gong, "Developing far-field speaker system via teacher-student learning," in *ICASSP2018*. IEEE, 2018, pp. 5699–5703.

[20] Tian Tan, Yanmin Qian, and Dong Yu, "Knowledge transfer in permutation invariant training for single-channel multi-talker speech recognition," in *ICASSP2018*. IEEE, 2018, pp. 5714–5718.

[21] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee, "Bridgenets: student-teacher transfer learning based on recursive neural networks and its application to distant speech recognition," in *ICASSP2018*. IEEE, 2018, pp. 5719–5723.

[22] Andrew Senior, Haşim Sak, Félix de Chaumont Quitry, Tara N. Sainath, and Kanishka Rao, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *ASRU2015*. IEEE, 2015, pp. 604–609.

[23] Ryoichi Takashima, Sheng Li, and Hisashi Kawai, "An investigation of a knowledge distillation method for ctc acoustic models," in *ICASSP2018*. IEEE, 2018, pp. 5809–5813.

[24] Yoon Kim and Alexander M. Rush, "Sequence-level knowledge distillation," in *EMNLP2016*, 2016.

[25] Ruoming Pang, Tara N. Sainath, Rohit Prabhavalkar, Suyog Gupta, Yonghui Wu, Shuyuan Zhang, and Chung-Cheng Chiu, "Compression of end-to-end models," in *Interspeech 2018*. ISCA, 2018, pp. 27–31.

[26] Jeremy H. M. Wong and Mark J. F. Gales, "Sequence student-teacher training of deep neural networks," in *Interspeech*, 2016.

[27] Naoyuki Kanda, Yusuke Fujita, and Kenji Nagamatsu, "Sequence distillation for purely sequence trained acoustic models," in *ICASSP2018*. IEEE, 2018, pp. 5964–5968.

[28] "Tensorflow," https://www.tensorflow.org/.

[29] "CNTK," https://github.com/Microsoft/CNTK.

[30] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] John Garofolo et al., "CSR-I (WSJ0) Complete LDC93S6A," DVD. Philadelphia: Linguistic Data Consortium, 1993.

[32] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, et al., "The kaldi speech recognition toolkit," in *ASRU2011*. IEEE Signal Processing Society, 2011.

[33] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[34] Emmanuel Vincent, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech & Language*, 2016.