INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Collaborative Training of Acoustic Encoders for Speech Recognition

Varun Nagaraja[*], *Yangyang Shi*[*], *Ganesh Venkatesh, Ozlem Kalinli, Michael L. Seltzer, Vikas Chandra*

Facebook, USA

{vnagaraja, yyshi}@fb.com

Abstract

On-device speech recognition requires training models of different sizes for deploying on devices with various computational budgets. When building such different models, we can benefit from training them jointly to take advantage of the knowledge shared between them. Joint training is also efficient since it reduces the redundancy in the training procedure's data handling operations. We propose a method for collaboratively training acoustic encoders of different sizes for speech recognition. We use a sequence transducer setup where different acoustic encoders share a common predictor and joiner modules. The acoustic encoders are also trained using co-distillation through an auxiliary task for frame level chenone prediction, along with the transducer loss. We perform experiments using the LibriSpeech corpus and demonstrate that the collaboratively trained acoustic encoders can provide up to a 11% relative improvement in the word error rate on both the test partitions.

Index Terms: speech recognition, knowledge distillation, co-distillation, collaborative training, transformer

1. Introduction

Speech interfaces are accessible today on many edge devices (e.g., mobile phone and smart speaker), which possess very different computational capabilities. Hence, it has become necessary to build speech recognition models of different sizes to accommodate the wide variety of computational resources. Many automatic speech recognition models [1, 2, 3] usually are proposed in a few different size configurations, which fall into the relative categories of small, medium, and large. Typically, the models of different sizes are trained independently of each other. This leads to inefficiency in the training steps due to repeated data loading and manipulation. The separately trained models might also be sub-optimal in performance since there is no knowledge shared between them.

One of the popular approaches for sharing knowledge between the models is Knowledge Distillation [4] (KD). The traditional KD technique is a two-step procedure that trains a teacher model in the first step and uses the teacher's supervision to train a student model in the second step. While this procedure can help in improving the model's performance, it increases the total time for building a model due to the serialized training procedure. The serialization overhead can be avoided using a single-step version of knowledge distillation referred to as co-distillation [5, 6]. Co-distillation trains models jointly and has improved the performance and training efficiency of image classifiers [7]. We propose a method that applies co-distillation to train a group of speech recognition models of different sizes.

A widely used on-device automatic speech recognition (ASR) model is the sequence transducer network [8]. A sequence transducer model consists of an acoustic encoder, a predictor, and a joiner network. The commonly used acoustic encoders are multi-layered RNN/LSTM [9, 10] and Transformers [1, 2, 3, 11]. In this work, we use the low latency streaming Transformer [12] based encoder proposed in the Emformer [2] architecture.

In the sequence transducer network, the acoustic encoder is the major contributor to the model's size and the computation cost. The acoustic encoder has a deeper structure than the predictor and the joiner. Hence, we design an on-device ASR model to have a configurable acoustic encoder with a shared predictor and a joiner network. A group of acoustic encoders are trained jointly in a single sequence transducer model. During deployment, an appropriate size encoder can be selected depending on the computational constraints.

We construct a group of acoustic encoders of different sizes by using different number of layers. The encoders share a set of low-level layers and branch off into additional layers according to the size requirements. An auxiliary task projects the last layer outputs from each branch to logits used for frame-level chenone [13] prediction. Such an auxiliary task was shown to be helpful by Liu *et al.* [14] for improving acoustic encoders in sequence transducer model. Along with the benefit of training for the auxiliary task in each branch, we apply a KL divergence loss on the output probabilities, which helps in knowledge distillation between the encoders.

Our proposed collaborative training method is a novel framework for single-step training of acoustic encoders of different sizes in a speech recognition model. We demonstrate on the LibriSpeech dataset [15] that the collaboratively trained models can provide a 3-11% relative improvement in the word error rate when compared to the models trained separately. The joint training also helps in sharing the computational resources since we can perform operations, e.g., data loading and manipulation, just once for all the models.

2. Related work

Knowledge distillation (KD) techniques have been used in the context of speech recognition for model compression [16, 17, 18], domain adaptation [19, 20, 21, 22, 23] and transferring knowledge from full-context to streaming scenarios [24, 25]. These methods have applied KD both at the sequence level [17, 18], and the frame-level [16, 23]. The early works on sequence level KD [26, 24] used a two-step procedure. However, a recently proposed method by Panchapagesan *et al.* [18] allows for single-step co-distillation in RNNT models. Yu *et al.* [25] used this loss function for training encoder modules capable of working in both streaming and full-context speech recognition scenarios. Wu *et al.* [27] applied the sequence level KD to train models of different sparsity levels. Their method results in unstructured sparsity that needs specialized implementation to

[*] Equal Contribution

http://dx.doi.org/10.21437/Interspeech.2021-354

fully exploit the computational benefit of sparsity. Our method varies the number of layers in the encoders to be able to reuse existing implementations for easy deployment. The methods [16, 23] which perform frame-level KD, are usually for acoustic encoders used in a hybrid system. These methods are also a two-step procedure that requires a trained teacher model. Our proposed method is a single-step procedure with frame-level co-distillation and uses a shared predictor and joiner modules for implicitly sharing the sequence level knowledge between the acoustic encoders.

Co-distillation based collaborative learning methods [7, 28, 29] have been used for image classification. Lan *et al*. [7] proposed the idea of using a multi-branch network with shared low-level layers and training the branches using co-distillation. We apply a similar concept and combine it with the auxiliary task idea of Liu *et al*. [14] for frame-level co-distillation.

Configurable neural networks are a class of models that are trained once and deployed in different configurations. There are two categories of such methods. In the first category [30, 31], an appropriate network setting is used for inference based on a predetermined computational budget. In the second category, the methods train networks to dynamically adjust their resources during inference. The second category can be further divided based on whether the methods allow an external agent to control the resources during inference. Some methods [32, 33, 27] can adjust the forward pass pathway in the network to reduce the computation based on the input. Another set of methods called *anytime inference* [34, 35] allow an external agent to stop the computation at any point and get the best possible prediction. Our method falls into the first category, where an appropriate acoustic encoder is selected based on a specific device's constraint.

3. Collaborative Training of Encoders

3.1. Low Latency Emformer Transducer

We focus on building models for low latency streaming on-device speech recognition using the Emformer [2] transducer. Emformer is an efficient extension of the Augmented Memory Transformer (AM-TRF) [1]. They both perform streaming Transformer [12] based speech recognition by splitting an utterance into multiple segments, and decoding a segment along with the context of surrounding segments. The model size and the computation cost of an Emformer encoder are determined by the input dimension, size of the feed-forward network, and the number of layers. We demonstrate a training methodology for a group of Emformer encoders of different depths.

3.2. Collaborative Training

As shown in Figure 1, we design a group of Emformer encoders with a shared set of layers F that then splits into multiple branches $\{B_0, B_1, \ldots, B_{n-1}\}$ depending on the desired number of encoders based on the size. The number of layers in the shared network is denoted by L_F, and the number of layers in a branch B_i is denoted by L_{B_i}. For deployment and decoding, we extract a copy of the shared layers and only one branch of the encoder based on the specific device requirement. Hence, the effective number of layers during the decoding is $L_F + L_{B_i}$ for a model created from branch B_i.

Figure 1 shows that the predictor and the joiner network are shared with all the encoder branches. Given the input acoustic feature sequence $\boldsymbol{x} = \{x_1, \ldots, x_T\}$ with sequence length T and target token sequence $\boldsymbol{y} = \{y_1, \ldots, y_U\}$ with length U

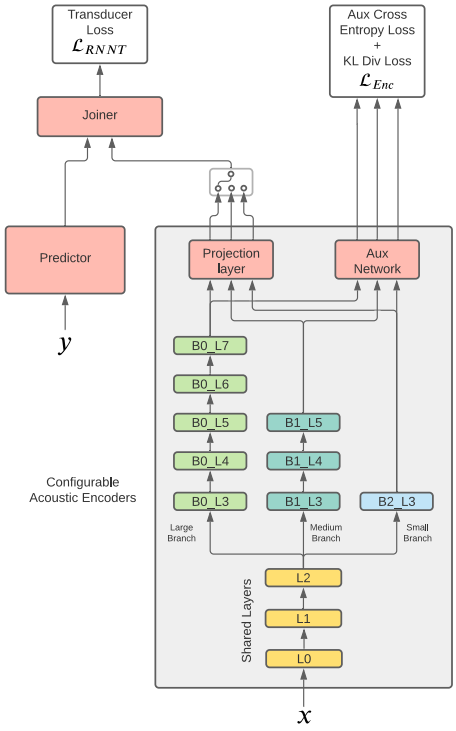

Figure 1: *Block diagram of the training setup for the collaborative acoustic encoders. The nodes in yellow are the shared layers between the branches.* B_0, B_1 *and* B_2 *are three different encoders with different sizes.*

where $y_u \in \mathcal{Y}$, we can get the embedding representation $\boldsymbol{h}_i^x$ for each encoder B_i and $\boldsymbol{h}^y$ for the predictor as follows:

$$\boldsymbol{h}_i^x = q(f_i(\boldsymbol{x})) \tag{1}$$

$$\boldsymbol{h}^y = g(\{\varnothing, \boldsymbol{y}\}). \tag{2}$$

where $\varnothing$ is a blank token. f_i and g are the projections from the acoustic encoder B_i and the predictor, respectively. q is the shared projection layer for each acoustic encoder. By combining each encoder's hidden representation with the predictor's output, the joiner generates the logits for each encoder B_i.

$$\boldsymbol{l}_i = z(\boldsymbol{h}_i^x, \boldsymbol{h}^y). \tag{3}$$

where each element $l_{(t,u),i}$ in $\boldsymbol{l}_i$ is the logit given the acoustic sequence $\{x_1, ..., x_t\}$ and label sequence $\{\varnothing, y_1, ..., y_u\}$. We can get the transducer loss [8] for each acoustic encoder B_i by passing the logits through a softmax function and applying the forward and backward algorithm. We use the sum of all the encoders' transducer loss as the final transducer loss for the sequence transducer model with multiple acoustic encoders.

$$\mathcal{L}_{Trans} = -\sum_{i=0}^{n-1} \log P_i(\boldsymbol{y}|\boldsymbol{x}) \tag{4}$$

The term $P_i(\boldsymbol{y}|\boldsymbol{x})$ is the summation of all the alignment probabilities $P_i(\boldsymbol{a}|\boldsymbol{x})$ where $\boldsymbol{a}$ is an alignment with elements $a_t \in \{\mathcal{Y} \cup \varnothing\}$. The removal of blank tokens from $\boldsymbol{a}$ gives the target sequence $\boldsymbol{y}$. In the work of Liu *et al*. [14], the outputs from the intermediate layers of an acoustic encoder are also

connected to the transducer loss through a joiner. However, its purpose is to stabilize the deep encoder training where the gradients are passed back to the encoder without updating the weights of the joiner using the intermediate transducer losses. In contrast to their work, we update the joiner weights using the transducer losses obtained from the embeddings of all the encoder branches.

The proposed collaborative training method also includes a shared auxiliary network that projects the last layer outputs from each of the branches to logits for predicting the frame-level chenone [13] targets. The auxiliary network consists of a hidden layer with RELU activation followed by an output layer. The same auxiliary network (MLP_{aux}) is used for all the branches. The output from the auxiliary network is passed through a softmax operator that produces a probability distribution $P_i(\boldsymbol{s}|\boldsymbol{x})$ over the chenone targets given the embedding representation from an encoder branch B_i. It is defined as

$$P_i(\boldsymbol{s}|\boldsymbol{x}) = \text{softmax}(\text{MLP}_{\text{aux}}(f_i(\boldsymbol{x})) \tag{5}$$

where $\boldsymbol{s}$ is the frame level sequence of chenone targets.

We apply the cross-entropy loss for chenone prediction from each branch and also a KL divergence loss between pairs of branches. Since the number of pairs can blow up combinatorially, we identify the largest encoder branch and create pairs with each of the remaining branches. The largest branch plays a role as a teacher in knowledge distillation. The loss for the outputs from the encoder branches can be written as

$$\mathcal{L}_{CE} = -\sum_{i=0}^{n-1} \log P_i(\boldsymbol{s}|\boldsymbol{x}) \tag{6}$$

$$\mathcal{L}_{KL} = -\sum_{i \neq K} P_K(\boldsymbol{s}|\boldsymbol{x}) \log \frac{P_i(\boldsymbol{s}|\boldsymbol{x})}{P_K(\boldsymbol{s}|\boldsymbol{x})} \tag{7}$$

$$\mathcal{L}_{Enc} = \mathcal{L}_{CE} + \mathcal{L}_{KL} \tag{8}$$

where K is the index for branch B_K with the largest size. $\mathcal{L}_{CE}$ in Eqn 6 is the cross entropy loss with the groundtruth and $\mathcal{L}_{KL}$ in Eqn 7 is the KL divergence loss with the soft targets from the largest branch. The entire model with all the branches is trained jointly in an end-to-end manner and the overall loss function is given by

$$\mathcal{L} = \sum_{\mathcal{D}} (\mathcal{L}_{Trans} + \lambda \mathcal{L}_{Enc}) \tag{9}$$

where λ is a hyperparameter to control for the weight given to the encoder loss and $\mathcal{D}$ represents the training dataset.

4. Experiments

4.1. Datasets and Setup

We perform experiments using the LibriSpeech corpus [15]. The LibriSpeech corpus contains around 960 hours of training data and around 5 hrs each of "dev-clean", "dev-other", "test-clean", "test-other" datasets.

The input acoustic features are 80-dimensional log Mel filter bank energies. The width of an audio frame is 25ms with a stride of 10ms. We augment the data using speed perturbation [36] and SpecAugment [37] without the time warping modifier. The 80 dimensional features for each audio frame are projected to a 128 dimensional vector and the frames are concatenated at a stride of 4 to form a 512 dimensional vector. The input dimension of each Emformer layer is 512, the number of heads in the self-attention layer is 8 and the size of the feed-forward network is 2048. A final projection layer outputs a 1024 dimensional vector to pass to the joiner. We use a dropout of 0.1 in the self-attention and the feed-forward layers. Since we perform experiments for low latency conditions, the center chunk size and look-ahead context size in Emformer are set to 160ms and 40ms, respectively. The algorithmic latency [2] of the acoustic encoders is 120 ms.

Table 1: *Number of parameters in the Speech Recognition model based on the number of layers in the acoustic encoder.*

Num Layers	20	18	14	10	7
Num Params (M)	76.7	70.4	57.8	45.2	35.7

The predictor is a three layer LSTM with an input and hidden dimension of 512 and the output is projected to 1024 dimensions before passing it to the joiner. The joiner adds the acoustic and label embeddings and projects it to the target space of word pieces obtained using SentencePiece [38]. For experiments which use the auxiliary task, the frame level chenone [13] targets are generated by forced alignment using a hybrid HMM-GMM system. This hybrid model was bootstrapped using the standard Kaldi [39] LibriSpeech recipe.

We use the Alignment Restricted RNNT [40] loss for training the models. All the models are trained for 120 epochs using a learning rate of 1e-3 with the ADAM optimizer. The learning rate is increased linearly from 2e-7 to 1e-3 over 10K warmup updates for all the experiments. We use the last checkpoint for performing decoding.

We experiment with acoustic encoders of different number of layers. The total number of parameters including the predictor and the joiner can be found in Table 1.

4.2. Results

We compare the performance of the collaboratively trained models with the models trained separately. We use a weight of 0.6 for the auxiliary cross-entropy loss when training the models individually based on the observation from the work [14]. When training the models collaboratively, we use a weight of 0.1 for the auxiliary cross-entropy loss and the distillation loss. Table 2 shows the word error rate results based on collaborative training method using 3 different encoders with the different weight values (λ in Eqn. 9) used during the collaborative training. We observe no single value that performs consistently well across all the model sizes and the dataset partitions. A value of 0.05 gives better results on the test-other partition, and a value of 0.2 gives better results on the test-clean partition for the 20 and 14 layers models. Finally, we use the value of 0.1 since it performs better on average across the model sizes.

Table 2: *Word error rate (WER) results from using different weight value for the encoder loss during collaborative training. The two numbers in each cell are the WER results on test-clean and test-other datasets, respectively.*

	Num Layers		
$\boldsymbol{\lambda}$	**20**	**14**	**7**
0.05	3.47, **9.14**	3.78, **9.67**	**4.84**, 12.80
0.1	3.50, 9.23	3.57, 9.80	4.88, **12.21**
0.2	**3.38**, 9.19	**3.56**, 9.74	4.90, 13.41
0.3	3.43, 9.21	3.60, 9.68	5.00, 13.18

The baseline models in our experiments are the ones trained separately. The first row of Table 3 shows the word error rate

Table 3: *Word error rate (WER) results from models trained with and without the different factors of the collaborative learning method. We also include results for collaboratively trained models with different number of branches. The two numbers in each cell are the WER results on test-clean and test-other datasets respectively. Bold values indicate the best performing model in a group when compared to the baselines. All the encoders have the same algorithmic latency of 120ms with 160ms chunk size and 40ms look-ahead context size.*

				Num Layers				
Shared Predictor and Joiner	**Shared Encoder Layers**	**Aux task with CE loss**	**Aux task with CE+KLDiv Loss**	**20**	**18**	**14**	**10**	**7**
-	-	-	-	3.76, 9.86	3.83, 9.82	3.87, 10.35	4.29, **11.31**	4.65, 12.50
-	-	y	-	3.58, 10.48	3.80, 10.91	3.94, 11.29	4.19, 11.76	4.75, 12.81
				4 branches				
y	-	-	-	4.18, 10.46	4.35, 10.55	4.52, 11.06	4.97, 12.01	-
y	-	y	-	3.90, 9.61	3.97, 9.92	4.25, 10.95	4.53, 11.68	-
y	-	y	y	3.83, 9.72	3.87, 9.69	4.10, 10.40	4.53, 11.33	-
y	y	-	-	3.69, 9.95	3.83, 10.16	3.96, 10.34	4.42, 11.85	-
y	y	y	-	3.52, 9.39	3.47, **9.49**	**3.70**, 10.24	**4.02**, 11.53	-
y	y	y	y	**3.45, 9.34**	**3.39**, 9.57	3.72, **10.10**	4.17, 11.57	-
				3 branches				
y	-	y	y	**3.38, 8.87**	-	**3.57, 9.77**	-	**4.38, 11.84**
y	y	y	y	3.50, 9.23	-	3.57, 9.80	-	4.88, 12.21
				2 branches				
y	y	y	y	3.49, 9.36	-	3.72, 9.95	-	-

(WER) results from the baseline model trained using the transducer loss without any auxiliary task. The second row of Table 3 shows the WER from different models trained using transducer loss together with the frame-level cross-entropy loss based on the chenone alignment. We observe that the performance is mixed when using the auxiliary task. This could indicate that the original weight for the auxiliary loss proposed in [14] might have been sub-optimal and requires additional tuning.

There are four critical factors in the proposed collaborative training method: shared predictor and joiner, shared low-level encoder layers, an auxiliary task with cross-entropy (CE) loss, and the auxiliary task with CE and KL divergence loss. We perform an ablation study of these factors with a model of four different encoder branches. We also experiment with a different number of branches ranging from two to four. When using the setting with shared encoder layers, the first six layers are common between the different encoder branches. We show the results grouped by the number of branches in Table 3 for the experiments done using collaborative training.

The collaborative training results with four branches show that the performance incrementally improves as we enable the different factors. The WER of the model that uses all the four factors of our proposed method improves by 3-11% and 2.5-5.5% on the test-clean and test-other partitions, respectively. The last two rows in the group using four branches differ in whether they use the KL divergence loss for the auxiliary task or not. The results indicate that the performance obtained from these two settings is very close to each other, but they are still significantly better than their baseline counterparts.

One of the interesting observations is that the performance of the largest branch sees an improvement along with that of the student branches. This can be attributed to the shared modules which are improved due to the training on the inputs from the different branches. An outlier result is that the ten layers model's performance did not show an improvement on the test-other partition.

Similar to the trend with the four encoder branches, the model with two encoder branches trained collaboratively by enabling all the factors also achieves significantly better results than the baseline models. The last row in Table 3 shows that the 20 layers branch gets a relative WER reduction over 7% on test-clean and 5% on test other, compared with a baseline model of the same size. The 14 layers branch gets a similar WER as the 20 layers baseline model with a 25% reduction in model size.

We observed that when training a collaborative set of three branches, the seven layers model results were worse than the baseline. Hence, we experimented with a setting that did not include the shared low-level encoder layers. Interestingly, this setting provided better performance than the other settings used for the three branches. The results show that without using layer sharing in encoders, the 20 layers model trained with collaborative learning achieves the best WER among all the cases, with over 10% improvement on the test-clean and test-other partitions. However, the training time as measured using the train wall time from the fairseq [41] logs is 28% slower when there are no shared layers between the different encoders.

5. Conclusions

We have presented a collaborative learning method to train a sequence transducer model with multiple acoustic encoders for low latency on-device ASR. Our method can train models of different sizes at once and allows picking one specific acoustic encoder for deployment that meets the computational constraint of a device. The proposed collaborative learning method takes advantage of shared predictor and joiner modules along with shared low level encoder layers to improve the performance and reduce the training time. A context-dependent graphemic state prediction task was used to provide the transducer model with forced alignment information that acted as a bridge for transferring knowledge among different acoustic encoders. The experiments on the LibriSpeech dataset based on a low latency constraint showed that the collaborative learning method can improve each acoustic encoder compared with training them separately. The large encoder got over 10% relative WER reduction on both test-clean and test-other evaluation datasets. The collaboratively trained smaller encoders also got 3-11% relative WER reduction on both evaluation sets when compared with the smaller baseline models.

6. Acknowledgements

We thank Yuan (June) Shangguan and Jay Mahadeokar for their help in setting up the experiments and discussions regarding the datasets and methods in this paper.

7. References

[1] C. Wu, Y. Wang, Y. Shi, C.-F. Yeh, and F. Zhang, "Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory," in *INTERSPEECH*, 2020.

[2] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *ICASSP*, 2021.

[3] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *INTERSPEECH*, 2020.

[4] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[5] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *CVPR*, 2018.

[6] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," in *ICLR*, 2018.

[7] X. Lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *NeurIPS*, 2018.

[8] A. Graves, "Sequence transduction with recurrent neural networks," in *Representation Learning Workshop, ICML*, 2012.

[9] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *ASRU*, 2017.

[10] Y. He, T. N. Sainath, and et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*, 2019.

[11] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP*, 2020.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[13] D. Le, X. Zhang, W. Zheng, C. Fügen, G. Zweig, and M. L. Seltzer, "From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition," in *ASRU*, 2019.

[14] C. Liu, F. Zhang, D. Le, S. Kim, Y. Saraf, and G. Zweig, "Improving rnn transducer based asr with auxiliary tasks," in *SLT*, 2020.

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *ICASSP*, 2015.

[16] Y. Chebotar and A. Waters, "Distilling knowledge from ensembles of neural networks for speech recognition," in *INTERSPEECH*, 2016.

[17] R. Pang, T. Sainath, R. Prabhavalkar, S. Gupta, Y. Wu, S. Zhang, and C.-C. Chiu, "Compression of end-to-end models," in *INTERSPEECH*, 2018.

[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.

[19] J. Li, L. M. Seltzer, X. Wang, R. Zhao, and Y. Gong, "Large-scale domain adaptation via teacher-student learning," in *INTERSPEECH*, 2017.

[20] T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono, "Domain adaptation of dnn acoustic models using knowledge distillation," in *ICASSP*, 2017.

[21] V. Manohar, P. Ghahremani, D. Povey, and S. Khudanpur, "A teacher-student learning approach for unsupervised domain adaptation of sequence-trained asr models," in *SLT*, 2018.

[22] Z. Meng, J. Li, Y. Gaur, and Y. Gong, "Domain adaptation via teacher-student learning for end-to-end speech recognition," in *ASRU*, 2019.

[23] L. Mošner, M. Wu, A. Raju, S. H. K. Parthasarathi, K. Kumatani, S. Sundaram, R. Maas, and B. Hoffmeister, "Improving noise robustness of automatic speech recognition via parallel data and teacher-student learning," in *ICASSP*, 2019.

[24] S. Kim, M. L. Seltzer, J. Li, and R. Zhao, "Improved training for online end-to-end speech recognition systems," in *INTERSPEECH*, 2018.

[25] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode asr: Unify and improve streaming asr with full-context modeling," in *ICLR*, 2021.

[26] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *EMNLP*, 2016.

[27] Z. Wu, D. Zhao, Q. Liang, J. Yu, A. Gulati, and R. Pang, "Dynamic sparsity neural networks for automatic speech recognition," in *ICASSP*, 2021.

[28] G. Song and W. Chai, "Collaborative learning for deep neural networks," in *NeurIPS*, 2018.

[29] Q. Guo, X. Wang, Y. Wu, Z. Yu, D. Liang, X. Hu, and P. Luo, "Online knowledge distillation via collaborative learning," in *CVPR*, 2020.

[30] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *ICLR*, 2018.

[31] J. Yu and T. Huang, "Universally slimmable networks and improved training techniques," in *ICCV*, 2019.

[32] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *ECCV*, 2018.

[33] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, 2018.

[34] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *ICLR*, 2018.

[35] A. Ruiz and J. Verbeek, "Anytime inference with distilled hierarchical neural ensembles," in *AAAI*, 2021.

[36] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *INTERSPEECH*, 2015.

[37] 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," in *INTERSPEECH*, 2019.

[38] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *EMNLP: System Demonstrations*, 2018.

[39] D. Povey, A. Ghoshal, G. Boulianne, N. Goel, M. Hannemann, Y. Qian, P. Schwarz, and G. Stemmer, "The kaldi speech recognition toolkit," in *ASRU*, 2011.

[40] J. Mahadeokar, Y. Shangguan, D. Le, G. Keren, H. Su, T. Le, C.-F. Yeh, C. Fuegen, and M. L. Seltzer, "Alignment restricted streaming recurrent neural network transducer," in *SLT*, 2021.

[41] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *NAACL-HLT: Demonstrations*, 2019.