INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

A Metric-Driven Approach to Conformer Layer Pruning for Efficient ASR Inference

Dhanush Bekal Karthik Gopalakrishnan Karel Mundnich Srikanth Ronanki
Sravan Bodapati Katrin Kirchhoff

AWS AI Labs, USA
{dkannang, karthgop, kmundnic, ronanks, sravanb, katrinki}@amazon.com

Abstract

Conformer-based end-to-end automatic speech recognition (ASR) models have gained popularity in recent years due to their exceptional performance at scale. However, there are significant computation, memory and latency costs associated with running inference on such models. With the aim of mitigating these issues, we evaluate the efficacy of pruning Conformer layers while fine-tuning only on 20% of the data used for the pre-trained model. We score Conformer layers using correlation, energy, and gradient-based metrics and rank them to identify candidate layers for pruning. We also propose an iterative pruning strategy which monitors and prunes layers that are consistently ranked low by the metrics during training. Using our methods, we prune large pre-trained offline and online (streaming) models by 20% and 40% with little impact on performance, while outperforming a strong knowledge distillation baseline.
Index Terms: Speech recognition, Conformer model, one-shot pruning, iterative pruning, efficient inference.

1. Introduction

End-to-end (E2E) ASR systems are increasingly supplanting conventional hybrid models owing to their superior performance. Various encoder architectures and decoding techniques have been proposed in the literature to train E2E models, with the most popular encoder architectures being LSTMs, Transformer [1] and Conformer [2] and decoding strategies being Connectionist Temporal Classification (CTC) [3, 4], Transducer [5], and Attention decoder [6, 7] or a combination of the previous three. Due to their exceptional performance, Conformer encoders [2] combined with either CTC or Transducer are ubiquitously used to train E2E-ASR models.

With the availability of thousands of hours of speech data and efficient training strategies, comes the ease and necessity to train larger models with higher capacity for improved performance. It has become increasingly prevalent to utilize ASR models that feature large encoders with more than 150 million parameters. The recently released Whisper model [8] was trained on 680,000 hours of data and the large version of the model has 1.5 Billion parameters. Such large-scale models have large computation and memory costs, to use them at inference time. This motivates the use of techniques like Knowledge Distillation (KD) and pruning to reduce model size and thus reduce costs while either maintaining performance or have minimal degradation compared to the full model. Knowledge Distillation has been well explored in literature [9, 10] and is also well studied and used in ASR to produce lightweight student models [11, 12, 13, 14].

Pruning is another popular technique used in vision and NLP to reduce model size [15] but fairly new in ASR [16, 17, 18, 19]. A node pruning method based on node and weight entropy is introduced in [20]. A scoring function is used to remove a significant number of nodes resulting in high real-time factor (RTF) gains and without much loss in word accuracy. [21] applies the Lottery Ticket Hypothesis for structured pruning and finding efficient sparse models across multiple architectures like CNN-LSTM, RNN-Transducer, and Transformer models. They show that the sparse light-weight models obtained from large models generalize well with minimal to no loss in performance. [22] introduced on-demand layer pruning where entire Conformer layers are removed using an iterative search method augmented with intermediate CTC. Despite some success, the main limitation of aforementioned pruning approaches in speech recognition is that the training is limited to less than 1000 hours. Typically, the models trained on such small amount of data do not require deep encoders to provide high performance. At the same time, the performance of the models with varying depth differs quite significantly when trained on large-scale labeled data [8].

In this work, we investigate the impact of pruning a 150M parameter model that was pre-trained on 50,000+ hours of labeled data. The overall pipeline involves three steps: training a large pre-trained model, pruning the model and fine-tuning the resulting small model. Specifically, we focus on pruning Conformer layers and reduce the depth of the encoder network during fine-tuning without much loss in accuracy. We use a variety of metrics for scoring, ranking the importance of encoder layers and showcase them as an effective way of pruning for ASR. We also present a strategy for iterative pruning of encoder layers over multiple epochs during training. We show that our methods help approach near pre-trained model performance while also achieving better performance than KD. Our main contributions are: (1) For Conformer-based E2E models, we demonstrate our layer pruning approach as a viable alternative to training a student model using Knowledge Distillation; (2) We present multiple metrics to rank the importance of a Conformer layer for pruning; and (3) We explore an iterative pruning approach to reduce model depth over multiple fine-tuning epochs.

2. Model Architecture

The models are trained using a joint CTC-Attention architecture [23, 24], which can be broken down into three main components: *Shared Encoder*, *CTC Decoder*, and *Attention Decoder*. During inference, only the shared Conformer encoder and CTC decoder are used to ensure a low inference latency.

2.1. Conformer Encoder

The Conformer Encoder is composed of multiple Conformer layers stacked on top of each other. Each Conformer layer is a

10.21437/Interspeech.2023-2183

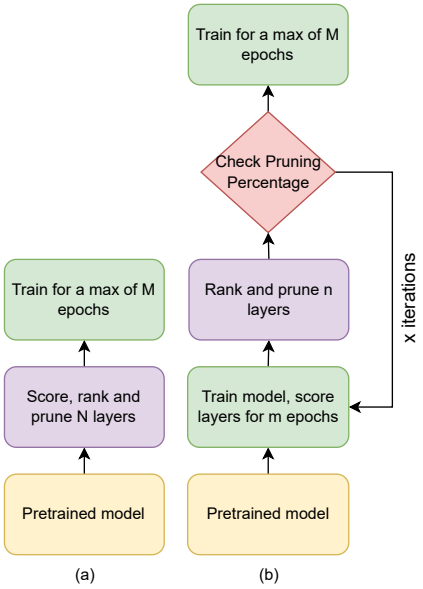

Figure 1: *Schematic diagram of our proposed pruning approach. Figure (a) shows one-shot pruning and fine-tuning and figure (b) shows iterative pruning strategy.*

modification of the Transformer layer which adds convolution modules along with self-attention modules. The self-attention learns global interactions whilst the convolutions efficiently capture the relative-offset-based local correlations, achieving the best of both worlds and leading to improvements in ASR performance. A standard Conformer layer contains two feed forward modules (FFN) sandwiching the multi-headed self-attention (MHSA) module and the convolution module. L conformer layers stacked on top of each other form the Conformer encoder [2]. The output from the final layer (X^L) of the shared encoder is fed into a shallow transformer decoder and the CTC decoder.

2.2. Connectionist Temporal Classification

CTC helps to solve the sequence prediction problem using conditionally independent monotonic alignments. It aligns T-length encoder output sequence $\boldsymbol{x} \in \mathbb{R}^{T\times D}$ and K-length target symbol sequence $y = \{y_1, y_2, ..., y_K \in V\}$, where D is the dimension of $\boldsymbol{x}$ and V is the vocabulary. Using an additional blank symbol b and defining the alignment as $a = \{a_1, a_2, ..., a_T \in V \cup \{b\}\}$, CTC computes the likelihood of the target sequence y as:

$$\mathcal{P}_{\text{CTC}}(y|\boldsymbol{x}) = \sum_{a\in\beta^{-1}(y)} \mathcal{P}(a|\boldsymbol{x}), \tag{1}$$

where $P(a|\boldsymbol{x})$ denotes the alignment probability and $\beta^{-1}(y)$ is the set of possible alignments that are compatible to y. During inference, the most probable alignment is found by either greedy or beam search decoding, allowing fast and non-autoregressive inference.

3. Model Pruning

In this section, we define four different metrics that can be used to score the importance of a Conformer layer. The metric scores are then used to rank Conformer layers and the least ranked layers are pruned from the encoder.

The notation for this section is as follows: let $\boldsymbol{X} \in \mathbb{R}^{B\times D\times T}$ be the input tensor to a layer, where B is the batch size, D is the size of the feature dimension, and T is the length in time. We use the notation $\boldsymbol{X}_{b..}$ to indicate the matrix of size $D \times T$, which is a slice of the tensor $\boldsymbol{X}$, $b \in \{\mathbf{1}, \ldots, B\}$. We define the metrics next.

3.1. Metrics

Correlation The correlation between the inputs and outputs of a layer hints at the amount of change induced by it. Early layers induce a lot of change as they convert input features into hidden representations. A high correlation score of a layer indicates that it is unimportant. The correlation metric is defined as follows:

$$\mathcal{S}_\rho = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{TD}\sum_j \text{diag}\left(\text{vec}(\boldsymbol{X}^i_{b..})\text{vec}(\boldsymbol{X}^{i-1}_{b..})^T\right)_j \tag{2}$$

where $\text{vec}(\boldsymbol{X}_{b..})$ is the vectorized form of the matrix $\boldsymbol{X}_{b..}$, B is the batch size, the superscript $\boldsymbol{X}^{i-1}$ denotes the input to a layer, $\boldsymbol{X}^i$ denotes the output, and $\text{diag}(\cdot)$ is the diagonal of the matrix resulting from the outer-product of the two vectors.

Energy In the energy-based metric [25], we rank the importance of Conformer layers based on the energy of the output of that layer. The higher the energy of the output, the more important the Conformer layer to maintain performance. The energy metric is calculated as follows:

$$\mathcal{S}_E = \frac{1}{DT}\sum_{b=1}^{B}\sum_{d=1}^{D}\left|\lambda_d - \text{tr}(\boldsymbol{X}_{b..}\boldsymbol{X}^T_{b..})\right| \tag{3}$$

where $\text{tr}(\cdot)$ is the trace of $\boldsymbol{X}_{b..}\boldsymbol{X}^T_{b..}$, and λ_d for $d \in \{1, \ldots, D\}$ are the eigenvalues of $\boldsymbol{X}_{b..}\boldsymbol{X}^T_{b..}$.

Gradient/Hessian The previous two metrics are calculated during the forward call of the model. We also explore ranking of a Conformer layer by computing its importance during a backward call. The absolute gradient flowing through a layer hints that a layer still has the capacity to learn more from the training data. Layers are ranked according to the net first-order (gradient) and second-order (Hessian) derivatives of their corresponding parameters. The higher the absolute derivatives, the higher the rank of a Conformer layer. Scores are computed as follows (we show the first-order case only for conciseness):

$$\mathcal{S} = \frac{1}{B}\sum_{b=1}^{B}\text{vec}(\boldsymbol{X}^{i-1}_{b..})\cdot\nabla^k_{\boldsymbol{X}^i}(\text{vec}(\boldsymbol{X}^i_{b..})), \tag{4}$$

where $\boldsymbol{X}^{i-1}$ denotes the input, and $\boldsymbol{X}^i$ denotes the output of a layer. We use torch autograd functions to compute the gradients of the layers. For $k = 1$, $S = \mathcal{S}_G$ and for $k = 2$, $S = \mathcal{S}_H$.

3.2. Conformer Layer Pruning

As shown in Figure 1, we present two approaches for pruning a Conformer encoder:

One-Shot Pruning In this approach, the Conformer layers of a pre-trained model are scored and ranked using one of the previously defined metrics. The lowest ranked layers are removed in one-shot as determined by the pruning percentage and the resulting smaller Conformer model is fine-tuned for M epochs.

Iterative Pruning In this approach, we prune the Conformer layers of a pre-trained model in an iterative fashion over multiple epochs. As shown in Figure 1 (b), we train for m epochs,

Table 1: *Comparison of WER for various scoring metrics trained in offline mode using the one-shot pruning approach. CV: Mozilla Common Voice, LS: Librispeech*

Model	Method	Datasets						
		WSJ *test92*	Vox	SLURP *Test*	CV	LS (test) *clean*	LS (test) *other*	DSTC-2
20-layer	-	4.6	9.1	19.4	9.2	3.5	7.0	7.9
16-layer	Energy	4.9	9.5	20.0	10.3	4.2	8.4	7.4
	Correlation	4.8	**9.1**	**19.7**	**9.4**	**4.1**	**8.3**	**7.0**
	Gradient	**4.7**	9.3	20.3	9.8	4.3	8.4	7.5
	Hessian	4.9	10.3	25.6	10.5	4.5	9.0	8.1
12-layer	Energy	4.8	9.8	20.5	10.2	4.7	9.2	**7.6**
	Correlation	**4.7**	**9.8**	**20.3**	**10.1**	4.6	**8.9**	7.9
	Gradient	4.9	9.6	21.9	10.6	**4.5**	9.1	7.9
	Hessian	5.0	9.9	22.2	13.0	5.0	10.4	8.5

score and prune n layers at the m^{th} epoch, training the resulting model for another m epochs and so on until the desired pruning percentage is reached. Further, we can prune the layers by checking for consistency of the ranks assigned to Conformer layers, *i.e.*, we can prune layers consistently ranked low over m epochs. To measure consistency of the rank of a Conformer layer, we measure its importance using our metrics at each training epoch and then use average rank during the pruning epoch to decide which layer is to be pruned. The consistency equation is defined as follows:

$$C_\mu = \frac{1}{m}\sum_{e=1}^{m}(\mathcal{S}_\mu^e), \tag{5}$$

where C_μ is the consistency rank of a layer and m is the total number of epochs over which the metric score of a layer is measured, and $\mathcal{S}_\mu^e$ is one of the scores for the metrics defined above for epoch e, so that $\mu \in \mathcal{M} = \{\rho, E, G, H\}$ (correlation, energy, gradient, or Hessian).

4. Experimental Setup

4.1. Data

Train We use two different-sized corpora: a large 50K+ hour English corpus and a 10K hour subset of this corpus, sampled from in-house paired audio and text data. The datasets are a good mix of 8kHz (upsampled during training) and 16kHz sampling rate and include a number of different accents, speakers, and background noise. We pre-train the model on the 50K+ hour corpus and the rest of pruning and distillation experiments are conducted on the 10K hour corpus (20% of overall data).

Evaluation We use six datasets for evaluation: (1) *WSJ/test_eval92 (test92)*, from which we use test92 [26], prepared using Kaldi's [27] WSJ recipe, which is 0.7hrs long; (2) *Voxpopuli* [28], from which we use the English test partition, which is 4.9hrs long; (3) *SLURP*: we use the test partition of this dataset [29] containing 2974 sentences totaling 10.3 hours; (4) *Common Voice*: we use version 5.1 of Common Voice data [30], with 16028 test utterances totalling 25.9 hrs; (5) *Librispeech*: we use the test-clean and test-other splits of Librispeech data [31]; (6) *DSTC-2*: we use the test split of the DSTC-2 dataset [32]. We report word error rate (WER) on all six datasets.

4.2. Models

To test our layer importance scoring metrics and iterative pruning approach, we pre-train a 20-layer Conformer model (150M parameters) using a joint CTC-Attention architecture described in Section 2. This model is used to initialize the parameters of the models to be pruned and trained. This model also serves as the teacher model to train student models.

Table 2: *Comparison of one-shot pruning approach with pre-trained and distilled models. The pruning models used for comparison are trained in offline mode using correlation scoring metric.*

Model	Method	Datasets						
		WSJ *test92*	Vox	SLURP *Test*	CV	LS (test) *clean*	LS (test) *other*	DSTC-2
20-layer	Trained	4.6	9.1	19.4	9.2	3.5	7.0	7.9
16-layer	Trained	5.2	10.2	21.6	11.0	4.7	9.0	7.8
	Distillation	5.3	9.7	20.9	10.1	4.5	8.6	7.4
	Pruning	**4.8**	**9.1**	**19.7**	**9.4**	**4.1**	**8.3**	**7.0**
12-layer	Trained	4.9	10.3	22.5	12.1	5.0	9.6	8.4
	Distillation	5.0	10.3	21.6	11.1	4.8	9.2	**7.8**
	Pruning	**4.7**	**9.8**	**20.3**	**10.1**	**4.6**	**8.9**	7.9

Offline Models All our Conformer models are trained with the CTC and inter-CTC objectives. The pre-trained model has a 20×1 architecture (encoder×decoder depths). We use a Macaron-style Conformer with layers having 512 hidden units, 8 attention heads and 31 convolutional filters. We use a vocabulary size of 2048 to train our models. During inference, we use full left and right context.

Online Models Similar to [33], we train a unified streaming and non-streaming model with Conformer architecture (as above), trained with dynamic chunk training (DCT) implemented with dynamic chunk attention masks for the self-attention matrices. For the convolution modules, we use dynamic chunk convolution (DCConv) instead of causal convolution which makes better use of the in-chunk future context and significantly improves streaming performance while maintaining non-streaming performance. We run inference using a chunk size of 64, with a hop size of 32 frames and 128 frames of left context.

4.3. Knowledge Distillation and Pruning

We train two student models (16-layer and 12-layer) with the pre-trained teacher model trained on 50K+ hours of data. We use a simple frame-based knowledge distillation approach, with the objective function of reducing the cosine distance between the logits of the final layer between teacher and student models (as in [34] but without the teacher classifier). The student models are initialized with a pre-trained model of same size, and trained with 10K hour dataset. For pruning, we use the pre-trained model to initialize the parameters of the model to be pruned and apply our scoring metrics on this pre-initialized model. We showcase our experiments on 20% and 40% pruning, *i.e.*, 16-layer and 12-layer pruned models. The metric scores are calculated using a validation set. We compare all four metric-based approaches and choose the best metric to conduct the iterative pruning experiments. For each training mode, we evaluate two models with number of iterations set to 2 and 4. We choose the epoch interval for iterative pruning based on the number of layers to be pruned, thus allowing the model sufficient time to learn before each iteration of pruning.

4.4. Training and Decoding

We perform all of our experiments in ESPnet [35] using AWS EC2 P4d instances. All our models are trained on 16kHz audio, with SpecAugment [36] applied over 80-dimensional extracted filter-bank features. We pre-train a 2048-token sub-word model from our training data using SentencePiece tokenizer [37]. We use Adam [38] with a learning rate of 5e-4, and a warm-up scheduler [1] with 50k steps, and train all models for 30 epochs. During inference, we use beam search CTC decoding with a 4-

Table 3: *Iterative pruning for offline and online models. Layers are pruned based on their input/output correlation. The "Pruning parameters" columns indicate the number of layers pruned at the corresponding epoch interval (EI) for each iteration before training continues. Bold numbers show improvements of more than 0.1 absolute WER over the one-shot baseline.*

Mode	Model	Method	Pruning parameters		Datasets						
			# Iterations	(layers, EI)	WSJ *test92*	Voxpopuli	SLURP *Test*	Common Voice	Librispeech (test) *clean*	Librispeech (test) *other*	DSTC-2
Offline	20-layer	Trained	–	–	4.6	9.1	19.4	9.2	3.5	7.0	7.9
	12-layer (pruned)	One-Shot	–	–	4.7	9.8	20.3	10.1	4.6	8.9	7.9
		Iterative	2	(4, 4)	4.8	9.7	20.3	10.0	4.4	8.9	**7.4**
		Iterative	4	(2, 2)	4.7	9.7	20.2	10.0	4.5	8.8	**7.4**
Online	20-layer	Trained	–	–	5.6	10.5	23.6	13.2	5.5	11.5	15.4
	12-layer (pruned)	One-Shot	–	–	6.3	11.9	26.7	16.3	6.4	13.1	12.3
		Iterative	2	(4, 4)	6.3	11.9	26.6	16.2	6.4	13.0	**12.1**
		Iterative	4	(2, 2)	6.5	11.8	26.7	**15.9**	6.3	**12.8**	**11.8**

gram language model (LM) trained on the transcriptions of the 50K+ hour corpus. The beam size is set to 100 and the LM weight is set to 0.6.

5. Results

5.1. Comparison of Metrics for One-Shot Pruning

In Table 1, we compare the performance of our scoring metrics to fine-tune both 20% and 40% pruned models. From the table, we see that the correlation metric performs the best and finds the most suitable Conformer layers to be removed to reduce model depth. Compared with the from-scratch trained 20% and 40% models shown in Table 2, we see that all our pruning metrics result in a better performing model.

Figure 2 presents the correlation-based importance scores for Conformer layers without any pruning. We observe a noticeable difference in importance scores and the ranking of Conformer layers over the course of training. We observe that early layers are consistently important, possibly to learn the input audio feature transformations. After a certain number of epochs, some layers lose importance possibly due to saturation in information gained from the data over epochs.

5.2. Comparison of Pruning and Knowledge Distillation

Table 2 shows results for pre-trained 20-layer model as well as a comparison between smaller models trained using different techniques. As expected, the smaller models perform worse than the pre-trained model on almost all datasets. We see that for both the 20% (16-layer) and 40% (12-layer) models, the pruned models perform best. When comparing against KD, our one-shot pruning approach achieves a relative WER (WERR) improvement of 6.6% and 4.6% averaged across all datasets. Our pruning approach best closes the gap with the 20-layer model having a minimum 1.5% WERR degradation on SLURP test set and a maximum WERR degradation of 18.5% on Librispeech test-other dataset. Surprisingly, we see that our pruned model performs better than the pre-trained model on DSTC-2.

5.3. Iterative Pruning in Offline and Online Models

We compare our iterative pruning approach by computing the scoring metrics and pruning the model layers across multiple epochs using the technique mentioned in Section 4.3.

Offline Experiments From Table 3, we observe that the iterative pruning approach achieves similar performance to the one-shot pruning in the offline mode. We also observe that iterative pruning shows improvement with some test sets when using a higher number of pruning iterations. This improvement is also more prevalent in the online mode. We conjecture that iterative pruning would likely outperform one-shot pruning for models with much higher capacity that are intended to be adapted to unseen target acoustic conditions.

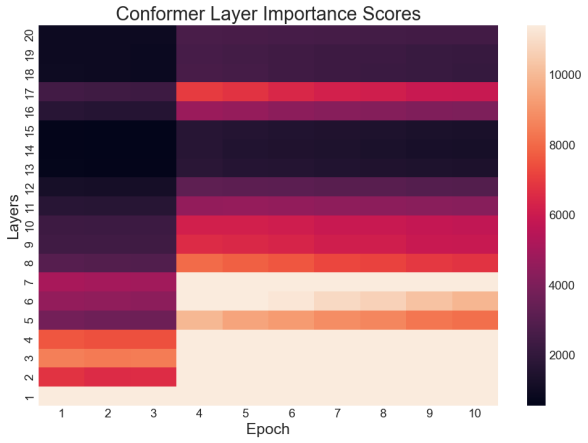

Figure 2: *Correlation-based importance scores for Conformer layers over the course of 10 epochs of training.*

Online Experiments We present the online inference results in Table 3, where we observe that the degradation of performance is much larger than that of offline models (and running offline inference). We hypothesize that intermediate layers are more important for online inference, given the limited context that these models have access to. Moreover, these models are fine-tuned with less data (10K hrs) than the teacher model (over 50K hrs), which we observe makes a difference in the performance of the unified streaming models.

6. Conclusion

In this work, we demonstrate scoring-metric-based Conformer layer pruning as a viable alternative to distillation for training smaller models. Our experiments show that correlation metric-based pruning leads to the best-performing smaller models. We also proposed an iterative pruning technique to reduce the depth of the model across multiple epochs. Our findings indicate that increasing the number of pruning iterations results in more effectively pruned models. Future work could focus on pruning individual components of the Conformer layers (either convolutions or attention heads) and explore threshold-based approaches for automated iterative pruning.

7. References

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

[2] A. Gulati, J. Qin, 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 *Proc. Interspeech*, 2020, pp. 5036–5040.

[3] A. Graves, A. rahman Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[4] S. Dingliwal, M. Sunkara, S. Ronanki, J. Farris, K. Kirchhoff, and S. Bodapati, "Personalization of ctc speech recognition models," in *Proc. Spoken Language Technology Workshop (SLT)*, 2023, pp. 302–309.

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

[6] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[7] L. Lu, C. Liu, J. Li, and Y. Gong, "Exploring transformers for large-scale speech recognition," in *Proc. Interspeech*, 2020, pp. 5041–5045.

[8] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[9] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[10] F. Ruffy and K. Chahal, "The state of knowledge distillation for classification," *arXiv preprint arXiv:1912.10850*, 2019.

[11] T. Moriya, H. Sato, T. Tanaka, T. Ashihara, R. Masumura, and Y. Shinohara, "Distilling attention weights for ctc-based asr systems," in *Proc. ICASSP*, 2020, pp. 6894–6898.

[12] 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 *Proc. ICASSP*, 2021, pp. 5879–5883.

[13] 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 *Proc. Interspeech*, 2020, pp. 546–550.

[14] H. Futami, H. Inaguma, M. Mimura, S. Sakai, and T. Kawahara, "Distilling the knowledge of bert for ctc-based asr," *arXiv preprint arXiv:2209.02030*, 2022.

[15] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[16] G. Mantena and K. C. Sim, "Entropy-based pruning of hidden units to reduce DNN parameters," in *Proc. Spoken Language Technology Workshop (SLT)*, 2016, pp. 672–679.

[17] C. Liu, Y. Shangguan, H. Yang, Y. Shi, R. Krishnamoorthi, and O. Kalinli, "Learning a dual-mode speech recognition model VIA self-pruning," in *Proc. Spoken Language Technology Workshop (SLT)*, 2022, pp. 273–279.

[18] C. J. Lai, Y. Zhang, A. H. Liu, S. Chang, Y. Liao, Y. Chuang, K. Qian, S. Khurana, D. D. Cox, and J. Glass, "PARP: prune, adjust and re-prune for self-supervised speech recognition," in *Proc. NeurIPS*, 2021, pp. 21 256–21 272.

[19] V. S. Lodagala, S. Ghosh, and S. Umesh, "PADA: pruning assisted domain adaptation for self-supervised speech representations," in *Proc. Spoken Language Technology Workshop (SLT)*, 2022, pp. 136–143.

[20] R. Takeda, K. Nakadai, and K. Komatani, "Node pruning based on entropy of weights and node activity for small-footprint acoustic model based on deep neural networks." in *Proc. Interspeech*, 2017, pp. 1636–1640.

[21] S. Ding, T. Chen, and Z. Wang, "Audio lottery: Speech recognition made ultra-lightweight, noise-robust, and transferable," in *International Conference on Learning Representations (ICLR)*, 2022.

[22] J. Lee, J. Kang, and S. Watanabe, "Layer pruning on demand with intermediate CTC," in *Proc. Interspeech*, 2021, pp. 3745–3749.

[23] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[24] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[25] Christi DiStefano and Gary Davis, "Matrix Energy," University of Massachusetts at Dartmouth, Tech. Rep., 2009. [Online]. Available: https://compmath.files.wordpress.com/2009/12/cdistefano_freportf09.pdf

[26] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "CSR-I (WSJ0) Complete LDC93S6A," *Web Download. Philadelphia: Linguistic Data Consortium*, vol. 83, 1993.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, 2011.

[28] C. Wang, M. Rivière, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. M. Pino, and E. Dupoux, "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *Proc. ACL/IJCNLP*, 2021, pp. 993–1003.

[29] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "SLURP: A spoken language understanding resource package," in *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 7252–7262.

[30] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proc. Language Resources and Evaluation Conference, (LREC)*, 2020, pp. 4218–4222.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[32] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proc. Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 263–272.

[33] X. Li, G. Huybrechts, S. Ronanki, J. Farris, and S. Bodapati, "Dynamic chunk convolution for unified streaming and non-streaming conformer asr," in *Proc. ICASSP*, 2023, pp. 1–5.

[34] S. Tian, K. Deng, Z. Li, L. Ye, G. Cheng, T. Li, and Y. Yan, "Knowledge Distillation For CTC-based Speech Recognition Via Consistent Acoustic Representation Learning," in *Proc. Interspeech*, 2022, pp. 2633–2637.

[35] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, 2018, pp. 2207–2211.

[36] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[37] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 66–71.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.