Self-Distillation Based on High-level Information Supervision for Compressing End-to-End ASR Model

Qiang Xu[1], *Tongtong Song*[1], *Longbiao Wang*[1*], *Hao Shi*[2*], *Yuqin Lin*[1], *Yongjie Lv*[1], *Meng Ge*[1], *Qiang Yu*[1], *Jianwu Dang*[1,3]

[1]Tianjin Key Laboratory of Cognitive Computing and Application,
College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Graduate School of Informatics, Kyoto University, Sakyo-ku, Kyoto, Japan
[3]Japan Advanced Institute of Science and Technology, Ishikawa, Japan

{xu_qiang, longbiao_wang}@tju.edu.cn, shi@sap.ist.i.kyoto-u.ac.jp

Abstract

Model compression of ASR aims to reduce the model parameters while bringing as little performance degradation as possible. Knowledge Distillation (KD) is an efficient model compression method that transfers the knowledge from a large teacher model to a smaller student model. However, most of the existing KD methods study how to fully utilize the teacher's knowledge without paying attention to the student's own knowledge. In this paper, we explore whether the high-level information of the model itself is helpful for low-level information. We first propose neighboring feature self-distillation (NFSD) approach to distill the knowledge from the adjacent deeper layer to the shallow one, which shows significant performance improvement. Therefore, we further propose attention-based feature self-distillation (AFSD) approach to exploit more high-level information. Specifically, AFSD fuses the knowledge from multiple deep layers with an attention mechanism and distills it to a shallow one. The experimental results on AISHELL-1 dataset show that 7.3% and 8.3% relative character error rate (CER) reduction can be achieved from NFSD and AFSD, respectively. In addition, our proposed two approaches can be easily combined with the general teacher-student knowledge distillation method to achieve 12.4% and 13.4% relative CER reduction compared with the baseline student model, respectively.

Index Terms: automatic speech recognition, self-distillation, teacher-student model, model compression

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) [1, 2, 3, 4, 5, 6] has gained more and more attention due to its convenience in human-computer interaction. The mainstream models mainly include connectionist temporal classification (CTC) [2, 7, 8], recurrent neural network-transducer (RNN-T) [3, 9, 10], and attention based encoder-decoder (AED) [11, 12, 13]. However, E2E ASR model often contains a large number of parameters, which requires large storage and computing capacities. So, it is often difficult to be deployed on embedded devices and on-device applications. Although reducing parameter numbers can alleviate these problems, this often leads to a sharp drop in performance. Therefore, it is important to compress the parameters of a large model while minimizing performance degradation.

Knowledge Distillation (KD) [14, 15, 16, 17] is one of the most common model compression methods. KD can be generally divided into three types: offline distillation [14, 18, 19, 20], online distillation [21, 22, 23] and self-distillation [24]. Offline distillation guides the training of a small-parameter model (student model) with a pre-trained large-parameter model (teacher model). Online distillation trains the teacher model and the student model simultaneously, and information from the teacher model is introduced into the student model during training. For self-distillation, the student model becomes its own teacher. Pseudo-targets are introduced into the training process to improve the performance of the model [24]. Compared with offline distillation and online distillation, self-distillation does not require extra teacher models. Self-distillation focuses on how to make better use of its own information rather than extract information from other large models. However, the current self-distillation method [24] does not entirely use the student's own knowledge to guide its training, but creates pseudo-targets to assist the training of the student model.

In this paper, we propose two novel self-distillation methods to utilize high-level information from the model itself without creating pseudo-targets. Our proposed self-distillation methods aim to supervise itself with high-level information. More specifically, we self-distill knowledge from deep encoder (decoder) layers to shallow layers. This is because deep layers can extract features with stronger representational capabilities than shallow layers[14, 22]. The proposed two self-distillation methods are called neighboring feature self-distillation (NFSD) and attention-based feature self-distillation (AFSD).

(1) In NFSD, every two adjacent layers are viewed as one group, and the adjacent groups do not contain the same layers. The features of the deeper layer are distilled to the shallow one in each group. The NFSD method is our initial attempt to use high-level information for self-distillation, proving the effectiveness of high-level information.

(2) We further propose the AFSD method to utilize more high-level information for self-distillation. In AFSD, for each layer, we first use an attention mechanism to calculate the similarity between it and all layers after it; then, we use the obtained similarity weights to fuse all deep features; finally, we distill the fused feature to it.

Furthermore, our methods can be easily integrated with existing teacher-student distillation methods, called NFSD-offline and AFSD-offline. The two ways first use the offline distillation method to obtain the student model, and then the NFSD and AFSD methods are applied to the student model, respectively.

This paper is organized as follows. Section 2 describes the related works. Section 3 describes our proposed methods. Section 4 describes the experimental setting and evaluations. The conclusions and the future works are given in Section 5.

*Corresponding author

2. Related work

The proposed methods are used on both the encoder and decoder sides, and we use U2 [12] for experiments. U2 is a popular hybrid CTC/attention network that supports streaming autoregressive and non-autoregressive decoding. In addition, to compare the performance with offline distillation and further combine our methods with it, we choose the commonly used teacher-student knowledge distillation method: patient knowledge distillation (PKD) [14]. In this section, we briefly review U2 and PKD.

2.1. U2

U2 [12] is a two-pass architecture that contains a Shared Encoder, a CTC Decoder, and an Attention Decoder. The Shared Encoder contains multiple Transformer [25] or Conformer [26] encoder layers, acting as an acoustic feature extractor. The CTC Decoder consists of a projection layer and a log softmax layer, supporting streaming decoding in the first pass. The Attention Decoder contains multiple transformer decoder layers, supporting autoregressive decoding and non-autoregressive rescoring in the second pass.

During training, U2 adopts the training method of joint CTC-Attention [27]. The training loss is combined with CTC loss and AED loss as follows:

$$\mathcal{L}_{\mathrm{U2}}(\mathbf{x},\mathbf{y}) = \lambda\mathcal{L}_{\mathrm{CTC}}(\mathbf{x},\mathbf{y}) + (1-\lambda)\mathcal{L}_{\mathrm{AED}}(\mathbf{x},\mathbf{y}) \quad (1)$$

where $\mathbf{x}$ stands for the acoustic features and $\mathbf{y}$ is the corresponding annotation. $\mathcal{L}_{\mathrm{CTC}}(\mathbf{x},\mathbf{y})$ and $\mathcal{L}_{\mathrm{AED}}(\mathbf{x},\mathbf{y})$ are the CTC and AED losses, respectively. λ is a hyperparameter that balances the importance of CTC and AED loss.

During decoding, U2 provides autoregressive and non-autoregressive decoding modes. In autoregressive decoding mode, the outputs of the CTC Decoder are ignored, the Attention Decoder combines the output of the Shared Encoder to generate outputs in an auto-regressive way. We call this decoding mode as **attention**. In non-autoregressive decoding mode, the CTC Decoder generates n-best hypotheses with the output of the Shared Encoder firstly, then n-best hypotheses are scored by the Attention Decoder with the output of the Shared Encoder to get the best hypothesis as the final decoding result. This decoding mode avoids the auto-regressive process and achieves better real-time factor(RTF). This decoding mode is called **rescoring**.

2.2. Patient knowledge distillation (PKD)

Patient knowledge distillation (PKD) is a common method for the typical teacher-student offline distillation, which distills the teacher's knowledge to the student. PKD has two methods called PKD-Skip and PKD-Last. In PKD-Skip, the student learns from every k layers of the teacher. In PKD-Last, the student learns from the last k layers of the teacher. For example, the teacher has 12 layers and the student has 4 layers, then $I_{pkd} = \{3,6,9,12\}$ for PKD-Skip or $I_{pkd} = \{9,10,11,12\}$ for PKD-Last. I_{pkd} refers to the teacher's knowledge that the student needs to learn. PKD-Skip performs slightly better than PKD-Last. The training loss introduced by the PKD is defined as the mean-square loss between the normalized hidden states:

$$\mathcal{L}_{\mathrm{PKD}} = \sum_{\substack{i=1 \\ j=I(i)}}^{M} \left\| \frac{\mathbf{H}_i^s}{\| \mathbf{H}_i^s \|_2} - \frac{\mathbf{H}_j^t}{\| \mathbf{H}_j^t \|_2} \right\|_2^2 \quad (2)$$

where $\mathbf{H}_i^s$ and $\mathbf{H}_j^t$ denote the outputs of the student's i-th layer and the teacher's j-th layer. M denotes the number of layers in the student network, the function I($\cdot$) refers to the set I_{pkd}.

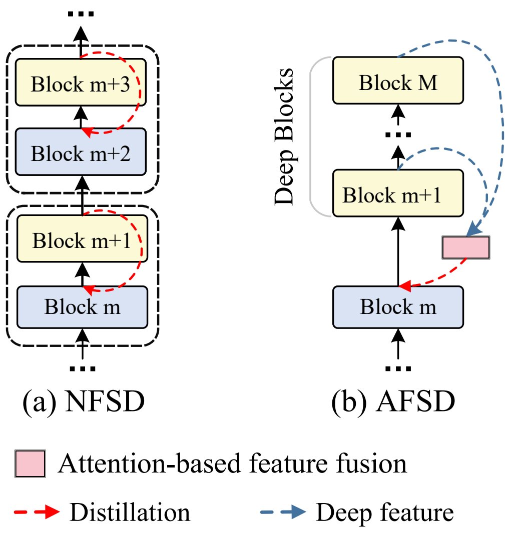

Figure 1: *Schematic diagram of the methods NFSD (a) and AFSD (b). The Block module represents the encoder or decoder layer. For NFSD, we tie every two blocks as a group and distill the feature from the higher block to another one in each group; For AFSD, We use Attention-based feature fusion module to fuse features from all higher blocks (Deep Blocks) after the current block and then distill the fused feature to the current block.*

3. Proposed methods

We propose two feature self-distillation methods (NFSD and AFSD) to improve the performance of the ASR model. NFSD distills adjacent deeper feature to the shallow feature; AFSD adopts dot attention to fuse multiple deeper features and distills the fused feature to the shallow feature. Figure 1 depicts a simple schematic of the specific distillation process of the NFSD and AFSD. Furthermore, we combine the proposed self-distillation methods with the offline distillation method to further improve the performance of the student model. In this section, we introduce the proposed methods in detail.

3.1. Neighboring Feature Self-Distillation (NFSD)

Our neighboring feature self-distillation (NFSD) method can be applied on both the encoder and decoder sides. For encoder and decoder, the principle of our self-distillation method is the same, so we only choose the encoder to describe our method. For ease of description, we denote the hidden output of each encoder layer as $\{\mathbf{H}_1^{\mathrm{E}},\ldots,\mathbf{H}_{L_e}^{\mathrm{E}}\}$, where L_e represents the number of the encoder layers. We tie every two layers of the L_e layers into $\frac{L_e}{2}$ groups and calculate the similarity of the two layers in each group. In each group, the shallow layer learns the feature representations from the adjacent deeper layer, this is because the feature representations of adjacent layers are the most similar. We use mean square error (MSE) to measure the similarity between the two layers in each group, as shown in Equation (3), where $\mathbf{H}_{2i-1}^{\mathrm{E}}$ and $\mathbf{H}_{2i}^{\mathrm{E}}$ refer to the outputs of two layers in the i-th group.

$$\mathcal{L}_{nfsd\_e} = \sum_{i=1}^{\frac{L_e}{2}} \mathrm{MSE}(\mathbf{H}_{2i-1}^{\mathrm{E}}, \mathbf{H}_{2i}^{\mathrm{E}}) \quad (3)$$

Since the decoder is the same as the encoder, the total self-distillation loss function of the NFSD is as follows:

$$\mathcal{L}_{\mathrm{NFSD}} = \alpha\mathcal{L}_{nfsd\_e} + \beta\mathcal{L}_{nfsd\_d} \quad (4)$$

where $\mathcal{L}_{nfsd\_e}$ and $\mathcal{L}_{nfsd\_d}$ are the self-distillation loss functions of the encoder and decoder sides. α and β are hyperparameters used to balance the importance of the two losses.

With NFSD, the training loss function of our entire model becomes:

$$\mathcal{L} = \mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{NFSD}} \tag{5}$$

where $\mathcal{L}_{\mathrm{U2}}$ represents the original loss function of U2, $\mathcal{L}_{\mathrm{NFSD}}$ is designed by ourselves. In $\mathcal{L}_{\mathrm{NFSD}}$, if α is equal to zero, we only distill the decoder side of the model. β is equal to zero means that only the encoder side is distilled.

3.2. Attention-based Feature Self-Distillation (AFSD)

NFSD does not fully utilize all high-level information. We further propose the AFSD method to solve this problem. AFSD fuses multiple deep features by dot-attention, then distills the fused feature to the shallow feature. Similarly, we still only describe the method details on the encoder side. Differently, in AFSD, the teacher of each layer is not its next layer but the fusion of all layers' features after it, as follows:

$$\mathbf{K}_i^{\mathrm{T}} = \sum_{j=i+1}^{L_e} \alpha_{i,j} \odot \mathbf{H}_j^{\mathrm{E}} \quad (i = 1, \ldots, L_e - 1) \tag{6}$$

$$\alpha_{i,j} = \frac{\exp(\mathrm{DotAttention}(\mathbf{H}_i^{\mathrm{E}}, \mathbf{H}_j^{\mathrm{E}}))}{\sum_{j'=i+1}^{L_e} \exp(\mathrm{DotAttention}(\mathbf{H}_i^{\mathrm{E}}, \mathbf{H}_{j'}^{\mathrm{E}}))} \tag{7}$$

where $\odot$ indicates the Hadamard product, $\mathbf{K}_i^{\mathrm{T}}$ is the knowledge that the i-th layer needs to learn. Then we sum all layers except the last layer (without distilled) to get the distillation loss function $\mathcal{L}_{afsd\_e}$ of the encoder side as Equation (8). Equation (9) is the total self-distillation loss function of AFSD.

$$\mathcal{L}_{afsd\_e} = \sum_{i=1}^{L_e - 1} \mathrm{MSE}(\mathbf{H}_i^{\mathrm{E}}, \mathbf{K}_i^{\mathrm{T}}) \tag{8}$$

$$\mathcal{L}_{\mathrm{AFSD}} = \alpha \mathcal{L}_{afsd\_e} + \beta \mathcal{L}_{afsd\_d} \tag{9}$$

where the meanings of α and β are the same as those in Equation (4). With AFSD, the training loss function of the entire model is as follows:

$$\mathcal{L} = \mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{AFSD}} \tag{10}$$

3.3. NFSD-offline and AFSD-offline

Since NFSD and AFSD only use the knowledge of the student model itself without the knowledge of the teacher, we further combine self-distillation with offline distillation, called NFSD-offline and AFSD-offline. We choose the most commonly used PKD [14] method as the offline distillation method, and self-distillation is our proposed NFSD and AFSD methods. The NFSD-offline and AFSD-offline methods are divided into two stages. The first stage is distilling a large teacher model to a smaller student model with the PKD method, as follows:

$$\mathcal{L} = \mathcal{L}_{\mathrm{U2}} + \gamma \mathcal{L}_{\mathrm{PKD}} \tag{11}$$

where γ is a hyperparameter that balances the importance of the PKD loss. $\mathcal{L}_{\mathrm{PKD}}$ is Equation (2), which refers to the training loss function of the PKD method. The second stage is to further apply the NFSD and AFSD methods to the student model obtained after offline distillation for self-distillation, the training loss function is the same as Equation (5) and Equation (10).

4. Experiments

All our experiments are conducted on a public Mandarin speech corpus AISHELL-1 [28], which contains 150 hours training set, 20 hours development set and 10 hours test set, the test set contains 7,176 utterances recorded by 20 speakers.

Table 1: *Results of applying two different training strategies on NFSD. One is one-stage training strategy (E1-E2), another is two-stage training strategy (E3-E4). $\mathcal{L}_{\mathrm{NFSD}}$ refers to $\alpha\mathcal{L}_{nfsd\_e} + \beta\mathcal{L}_{nfsd\_d}$. α/β means α equals to β.*

Exp.ID	α/β	**Loss ($\mathcal{L}$)**		**CER**	
		first-stage	second-stage	attention	rescoring
E1	0.1	$\mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{NFSD}}$	-	6.92	7.00
E2	0.2	$\mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{NFSD}}$	-	6.82	6.97
E3	0.1	$\mathcal{L}_{\mathrm{U2}}$	$\mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{NFSD}}$	6.75	6.74
E4	0.2	$\mathcal{L}_{\mathrm{U2}}$	$\mathcal{L}_{\mathrm{U2}} + \mathcal{L}_{\mathrm{NFSD}}$	**6.73**	**6.66**

4.1. Baseline models

(1) **Teacher**: The teacher model contained 47M parameters. It used two convolution sub-sampling layers with kernel size 3*3 and stride 2 as subsampling module in the front of the shared encoder. In addition, the teacher contained 12 conformer encoder layers and 6 transformer decoder layers with 4 multi head attention for Shared Encoder and Attention Decoder. All conformer and transformer layers used 256 attention dimensions and 2048 feed forward dimensions.

(2) **Student**: The student model contained 13.9M parameters. The subsampling module of the student was the same with the teacher. Differently, the student contained 4 conformer encoder layers and 2 transformer decoder layers with 4 multi head attention for Shared Encoder and Attention Decoder. All conformer and transformer layers used 256 attention dimensions and 1024 feed forward dimensions.

(3) **PKD**: The PKD model was the student model obtained by distilling the teacher model. Its model structure was the same with **Student**. The teacher-student offline distillation method PKD [14] was used in the training process.

4.2. Experimental setup

We used 80 dimensional log-mel filter bank (Fbank) computed on 25 ms window with 10 ms shift as feature. We also used SpecAugment [29]. Our modeling units included 4233 characters (including a padding symbol <pad>, an unknown symbol <unk>, and a start-or-end-of-sentence symbol <sos/eos>). We set γ in Equation (11) to be 0.2. In the training stage, we used Adam optimizer [30] and a varying warmup learning rate [25] with warmup steps set to 25000. In the inference stage, we used two streaming decoding algorithms to measure the performance of the model, attention for autoregressive decoding and rescoring for non-autoregressive decoding. The decoding chunk size was 16, and the latency was about 640 ms. Moreover, we averaged the top 20 best models, which had a lower loss on the dev set at the training stage as the final test model. For the U2 teacher model and student model, we used wenet [31] end-to-end speech recognition toolkit for all experiments.

4.3. Comparison of different training strategies on NFSD

We tried two different training strategies on the NFSD method in Table 1: one was a one-stage strategy (E1-E2) to directly train the student model from scratch with Equation (5) until fully converged; another was a two-stage training (E3-E4), pre-training the student model with Equation (1) until fully converged firstly, and then training the student model with Equation (5) until fully converged. For α and β in Equation (4), we set

Table 2: *CERs of different knowledge distillation methods on three baseline models. where $\mathcal{L}_{\text{NFSD}}$ $(0.2\mathcal{L}_{nfsd\_e} + 0.2\mathcal{L}_{nfsd\_d})$ and $\mathcal{L}_{\text{AFSD}}$ $(0.2\mathcal{L}_{afsd\_e} + 0.2\mathcal{L}_{afsd\_d})$ are from Equation (4) and Equation (9). $\mathcal{L}_{\text{U2}}$ and $\mathcal{L}_{\text{PKD}}$ are Equation (1) and Equation (2).*

Exp. ID	**System**	**Params (M)**	**Loss ($\mathcal{L}$)**		**CER**	
			first-stage	**second-stage**	**attention**	**rescoring**
T0	Teacher (baseline)	47	$\mathcal{L}_{\text{U2}}$	-	5.95	6.01
T1	+ NFSD	47	$\mathcal{L}_{\text{U2}}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{NFSD}}$	5.76	5.62
T2	+ AFSD	47	$\mathcal{L}_{\text{U2}}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{AFSD}}$	**5.67**	**5.56**
T3	Student (baseline)	13.9	$\mathcal{L}_{\text{U2}}$	-	7.11	7.23
T4	+ NFSD	13.9	$\mathcal{L}_{\text{U2}}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{NFSD}}$	6.73	6.66
T5	+ AFSD	13.9	$\mathcal{L}_{\text{U2}}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{AFSD}}$	**6.68**	**6.63**
T6	PKD (baseline)	13.9	$\mathcal{L}_{\text{U2}} + 0.2\mathcal{L}_{\text{PKD}}$	-	6.85	6.76
T7	+ NFSD (NFSD-offline)	13.9	$\mathcal{L}_{\text{U2}} + 0.2\mathcal{L}_{\text{PKD}}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{NFSD}}$	6.55	6.33
T8	+ AFSD (AFSD-offline)	13.9	$\mathcal{L}_{\text{U2}} + 0.2\mathcal{L}_{PKD}$	$\mathcal{L}_{\text{U2}} + \mathcal{L}_{\text{AFSD}}$	**6.53**	**6.26**

α equal to β. From the results shown in Table 1, the two-stage training strategy performed better than one-stage and it could obtain the best results when the weights of α and β were set to 0.2, so all our next experiments used the two-stage training strategy, and α and β were both fixed to 0.2.

4.4. Effects of directly decreasing parameters

In Table 2, T0 and T3 were our pre-trained baseline teacher and student models, trained with Equation (1). We compressed the teacher's encoder to 4 layers and decoder to 2 layers, the compression ratio was about 3.37 times. Due to the rapid reduction in the number of encoder and decoder layers, the student's ability of acoustic feature representation and language modeling became poorer than teacher, so the student model had a relative drop of **19.5%** and **20.3%** in the attention and rescoring decoding modes, respectively.

4.5. Effects of NFSD and AFSD

All the results were shown in Table 2. First, we verified the NFSD method on the baseline student model. From the results of T4, the NFSD method had greatly improved the performance of the student model and even surpassed the PKD model (T6). It argued that the deeper layers of the model itself had more efficient feature representation, which could be easily integrated into shallower structures. In this case, we further proposed the AFSD method to exploit more high-level information. T5 showed that the AFSD method achieved a more efficient performance improvement than NFSD, achieving **6.0%** and **8.3%** relative CER reduction in the attention and rescoring decoding modes compared with the baseline student model (T3). Since our proposed methods did not require an extra teacher model, we also adopted them to the baseline teacher model. The results of T1 and T2 showed that our methods could greatly improve the performance of the teacher model, and AFSD was also more efficient than NFSD. In addition, the AFSD method obtained **4.7%** and **7.5%** relative CER reduction in the attention and rescoring decoding modes compared with the baseline teacher model (T0). This showed that more high-level information is more effective for self-distillation.

4.6. Effects of NFSD-offline and AFSD-offline

We further verified the effectiveness of the combination of offline distillation and self-distillation. In Table 2, T6 was our baseline PKD model, which was obtained by distilling the baseline teacher model (T0) with the PKD method [14]. T7 and T8 were the results of the NFSD-offline and AFSD-offline methods. Among them, the AFSD-offline method improved more significantly, achieving **8.2%** and **13.4%** relative CER reduction compared with the baseline student model (T3) in the attention and rescoring decoding modes. In addition, the AFSD-offline method also achieved **4.7%** and **7.4%** relative CER reduction compared with the PKD model (T6) in the attention and rescoring decoding modes. It was worth noting that the AFSD-offline method only degraded 4.2% relative CER in the rescoring decoding compared with the baseline teacher model (T0), which was almost a lossless knowledge distillation. This showed that high-level information could still be exploited from self-distillation even when the feature representation is good enough. It argued that finding efficient deep feature representations was necessary.

5. Conclusions and future work

In this paper, we proposed NFSD and AFSD self-distillation methods to improve the performance of the ASR model. The proposed methods solved the problem of less information utilization of the model itself, and explored the potential of high-level information for self-distillation. Our methods worked on both large-scale teacher models and small-scale student models. For the compressed student model, our proposed methods even achieved better performance than the offline distillation PKD method, without a large teacher model. In addition, both of our methods only introduced an extra loss function during training without increasing the amount of parameters. Furthermore, we proposed NFSD-offline and AFSD-offline methods to further improve the performance of the compressed student model, basically achieving lossless knowledge distillation with the compression ratio of 3.37 times. The methods provided a possibility for the combination of offline distillation and self-distillation. In future work, we will improve the existing teacher-student knowledge distillation methods to further improve the performance of the combined AFSD-offline method.

6. Acknowledgements

This work was supported partially by the National Key RD Program of China under Grant 2018YFB1305200, the National Natural Science Foundation of China under Grant 62176182 and Alibaba Group through Alibaba Innovative Research Program.

7. References

[1] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[3] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[4] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition." in *Interspeech*, 2017, pp. 939–943.

[5] E. Battenberg, J. Chen, R. Child, A. Coates, Y. G. Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

[6] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," *arXiv preprint arXiv:1712.05382*, 2017.

[7] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[8] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[9] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.

[10] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[11] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[12] B. Zhang, D. Wu, Z. Yao, X. Wang, F. Yu, C. Yang, L. Guo, Y. Hu, L. Xie, and X. Lei, "Unified streaming and non-streaming two-pass end-to-end model for speech recognition," *arXiv preprint arXiv:2012.05481*, 2020.

[13] N. Moritz, T. Hori, and J. Le Roux, "Triggered attention for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5666–5670.

[14] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "Patient knowledge distillation for bert model compression," *arXiv preprint arXiv:1908.09355*, 2019.

[15] R. M. Mun'im, N. Inoue, and K. Shinoda, "Sequence-level knowledge distillation for model compression of attention-based sequence-to-sequence speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6151–6155.

[16] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[17] M. Huang, Y. You, Z. Chen, Y. Qian, and K. Yu, "Knowledge distillation for sequence model." in *Interspeech*, 2018, pp. 3703–3707.

[18] S. Panchapagesan, D. S. Park, C.-C. Chiu, Y. Shangguan, Q. Liang, and A. Gruenstein, "Efficient knowledge distillation for rnn-transducer models," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5639–5643.

[19] R. Masumura, N. Makishima, M. Ihori, A. Takashima, T. Tanaka, and S. Orihashi, "Hierarchical transformer-based large-context end-to-end asr with large-context knowledge distillation," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5879–5883.

[20] T. Doutre, W. Han, M. Ma, Z. Lu, C.-C. Chiu, R. Pang, A. Narayanan, A. Misra, Y. Zhang, and L. Cao, "Improving streaming automatic speech recognition with non-streaming model distillation on unsupervised data," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6558–6562.

[21] R. V. Swaminathan, B. King, G. P. Strimel, J. Droppo, and A. Mouchtaris, "Codert: Distilling encoder representations with co-learning for transducer-based speech recognition," *arXiv preprint arXiv:2106.07734*, 2021.

[22] V. Nagaraja, Y. Shi, G. Venkatesh, O. Kalinli, M. L. Seltzer, and V. Chandra, "Collaborative training of acoustic encoders for speech recognition," *arXiv preprint arXiv:2106.08960*, 2021.

[23] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Universal asr: Unify and improve streaming asr with full-context modeling," *arXiv preprint arXiv:2010.06030*, 2020.

[24] T. Moriya, T. Ochiai, S. Karita, H. Sato, T. Tanaka, T. Ashihara, R. Masumura, Y. Shinohara, and M. Delcroix, "Self-distillation for improving ctc-transformer-based asr systems." in *INTERSPEECH*, 2020, pp. 546–550.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[26] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[27] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 4835–4839.

[28] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.

[29] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] Z. Yao, D. Wu, X. Wang, B. Zhang, F. Yu, C. Yang, Z. Peng, X. Chen, L. Xie, and X. Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," *arXiv preprint arXiv:2102.01547*, 2021.