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

Layer Pruning on Demand with Intermediate CTC

Jaesong Lee[1], *Jingu Kang*[1], *Shinji Watanabe*[2]

[1]Naver Corporation, Korea
[2]Carnegie Mellon University, USA

jaesong.lee@navercorp.com, kang.jingu@navercorp.com, shinjiw@ieee.org

Abstract

Deploying an end-to-end automatic speech recognition (ASR) model on mobile/embedded devices is a challenging task, since the device computational power and energy consumption requirements are dynamically changed in practice. To overcome the issue, we present a training and pruning method for ASR based on the connectionist temporal classification (CTC) which allows reduction of model depth at run-time without any extra fine-tuning. To achieve the goal, we adopt two regularization methods, intermediate CTC and stochastic depth, to train a model whose performance does not degrade much after pruning. We present an in-depth analysis of layer behaviors using singular vector canonical correlation analysis (SVCCA), and efficient strategies for finding layers which are safe to prune. Using the proposed method, we show that a Transformer-CTC model can be pruned in various depth on demand, improving real-time factor from 0.005 to 0.002 on GPU, while each pruned sub-model maintains the accuracy of individually trained model of the same depth.

Index Terms: end-to-end automatic speech recognition, connectionist temporal classification, pruning, non-autoregressive

1. Introduction

End-to-end automatic speech recognition (ASR) has recently become a popular approach. It gives strong performance and simple model design compared to traditional systems like hidden Markov model. Also, with the advances of hardware [1] and software [2], it became possible to deploy the end-to-end ASR system on the mobile devices like smartphones and embedded devices. However, developing and deploying an end-to-end ASR model on such devices is still challenging, because of the varying computational power of the devices [1]. Recent high-end devices have very fast CPU and large memory and some of them also have dedicated hardware like neural processor unit (NPU). On the other hand, low-end and old-generation devices with limited computational power are also widely used. This gives a critical problem for on-device ASR design. A large model gives high recognition accuracy but requires significant computational cost. Without the dedicated processor, the device consumes significant energy and time, affecting battery life [1], thus small and fast models should be needed for such devices. This requires designing several models with different size, complicating the development and deployment of on-device ASR.

Recently, the idea of *on-demand pruning* has been popularized [3, 4, 5, 6]. Conventional pruning methods require three steps: 1) training large model, 2) pruning the model, and 3) fine-tuning the resulting small model. On the other hand, pruning on demand requires *no fine-tuning* after pruning. This property is especially helpful for on-device ASR design. Only one model is required for development and deployment, and each device may adjust the level of computation given its own limit. In this end, we consider the problem of on-demand pruning for ASR modeling.

Connectionist temporal classification (CTC) [7] has been a widely used method for end-to-end ASR modeling [8, 9, 10, 11, 12], and it is especially an attractive method for the on-device ASR. For low-end devices, CTC is suitable for lightweight modeling. Compared to attentional encoder-decoder architectures [13, 14, 15, 16] or RNN-Transducer [17], CTC-based models do not require separate decoder networks, thus they require less computational cost (time and memory) both for training and inference. Also, for high-end devices equipped with dedicated processors, CTC with modern architectures (e.g., Transformer [18, 19] and Conformer [20, 21]) allows fast parallel inference, coming from the *non-autoregressive* property of CTC. Also, while CTC has been regarded weaker than encoder-decoder, various ways to improve CTC, including pretraining [12] and regularization [22, 23], have been developed. Also, there are active researches on CTC variants [12] and on non-autoregressive modeling based on CTC [24, 25, 26], suggesting that effective pruning for CTC can also be applied to other variants or non-autoregressive models.

In this work, we consider the on-demand layer pruning problem: training a deep neural network and removing some of the layers, thus reducing the depth of the network, without any fine-tuning. To achieve this, we employ two methods, intermediate CTC [22] and stochastic depth [27, 28, 5]. Although they are originally introduced as regularizers during training, we show they can be applied to layer pruning method as well. Note that we also explore in-place distillation [4], an application of knowledge distillation [29, 30, 31, 32, 33] to on-demand pruning, but find degraded result in our ablation study.

Our main contributions are:

- We employ singular vector canonical correlation analysis (SVCCA) to analyze intermediate CTC and stochastic depth in context of layer pruning.
- We show that intermediate CTC, along with stochastic depth, not only improves training of CTC models, but also provide a natural way to prune layers.
- We develop an iterative search method augmented with intermediate CTC for pruning layers.
- We present experimental results that the proposed model and its induced sub-models match the accuracy of individually trained models with same depth, without any fine-tuning, thus proving the effectiveness of our method.

2. Architecture

2.1. Connectionist Temporal Classification

Connectionist temporal classification (CTC) solves the sequence prediction problem by introducing a monotonic alignment. For the encoder output $x \in \mathbb{R}^{T \times D}$ of length T and fea-

http://dx.doi.org/10.21437/Interspeech.2021-1171

ture dimension D, the CTC layer computes the likelihood of target sequence y:

$$P(y|x) := \sum_{a \in \beta^{-1}(y)} P(a|x), \tag{1}$$

where $\beta^{-1}(y)$ is the set of possible alignments that are compatible to y and are of length T, and a is an alignment in the set. The probability of an alignment is modeled as a factorized distribution:

$$P(a|x) := \prod_t P(a[t]|x[t]) \tag{2}$$

$$P(a[t]|x[t]) := \mathrm{Softmax}(\mathrm{Linear}(x[t]))_{a[t]} \tag{3}$$

where $a[t]$ and $x[t]$ mean the t-th value of a and x respectively.

During inference, the most probable alignment is found by greedy decoding, allowing fast and non-autoregressive inference.

2.2. Transformer encoder

Transformer [18] is a multi-layer architecture with self-attention and residual connection [34]. Transformer layer uses self-attention to learn global information and residual connection to help training of a deep neural network.

We use the encoder part of the Transformer architecture. Let L be the number of layers in the encoder. The l-th layer computes the representation x_l for given input x_{l-1} by:

$$x_l^{\mathrm{MHA}} = \mathrm{SelfAttention}(x_{l-1}) + x_{l-1} \tag{4}$$

$$x_l = \mathrm{FeedForward}(x_l^{\mathrm{MHA}}) + x_l^{\mathrm{MHA}}, \tag{5}$$

where x_0 indicates the input of the Transformer encoder.

The final representation x_L is then fed to CTC layer to minimize the following loss:

$$\mathcal{L}_{\mathsf{CTC}} := -\log P_{\mathsf{CTC}}(y|x_L). \tag{6}$$

2.3. Stochastic depth and LayerDrop

Stochastic depth [27, 28] is a regularization method designed for deep residual networks [34]. During training, each layer is randomly skipped or not with a given probability p. For each iteration, u is sampled from a Bernoulli distribution so that the probability of $u = 1$ is p and the probability of $u = 0$ is $1 - p$. The output is computed by modifying Eqs. (4) and (5) as:

$$x_l^{\mathrm{MHA}} = \frac{u}{p} \cdot \mathrm{SelfAttention}(x_{l-1}) + x_{l-1} \tag{7}$$

$$x_l = \frac{u}{p} \cdot \mathrm{FeedForward}(x_l^{\mathrm{MHA}}) + x_l^{\mathrm{MHA}}. \tag{8}$$

If $u = 0$, the residual parts are skipped (i.e., $x_l = x_{l-1}$).

LayerDrop [5] is an application of stochastic depth to layer pruning. After training the model with stochastic depth, removing some layers from the model gives new smaller sub-model which also has reasonable performance without any fine-tuning.

2.4. Intermediate CTC

Intermediate CTC [22] is an auxiliary loss designed for CTC modeling. It regularizes the model using an additional CTC loss attached at the intermediate layer of the encoder.

Let $l_1, \ldots, l_K$ be the K positions ($K < L$) of the intermediate layers. The intermediate loss is defined as:

$$\mathcal{L}_{\mathsf{InterCTC}} := \frac{1}{K} \sum_k -\log P_{\mathsf{CTC}}(y|x_{l_k}). \tag{9}$$

Then, the training objective is defined by combining Eqs. (6) and (9):

$$\mathcal{L} := (1 - w)\mathcal{L}_{\mathsf{CTC}} + w\mathcal{L}_{\mathsf{InterCTC}} \tag{10}$$

with a hyper-parameter w.

Intermediate CTC has very small overhead during training, because the intermediate representation x_{l_k} is naturally obtained when the final output x_L is computed. The only additional cost is to compute CTC loss for the given representation, which is much smaller than the cost of the encoder. [22] explores various choices for the intermediate layer positions, and concludes that it is sufficient to use one layer ($K = 1$) in the middle ($l_1 = \lfloor L/2 \rfloor$) for the regularization purpose. We revisit the effect of variants at Section 3.

Note that CTC and intermediate CTC share the same linear projection layer of Eq. (3), as intermediate CTC is treated as a regular CTC loss except that all of the encoder layers after the intermediate layer are skipped. This property is used at Section 4.1.

3. Layer similarity analysis

Intuitively, if a layer of the model can be pruned without much degradation of accuracy, it would mean the next layer behaves similarly regardless that the layer is removed or not. Therefore, measuring "similarity" of layer behavior would give a good intuition for layer pruning. We measure the layer similarity using singular vector canonical correlation analysis (SVCCA) [35].

SVCCA accepts two matrices and finds two linear projections which maximize correlation between the projected matrices. Then it averages the correlation coefficients into a scalar value. The result, mean SVCCA similarity, indicates the similarity of two matrices up to linear transformation. The similarity of two layers can be measured by computing the mean SVCCA similarity of the layer outputs.

We first collect outputs of all layers from each input of the validation set. For each layer, the outputs are concatenated into a single matrix. Then, for each pair of layers, we compute the mean SVCCA similarity of the corresponding matrices. We also collect the inputs of the first layer and compute the similarity between it and the other layers as well. The input is denoted as "0th layer" in the following figures.

3.1. Effect of intermediate CTC and stochastic depth

We first investigate how the two regularization techniques, stochastic depth and intermediate CTC, affect layer similarity. We train a 24-layer Transformer CTC model with four different configurations: (a) baseline, (b) with intermediate CTC ($w = 0.3$, at 12th layer), (c) with stochastic depth, and (d) with both. Figure 1 shows the similarity matrices of the four models.

From Figures 1 (a) and (b), we observe that intermediate CTC increases similarity of layers between the intermediate one (12th) and the final one (24th), at the cost of decreasing similarity of the first layers a bit. Especially, the intermediate layer is significantly similar to last layer and its nearby layers.

On the other hand, Figure 1 (c) shows that stochastic depth evenly increases the similarity of the last half of layers. Especially we see the neighboring layers have very high similarities. This can be explained by that the layer should behave similarly regardless of whether its previous layer has been skipped or not.

Finally, Figure 1 (d) shows the combination of two regularizations greatly increases similarity among all layers between the intermediate layer and the last layer. This gives us an insight that the both regularizations would help pruning the layers.

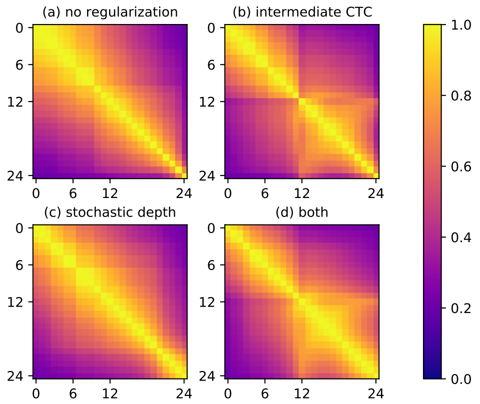

Figure 1: *SVCCA similarity matrices for various regularizations. See Section 3.1 for details.*

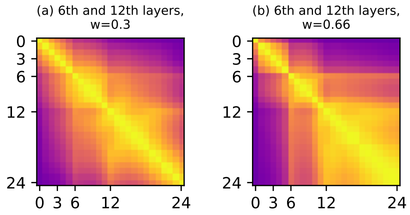

Figure 2: *SVCCA similarity matrices for intermediate CTC variants. See Section 3.2 for details.*

3.2. Effect of intermediate CTC variants

The position variant of intermediate CTC is first explored by [22], concluding that its impact to the accuracy is minimal. However, we find the variant gives big difference on layer similarity and layer pruning.

First, we put an additional intermediate layer at the 6th layer, in addition to the 12th one. As shown at Figure 2 (a), it increases the similarity of layers below the 12th layer, compared to Figure 1 (d). This suggests that multiple branch variant affects the performance of pruning.

We then investigate the effect of the weight for the intermediate loss. [22] used $w = 0.3$ as it is sufficient to enhance the model performance, but it may not be an optimal value for pruning, especially if there are multiple intermediate layers. Thus, we give equal weight to the two intermediate layers and the final layer, i.e., $w = 0.66 \approx 2/3$. Figure 2 (b) shows that the increase of intermediate weight helps overall similarity.

From the observations, we use the newly found hyperparameters for the main experiments at Section 5.

4. Layer pruning on demand

Based on the layer similarity analysis of Section 3, we propose two strategies to remove the encoder layers without much degradation of the overall accuracy. First, we show that intermediate CTC, originally proposed for regularization, can be used for layer pruning as well. Second, we present a more fine-grained pruning strategies which combines iterative search and intermediate CTC. We emphasize that the strategies are for on-demand pruning, and there is no fine-tuning step after pruning.

4.1. Intermediate CTC as layer pruning

The intermediate CTC loss is originally proposed for regularization, and it is not used during inference. However, in the context of layer pruning, the intermediate layer suggests a natural strategy to induce a smaller sub-model from the original model: remove all of the layers after the intermediate layer.

Moreover, we note that the method can be even extended to *any depth*, even if the corresponding layer is not explicitly trained with the intermediate CTC loss. It is because the linear projection layer of Eq. (3) is independent of specific layer and is shared by the intermediate and last layers, enabling that it can be used by other layers. Also, with the improved regularization presented at Section 3.2, we expect layers between the intermediate layer and the last layer give reasonable performance.

For the layers used as intermediate branches during training, we find the performance of their sub-models is as same as the one of the individually trained models of same depth. For the other layers, the performance is only slightly worse than one of individual models. We show the experimental result at Section 5.

4.2. Iterative layer pruning

LayerDrop [5] presents pruning strategies for stochastic depth regularization. The authors tried two strategies, removing every other layer and exhaustive search, and concluded the two strategies do not give significant difference. They also found removing consecutive layers causes significant degradation.

However, we find removing the last half gives nearly optimal performance, due to intermediate CTC. Especially, the intermediate sub-model is significantly better than removing every other layer. This suggests careful pruning strategy would be beneficial.

In this end, we suggest an iterative search augmented with the intermediate model. We start with the original model, and iteratively decrease model depth by one.

For brevity, we denote sub-models as the set of number indicating the layer of the original model. For example, if the 2-layer sub-model uses the 2nd and 4th layers of the original model, we denote the model as $\{2, 4\}$.

For a given sub-model of depth k, we find a new sub-model of depth $k - 1$ by the following steps:

1. For each layer of the current model, we induce a new sub-model by removing the layer from the model. For example, from a given model $\{2, 3, 4\}$, we induce three sub-models: $\{2, 3\}, \{2, 4\}, \{3, 4\}$.
2. We also induce an intermediate sub-model $\{1, \dots, k - 1\}$ from the original model. In the current example, the intermediate sub-model is $\{1, 2\}$.
3. Evaluate the induced sub-models using the validation set and choose the sub-model with the best accuracy.

Step 2 ensures that intermediate sub-models are always included in search space. Without Step 2, we found the iterative search may not discover them and sometimes perform worse.

Note that when a model is evaluated, its intermediate sub-models can be evaluated as well with very small additional cost. For example, evaluation of $\{2, 3, 4\}$ can be done along with evaluations of $\{2, 3\}$ and $\{2\}$. This reduces computational cost for the search.

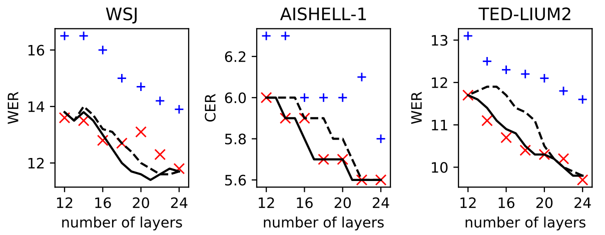

Figure 3: *Word error rates (WERs) for WSJ and TED-LIUM2 and character error rates (CERs) for AISHELL-1.* $+$*: baseline A,* ×*: baseline B, dashed: intermediate sub-models, solid: sub-models found by iterative search. Note that all sub-model is induced from the one model without fine-tuning. See Section 5.1 for details.*

5. Experiments

We use three ASR corpora for the experiments: Wall Street Journal (WSJ) [36] (English; 81 hours), AISHELL-1 [37] (Chinese; 170 hours), and TED-LIUM2 [38] (English; 207 hours).

We use ESPnet [39] for the experiments. For the input, we use 80-dimensional log-mel features with 3-dimensional pitch features and apply SpecAugment [40] during training. We put two convolution layers of stride 2 before Transformer layers. For the output, we use character-based tokenization for WSJ and AISHELL-1 and 2000 subwords using SentencePiece [41] for TED-LIUM2.

5.1. Main results

For layer pruning, we train a 24-layer Transformer model with intermediate CTC and stochastic depth. Following the findings of Section 3, we put intermediate branches at the 6th and 12th layers, and set weight $w = 0.66 \approx 2/3$. The model is called as the pruning-aware model. After the training, we induce smaller sub-models from the pruning-aware model by removing the layers up to the half, either using an intermediate strategy or iterative strategy in Section 4. We do not fine-tune any sub-models after pruning.

To measure the performance of sub-models, we train individual baseline models with the same depth from scratch. For each baseline, we prepare two configurations: (A) not using intermediate CTC or stochastic depth, and (B) using both of intermediate CTC and stochastic depth following [22]. The baseline B is more challenging for sub-models to reach, as the regularizations are not only useful for pruning but also for improving individual models [22, 27, 28, 5].

All models are trained for 100 epochs for WSJ, and for 50 epochs for AISHELL-1 and TED-LIUM2. We emphasize that the pruning-aware model uses exactly same number of epochs as the individual baseline models do. Thus, it only requires computational cost for training that the 24-layer baseline model does. During testing, we use greedy decoding and do not use any external language models (LMs), enabling fast, parallel and non-autoregressive generation.

Figure 3 shows the word error rates (WERs) for WSJ and TED-LIUM2 and character error rates (CERs) for AISHELL-1. The baselines are displayed as individual marks and pruned sub-models are displayed as lines. We first see the pruning-aware model and 24-layer baseline B have nearly same error rates. This indicates that the proposed pruning method does not harm the full model performance. Also, the half-sized sub-model and 12-layer baseline B also have nearly same error rates. Iterative pruning reaches the individual baseline B models for most of cases. This shows the effectiveness of the suggested method.

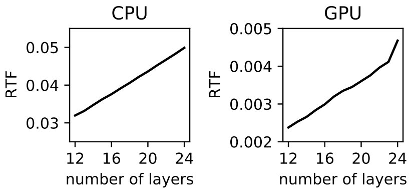

Figure 4: *Real-time factor (RTF) of the proposed model for varying depth. See Section 5.1 for details.*

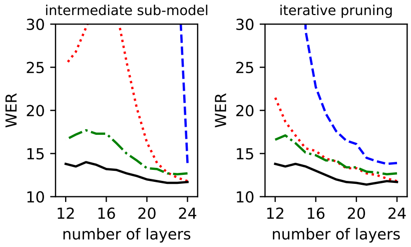

Figure 5: *Word error rates (WERs) for WSJ for sub-models induced from 24-layer model. Solid: proposed model, dashed: baseline A, dotted: baseline B. dashdot: baseline B with in-place distillation. See Section 5.2 for details.*

We measure the real-time factor (RTF) of the pruning-aware model for each number of layers. We use 1 Xeon CPU of 2.2GHz and 1 P40 GPU respectively. Figure 4 shows that the proposed model is very fast (RTF 0.05 on CPU and 0.005 on GPU), due to lightweight and non-autoregressive properties of CTC, and it can further improve RTF (2.5x on GPU) at the cost of small degradation on WER (18%).

5.2. Ablation study

In this section, we show how the change of intermediate CTC proposed at Section 3.2 improves sub-models. We prune 24-layer baseline models of Section 5.1 and compare their performance to the proposed pruning-aware model. Figure 5 shows the WER of sub-models induced by each model, either by intermediate strategy (left) or iterative pruning (right).

We see that baseline A (dashed), without any regularization, does not work well with pruning. It is much improved by baseline B (dotted), but it is still not close to individual baselines. We also see that, for layers between the intermediate and final one, the intermediate sub-model performs poorly.

Finally, the proposed model (solid) gives huge improvement over baseline B, implying that the adjustment of intermediate CTC is very important for pruning, although it does not affect the performance of the full model.

As a preliminary experiment, we also apply in-place distillation [4] to the baseline B. The result (dashdot) shows that, while it improves intermediate sub-models, it does not improve the iteratively pruned sub-models and degrades the full model.

6. Conclusion

We present a training method for CTC-based ASR models, which allows reducing model depth after training, without any fine-tuning. The method is based on intermediate CTC and stochastic depth, previously used as regularization during training. We investigate the behavior of the regularization methods using SVCCA and improve intermediate CTC for layer pruning while retaining the regularization power. We empirically show the 24-layer Transformer CTC model can be pruned to 12-layer which reaches the performance of models trained from scratch.

7. References

[1] A. Ignatov *et al.*, "Ai benchmark: All about deep learning on smartphones in 2019," 2019.

[2] J. Lee *et al.*, "On-device neural net inference with mobile gpus," 2019.

[3] J. Yu *et al.*, "Slimmable neural networks," in *Proc. ICLR*, 2019. [Online]. Available: https://openreview.net/forum?id=H1gMCsAqY7

[4] J. Yu and T. Huang, "Universally slimmable networks and improved training techniques," in *Proc. ICCV*, 2019. [Online]. Available: https://arxiv.org/abs/1903.05134

[5] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," in *Proc. ICLR*, 2020. [Online]. Available: https://openreview.net/forum?id=SylO2yStDr

[6] T. Vu, M. Eder, T. Price, and J.-M. Frahm, "Any-width networks," 2020.

[7] A. Graves *et al.*, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[8] K. Audhkhasi *et al.*, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *Proc. ICASSP*, 2018.

[9] ——, "Forget a Bit to Learn Better: Soft Forgetting for CTC-Based Automatic Speech Recognition," in *Proc. Interspeech*, 2019.

[10] V. Pratap *et al.*, "Scaling up online speech recognition using convnets," in *Proc. Interspeech*, 2020.

[11] S. Kriman *et al.*, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *Proc. ICASSP*, 2020.

[12] A. Baevski *et al.*, "wav2vec 2.0: A framework for self-supervisedlearning of speech representations," in *Proc. NeurIPS*, 2020.

[13] J. K. Chorowski *et al.*, "Attention-based models for speech recognition," in *Proc. NeurIPS*, 2015.

[14] W. Chan *et al.*, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016.

[15] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018.

[16] S. Karita *et al.*, "Improving Transformer-Based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration," in *Proc. Interspeech*, 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1938

[17] A. Graves, "Sequence transduction with recurrent neural networks," 2012.

[18] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[19] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. Interspeech*, pp. 66–70, 2019.

[20] A. Gulati *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020.

[21] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on espnet toolkit boosted by conformer," *arXiv preprint arXiv:2010.13956*, 2020.

[22] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. ICASSP*, 2021. [Online]. Available: https://arxiv.org/abs/2102.03216

[23] C. Talnikar *et al.*, "Joint masked cpc and ctc training for asr," 2021.

[24] Y. Higuchi *et al.*, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," in *Proc. Interspeech*, 2020.

[25] W. Chan *et al.*, "Imputer: Sequence modelling via imputation and dynamic programming," in *Proc. ICML*, 2020.

[26] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," 2020.

[27] G. Huang *et al.*, "Deep networks with stochastic depth," in *Proc. ECCV*, 2016. [Online]. Available: https://arxiv.org/abs/1603.09382

[28] N.-Q. Pham *et al.*, "Very Deep Self-Attention Networks for End-to-End Speech Recognition," in *Proc. Interspeech*, 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2702

[29] J. Li *et al.*, "Learning small-size dnn with output-distribution-based criteria," in *Proc. Interspeech*, 2014.

[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[31] R. Takashima, S. Li, and H. Kawai, "An investigation of a knowledge distillation method for ctc acoustic models," in *Proc. ICASSP*, 2018.

[32] Z. Meng *et al.*, "Domain adaptation via teacher-student learning for end-to-end speech recognition," in *Proc. ASRU*, 2019.

[33] T. Moriya *et al.*, "Self-distillation for improving ctc-transformer-based asr systems," in *Proc. Interspeech*, 2020.

[34] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[35] M. Raghu *et al.*, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *Proc. NeurIPS*, 2017. [Online]. Available: https://arxiv.org/abs/1706.05806

[36] D. B. Paul and J. M. Baker, "The design for the wall street journal-based CSR corpus," in *Proc. Workshop on Speech and Natural Language*, 1992.

[37] H. Bu *et al.*, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *Proc. O-COCOSDA*, 2017.

[38] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proc. LREC*, 2014.

[39] S. Watanabe *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018.

[40] D. S. Park *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019.

[41] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. EMNLP: System Demonstrations*, 2018. [Online]. Available: https://www.aclweb.org/anthology/D18-2012