Knowledge Distillation via Module Replacing for Automatic Speech Recognition with Recurrent Neural Network Transducer

Kaiqi Zhao[1], Hieu Duy Nguyen[2], Animesh Jain[2], Nathan Susanj[2], Athanasios Mouchtaris[2], Lokesh Gupta[2], Ming Zhao[1]*

[1]Arizona State University
[2]Amazon Alexa Speech

kzhao27@asu.edu, {hieng, anijain, nsusanj, mouchta, lokeshgu}@amazon.com, mingzhao@asu.edu

Abstract

Automatic Speech Recognition (ASR) is increasingly used by edge applications such as intelligent virtual assistants. However, state-of-the-art ASR models such as Recurrent Neural Network - Transducer (RNN-T) are computationally intensive on resource-constrained edge devices. Knowledge Distillation (KD) is a promising approach to compress large models by using a large model ("teacher") to train a small model ("student"). This paper proposes a novel KD method called Log-Curriculum based Module Replacing (LCMR) for RNN-T. LCMR compresses RNN-T and addresses its unique characteristics by replacing teacher modules including multiple LSTM/Dense layers with substitutional student modules that contain less Long Short Term Memory (LSTM)/Dense layers. LCMR employs a novel nonlinear Curriculum Learning driven replacement strategy to further improve the performance by updating replacing rates with a dynamic, smoothing mechanism. Under LCMR, the student and teacher are able to interact at gradient level, and tranfser knowledge more effectively than conventional KD. Evaluation shows that LCMR reduces word-error-rate (WER) by 14.47%-33.24% relative compared to conventional KD.

Index Terms: Knowledge Distillation, Module Replacing, RNN-T, Model Compression

1. Introduction

Deep neural networks are increasingly deployed to edge devices such as Internet of Things (IoTs) and provide important benefits including: 1) *Personalization*: user- or situation-specific requirements can be met more effectively; 2) *Responsiveness*: on-device models can respond faster, thus reducing latency and friction; and 3) *Privacy*: sensitive information does not leave the customer's devices. Automatic Speech Recognition (ASR), which is an important component of speech services is also moving towards edge processing. Deploying on-device ASR models however leads to multiple challenges due to the mismatch between resource-demanding ASR models and resource-constrained edge devices [1,2]. Many model compression techniques [3–9] have been proposed to address such issue. Among these efforts, one of the most promising approaches is Knowledge Distillation (KD) [10–14], which achieves model compression by using the original model to train a smaller one.

Various KD techniques have been investigated under different ASR architectures [12, 15–20]. However, existing KD approaches for ASR 1) require tedious hyper-parameter tuning to balance for the weights of different loss terms, 2) require a large pre-trained teacher model, 3) omit important gradient-level information during transferring knowledge. Recently, a Module Replacing (MR) based mechanism was proposed to compress large Bidirectional Encoder Representations from Transformers (BERT) [10] for natural language understanding (NLU) tasks. BERT and some other similar models are particularly suitable for compression as they are feed-forward only and does not have any feedback mechanism. The effectiveness of MR on recurrent architectures such as RNN-T, which have different model blocks serving different purposes, is currently unexplored.

In this work, we propose a novel KD method for RNN-T, subsequently referred as Log-Curriculum based Module Replacing (LCMR). LCMR applies MR to compress RNN-T and addresses its unique characteristics by replacing teacher modules including multiple LSTM/Dense layers with corresponding substitutional student modules that contain less LSTM/Dense layers. The replacement process is controlled by a novel logarithmic Curriculum Learning driven strategy which improves MR effectiveness by updating replacing rates with a dynamic, smoothing mechanism during training.

Compared to traditional KD approaches for RNN-T models, the proposed LCMR has several advantages. First, LCMR is hyperparameter-free, since it only uses the RNN-T loss as the training loss and does not require the weight adjustment of multiple loss terms that appear in previous KD methods. Second, during training, the gradients are calculated across all the modules of the teacher and student, allowing a gradient-level interaction. Compared to the output-level communication in previous KD approaches, LCMR thus allows the student to learn the intermediate layer state and to mimic the teacher at a more granular level, leading to faster and better distillation. Finally, unlike the existing MR works in which the teacher model knowledge is distilled into the student under a constant or linearly increasing rate, LCMR employs a non-linear Curriculum Learning [21] driven replacement strategy to further improve the performance of MR.

Our results demonstrated that LCMR outperforms traditional lattice-based KD [12] significantly on RNN-T models. Specifically, for LibriSpeech dataset, LCMR yields relative word-error-rate (WER) reductions of 14.47%-33.24% than lattice-based KD, and 20.50%-37.55% compared to training the student directly without KD. In addition, LCMR also delivers a faster convergence speed than both without KD and with lattice-based KD scenarios.

∗This work was conducted during Kaiqi's internship in Amazon.com

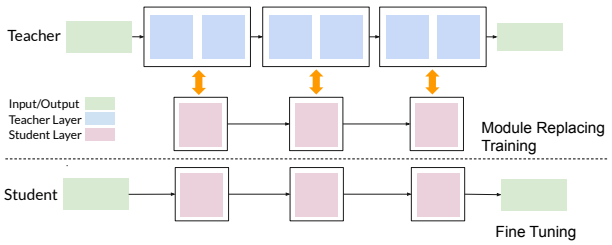


Figure 1: *Workflow of LCMR, including module replacing training and fine-tuning.*

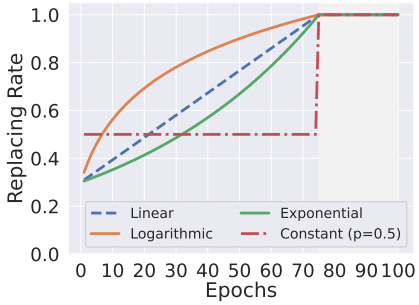


Figure 2: *The replacing curves of four replacement schedulers. The shallow shades of gray denote the fine-tuning phases (75 to 100 epochs).*

2. Background and Related works

Various KD techniques have been investigated under different ASR architectures, including Deep Neural Network - Hidden Markov Models (DNN-HMM) [15–18], Connectionist Temporal Classification (CTC) [19, 20] and Listen-Attend and Spell (LAS) models [22]. Instead of relying on transcription-level distillation, Panchapagesan et. al. proposed a lattice distillation method for RNN-T, another well-adopted ASR architecture [12]. It defines the distillation loss as the KL divergence between teacher and student posteriors, and the overall training loss as a linear combination of the RNN-T loss and the lattice-based distillation loss [12], as follows: $L = L_{rnnt} + \alpha * L_{distill}$, where α is the weight that balances the loss terms.

However, existing KD approaches for ASR, in general, and RNN-T, in particular, possess multiple drawbacks. First, nearly all KD techniques require a delicate balance for the weights of different loss terms, which leads to tedious hyper-parameter tuning, such as α in the above loss function. Second, conventional KD requires a large pre-trained teacher, which is time-consuming to create. Finally, conventional KD transfers the knowledge only from the softmax outputs, thus omitting important gradient-level information.

Recently, a KD technique based on Module Replacing (MR) mechanism was proposed for NLU tasks using BERT architecture [10]. MR replaces the teacher's, i.e., 12-layer encoder with 6 layers by a probability controlled by a constant or linearly increasing replacing rate during training. However, there are several limitations. First, it shows good performance only for compressing large NLU models (specifically BERT-based ones), but lacks the investigation of other domains, such as speech recognition. The effectiveness of MR on recurrent architectures, which have different model blocks serving different purposes such as RNN-T, is unclear. Second, it employs simple constant or linear schedulers to update the replacing rate, which mismatches the dynamic, smoothing updating process of the learning rates and leads to sub-optimal results. This paper proposes an MR-based KD method tailored for ASR models and employing a novel nonlinear replacing rate scheduler, as discussed in the rest of the paper.

Table 1: *Model configurations under two Experiment setups*

	Experiment 1		Experiment 2	
	Teacher	Student	Teacher	Student
Num. of layers (Encoder)	4	2	6	3
Num. of layers (Pred. Network)	2	1	2	1
Num. of Parameters (M)	40.61	21.71	57.4	30.11

3. Proposed Method

3.1. Module Replacing

The RNN-T architecture considered in this paper consists of encoder, prediction, and joint networks. Under LCMR, multiple LSTM layers in the teacher encoder and the prediction network are replaced with a smaller number of LSTM layers, which will combine into the student model. Figure 1 illustrates the workflow of LCMR.

We define a Module as a group of LSTM/Dense layers. Let $x = (x_1, x_2, ..., x_T)$, and $y = (y_1, y_2, ..., y_U)$ denote an input sequence with the length of T and a target token sequence with the length of U, respectively. $H^{teacher}$, $H^{student}$, $F^{teacher}$, and $F^{student}$ represent the module of the prediction network and encoder, in the teacher and student models, respectively. Consider the ith module in the prediction network, the output vectors from the teacher and student models can be described as $p_i^{teacher} = H_i^{teacher}(p_{i-1})$ and $p_i^{student} = H_i^{student}(p_{i-1})$, respectively, where p_{i-1} is the output of the $(i-1)$th module.

During the module replacing training phase, LCMR combines (possibly) different teacher and student modules in each training step. Consider training step $\mathcal{T}$ for the ith module, we generate an independent Bernoulli random variables, denoted as r_i, which is either equal to 1 or 0:

$$r_i \sim Bernoulli(p) \tag{1}$$

where $0 < p \leqslant 1$, and $r_i \in \{0, 1\}$. The replacing rate p controls the probability of r_i being 1. When r_i is equal to 1, the output of the ith module comes from the student's module, i.e., the teacher's $i$module is replaced.

The output vectors of the ith module are thus mathematically represented as follows:

$$\begin{aligned} p_i &= (1 - r_i) * p_i^{teacher} + r_i * p_i^{student} \\ &= (1 - r_i) * H_i^{teacher}(p_{i-1}) + r_i * H_i^{student}(p_{i-1}) \end{aligned} \tag{2}$$

where $*$ denotes the element-wise multiplication. Similar mechanism is applied for the jth encoder modules $F_j^{teacher}$ and $F_j^{student}$.

Considering the forward propagation, for each module, either the teacher's or the student's LSTM/Dense layers are selected to calculate the output features; and for the whole model, modules from the teacher and student are combined during forward propagation. The prediction network and encoder take $y = (y_1, y_2, ..., y_U)$ and $x = (x_1, x_2, ..., x_T)$ as the input, and generate the output features, respectively. Then, the output features are used by the joint network to compute a probability distribution $P(y|x)$.

During back-propagation, LCMR minimizes the loss:

$$L = -logP(y|x) \tag{3}$$

During the module replacing training phase, we aim to distill the knowledge from the teacher to the student by substituting student modules into teacher model. We propose two training strategies for LCMR back-propagation, depending on the availability of pre-trained teacher models. Consider the first regime with non-pretrained teacher model. In this case, LCMR calculates the gradient with respect to the all the weights from the teacher and student modules that are combined/trained during the training step $\mathcal{T}$, and back-propagated to update those weights. Under the second regime, when a pre-trained teacher is available, it freezes teacher model's weights and updates only the student modules' weights. Therefore, different from traditional KD approaches, the gradient is calculated across both the teacher and student modules, allowing a more granular distilled information.

After module replacing training, LCMR combines all student modules into a student model:

$$\begin{aligned} P^{student} &= \{H_1^{student}, H_2^{student}, ..., H_n^{student})\} \\ E^{student} &= \{F_1^{student}, F_2^{student}, ..., F_m^{student})\} \end{aligned} \tag{4}$$

where n and m are the number of modules in the prediction network and encoder, respectively.

Finally, the student model is fine-tuned for several epochs by optimizing the RNN-T loss (Equation 3).

3.2. Logarithmic Curriculum Replacement Strategy

We propose a novel logarithmic curriculum replacement strategy to further improve the performance of MR for speech recognition tasks with RNN-T. Let $\mathcal{T}$ denote the $\mathcal{T}$-th iteration/training step, so that the replacing rate $p_{\mathcal{T}}$ at iteration $\mathcal{T}$ can be calculated as:

$$p_{\mathcal{T}} = min(log_B^{k\mathcal{T}+b}, 1.0) \tag{5}$$

where B is the base value of the logarithmic function, log_B^b is the basic replacing rate, and k is the coefficient. Figure 2 illustrates the replacing curves of Constant, Linear, Logarithmic, and Exponential Replacement Schedulers. The shallow shades of gray denote the fine-tuning phases (75 to 100 epochs), in which the replacement rate $p_{\mathcal{T}} = 1$, i.e. LCMR has a full student model.

Consider the early training phase, the teacher is more likely to predict correct features so LCMR selects more modules from the teacher than the student to provide more guidance for training, by setting a small replacing rate. Meanwhile, the learning rate increases to a large value with a warm-up mechanism and stays at that value. Thus, the coordination between the replacing rate updating process and the learning rate evolving mechanism enables the student to learn from the teacher. Later, as more student modules are trained, LCMR reduces the distillation from teacher to student by setting larger, asymptotically approaching 1, replacing rates and finally fine-tunes the student only, while the gradual decaying learning rate enables the student to explore in a small range. Also, updating the replacing rate with a dynamic, smoothing mechanism is consistent with the updating process of the learning rate, thus leading to a faster and better distillation. In comparison, the constant or linear replacement scheduler results in discrepancy between replacing and learning rates, thus reducing MR effectiveness.

Table 2: *WER (%) on LibriSpeech in Experiment 1*

Method	dev	dev-other	test	test-other
Baselines				
Teacher	10.95	24.52	11	25.53
Student w/o KD	21.84	39.82	21.81	40.55
Teacher-Student				
Student + LCMR	**13.64**	**28.45**	**14.14**	**29.49**
Student + KD	20.43	37.74	20.37	38.43

Table 3: *WER (%) on LibriSpeech in Experiment 2*

Method	dev	dev-other	test	test-other
Baselines				
Teacher	7.5	16.7	7.49	17.31
Student w/o KD	13.21	28.87	13.54	29.57
Teacher-Student				
Student + LCMR	**9.87**	**22.83**	**10.04**	**23.51**
Student + KD	12.27	26.69	12.47	27.94

4. Experiment Setup

We compare LCMR with three baselines: 1) a large model that is not compressed and directly trained (Teacher), 2) a small model that is directly trained without knowledge distillation (Student w/o KD), and 3) a small model that has the same architecture with the student above, and is trained with traditional lattice-based KD (Student+KD) [12]. We denote the same small model trained with the proposed LCMR as "Student+LCMR".

Dataset. We use Librispeech dataset to train and evaluate the models, which contains 1000 hours of English speech in the form of raw audio [23]. The training dataset is split into three subsets, with approximate size 100, 360 and 500 hours. We merge all the three subsets together as one whole training dataset to train the models. The testing dataset is divided into four subsets, including dev, dev-other, test, and test-other. We evaluate the models on each of the four subsets separately.

Models. The LSTM layers in RNN-T encoder and prediction network consist of 1024 and 512 hidden units, respectively. The joint network is a feed-forward network with 512 units, followed by a softmax layer with output size 1024. The number of modeled word pieces is 1023. Table 1 shows the architecture and the number of parameters of the teacher and student models used in Experiment 1 and Experiment 2. Note that the conformer model used in [12] is non-streaming architecture with full-text audio features, which is different from ours streaming model.

Implementation Details. We use Layer-wise Adaptive Moments optimizer for Batch training (LAMB) optimizer to train the models for 100 epochs. Weight decay is set to 1e-3. The learning rate ramps up to 5e-4 linearly in the first epoch and decays to 1e-5 exponentially from 20 to 100 epochs. We apply data bucketing to reduce data padding. The number of data buckets is set to 6. We conduct experiments on 8 Nvidia V100 16GB GPUs and set gradient accumulation steps to 8. The global batch size is set to 256 in Experiment 1, and to 128 in Experiment 2. Since the models are large in Experiment 2, we use a small batch size to bypass out-of-memory issue. The training time for each task varies depending on the different training sets. For example, it takes about 38 hours to train the student with LCMR in Experiment 1. For conventional KD, similar to [12], we set the weight α that balances the RNN-T loss and distillation loss to $= 1e - 3$.

Table 4: *WER (%) of the student trained with LCMR using two training strategies on LibriSpeech in Experiment 1*

Method	dev	dev-other	test	test-other
Student+LCMR (Strategy 1)	18.34	35.68	18.16	36.89
Student+LCMR (Strategy 2)	**13.64**	**28.45**	**14.14**	**29.49**

Table 5: *WER (%) of the student trained with LCMR using different replacement schedulers in Experiment 1.*

Scheduler	dev	dev-other	test	test-other
Constant	14.91	30.12	15.48	30.97
Linear	14.82	30.14	15.31	30.98
Logarithmic	**13.64**	**28.45**	**14.14**	**29.49**
Exponential	14.99	30.35	15.49	30.97

5. Results and Discussions

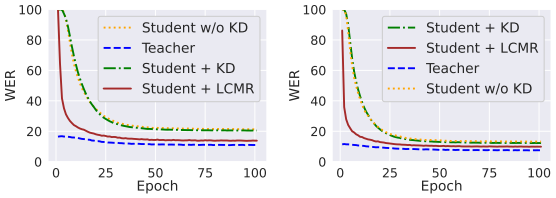
Table 2 and 3 present the Word Error Rate (WER) of the teacher and student in Experiment 1 and 2, respectively. For LCMR, the student is trained using the strategy where the pre-trained teacher is available. In all cases, LCMR significantly outperforms conventional KD by 14.47% to 33.24% relative. Compared to the student trained without KD, LCMR reduces WER by 20.50% to 37.55% relative. Most surprisingly, LCMR enables the small student achieving comparable performance to the large teacher with only 50% number of parameters. This confirms that LCMR is beneficial to on-device ASR applications which require high-performance small models on resource-constrained edge devices. Figure 3 illustrates the WER transitions for all the models in each epoch during training on dev dataset. As observed, the convergence speed of the student trained under LCMR is much faster than all the baselines.

The results confirm that LCMR outperforms KD in both converge speed and WER. LCMR enables the student and teacher to interact at the gradient level. In comparison, conventional KD lets the student mimic the teacher behaviour using only the lattice of its last layer, thus losing the information from intermediate layers and leading to insufficient learning.

The Effect of Training Strategies. We analyze the impact of two training strategies: 1) Training the teacher and student together (Strategy 1); or 2) Using a pre-trained teacher to train the student (Strategy 2). As shown in Figure 4(a) and Table 4, Strategy 2 achieves lower relative WER by 20.06% to 25.63% and a faster convergence speed than Strategy 1. Although Strategy 1 is worse than Strategy 2, it still outperforms conventional KD by 4.01% to 10.85% relative.

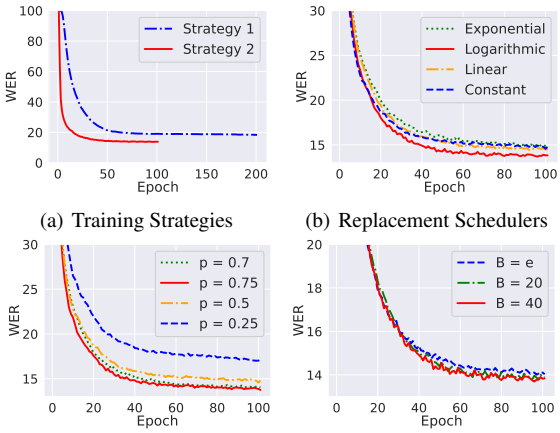
The Effect of Replacement Scheduler. Figure 4(b) and Table 5 show WER of the student trained using constant, linear, exponential and logarithmic replacement schedulers. Figure 2 illustrates the replacing curves of each scheduler. The proposed log-curriculum based replacement scheduler achieves the fastest convergence speed, and yields the lowest WER, outperforming the other three schedulers by 4.78% to 9.01% relative. The replacing rate p of constant replacement scheduler and the base value B of logarithmic replacement scheduler is set to 0.75 and 40.0, due to their best performance on the hyper-parameter search over $\{0.25, 0.5, 0.7, 0.75\}$ and $\{e, 20.0, 40.0\}$, respectively.

The Effect of Replacing Rate. Figure 4(c) shows the WER of constant replacement schedulers with different replacing rates $p = \{0.25, 0.5, 0.7, 0.75\}$ on dev dataset. A replacing rate above 0.5 achieves best possible performance. The performance drops if the replacing rate is too small (e.g., $p = 0.25$). These results are consistent with the performance of MR with the constant replacement strategy on BERT [10]. Figure 4(d) shows the WER of the logarithmic replacement scheduler with different base values $B = \{e, 20.0, 40.0\}$, which affect the replacing rate in each epoch. A larger base value (e.g., $B = 40.0$) results into a slightly better performance than a smaller one. It is, however, worth to note that there is little performance variance for the log-base values.



(a) Experiment 1 (b) Experiment 2

Figure 3: *WERs (%) for each epoch on dev dataset under Experiment 1 (left) and Experiment 2 (right).*



(a) Training Strategies (b) Replacement Schedulers

(c) Constant Replacement (d) Logarithmic Replacement

Figure 4: *WER (%) of (a) two training strategies, (b) four replacement schedulers, (c) constant replacement schedulers with different replacing rates, and (d) logarithmic linear replacement schedulers with different base values on dev dataset in Experiment 1.*

6. Conclusion

In this paper, we propose Log Curriculum Module Replacing (LCMR) based knowledge distillation (KD) for Automatic Speech Recognition (ASR) task with Recurrent Neural Network - Transducer (RNN-T). Different from conventional/lattice-based KD, which has several loss terms, large pre-trained teacher, and output distillation only, LCMR is hyperparameter-free and more importantly enables a gradient-level interaction between student and teacher modules/layers. It is observed that LCMR outperforms lattice-based KD by 14.47%-33.24%. Furthermore, we show that LCMR also yields better performance than constant or linear scheduler, achieving 4.78%-9.01% relative improvement. We hypothesize the reason to be the consonance between logarithmic replacement scheduler and exponential decay rate. Due to the space constraint, theoretical and empirical investigations on this interesting topic will be delegated to subsequent studies.

7. References

[1] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proceedings of the 2018 Workshop on Mobile Edge Communications*, 2018, pp. 31–36.

[2] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[3] M. T. Chu, R. E. Funderlic, and R. J. Plemmons, "Structured low rank approximation," *Linear algebra and its applications*, vol. 366, pp. 157–172, 2003.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[6] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[7] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[8] H. D. Nguyen, A. Alexandridis, and A. Mouchtaris, "Quantization aware training with absolute-cosine regularization for automatic speech recognition." in *Interspeech*, 2020, pp. 3366–3370.

[9] K. Zhen, H. D. Nguyen, F.-J. Chang, A. Mouchtaris, and A. Rastrow, "Sparsification via compressed sensing for automatic speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6009–6013.

[10] C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou, "Bert-of-theseus: Compressing bert by progressive module replacing," *arXiv preprint arXiv:2002.02925*, 2020.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[12] 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.

[13] G. Kurata and G. Saon, "Knowledge distillation from offline to streaming rnn transducer for end-to-end speech recognition." in *Interspeech*, 2020, pp. 2117–2121.

[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[15] 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 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6475–6479.

[16] S. Watanabe, T. Hori, J. Le Roux, and J. R. Hershey, "Student-teacher network learning with enhanced features," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5275–5279.

[17] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4820–4824.

[18] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers." in *Interspeech*, 2017, pp. 3697–3701.

[19] R. Takashima, L. Sheng, and H. Kawai, "Investigation of sequence-level knowledge distillation methods for ctc acoustic models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6156–6160.

[20] G. Kurata and K. Audhkhasi, "Guiding ctc posterior spike timings for improved posterior fusion and knowledge distillation," *arXiv preprint arXiv:1904.08311*, 2019.

[21] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

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

[23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.