INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Intra-ensemble: A New Method for Combining Intermediate Outputs in Transformer-based Automatic Speech Recognition

DoHee Kim, Jieun Choi and Joon-Hyuk Chang

Hanynag University, Seoul, Republic of Korea
{dohe0342, uni1018, jchang}@hanyang.ac.kr

Abstract

Deep learning models employ various regularization techniques to prevent overfitting and enhance generalization. In particular, an auxiliary loss, as proposed for connectionist temporal classification (CTC) models, demonstrated the potential for intermediate prediction to be useful by enabling sub-models to recognize speech accurately. We propose a new method called Intra-ensemble, which combines these accurate intermediate outputs into a single output for both training and inference, considering the importance of the intermediate layer using learnable parameters. Our approach is applicable to CTC models, attention-based encoder-decoder models, and transducer structures and demonstrated performance improvements of 13.5%, 3.0%, and 4.1% respectively, in the LibriSpeech evaluation. Furthermore, through various analytical experiments, we found that the sub-models contributed significantly to performance improvement.

Index Terms: speech recognition, ensemble

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) systems have become the mainstream ASR scientific research topic, owing to the remarkable progress in deep neural networks [1, 2]. There are three fundamental categories of E2E models: connectionist temporal classification (CTC) [3] models [4–11], attention-based encoder-decoder (AED) models [12–14], and transducers [15–17]. The CTC model comprises an encoder that maps speech to acoustic features and a linear layer that subsequently maps these features to the corresponding label probability at each time step. The AED model comprises an encoder that encodes the acoustic features and a decoder that generates sentences in an autoregressive manner. The transducer model comprises an encoder, prediction network, and joint network. The encoder maps input speech features to a sequence of hidden representations. The prediction network generates hidden representations using the embedding vector of the preceding token. The joint network amalgamates the outputs of the encoder and prediction network to generate the next potential token. The encoder, which is shared across all three models, usually relies on the transformer structure [18, 19] and, in some instances, includes convolution modules to better leverage the local context [20].

In addition to the architectures, deep learning models employ various regularization techniques during training to prevent overfitting and enhance generalization. With unlimited computation, the optimal regularization approach would be to average the predictions of all possible parameter settings [21]. Nonetheless, this ensemble method has the drawbacks of requiring multiple training and increasing the decoding time. Dropout [22] is a technique that trains models by randomly deactivating nodes among connected nodes, and LayerDrop [23] extended this approach to the layers. This can be viewed as training an ensemble of sub-models. Another approach involves adding an auxiliary loss. InterCTC [10] achieved a regularization effect by including the CTC loss in the intermediate layers of the CTC model. Additionally, [5] presented guidance by hierarchically dividing token units of intermediate loss, while Self-conditioned CTC [11] improved the performance by assuming that sub-models can already recognize speech and used it as a condition. The addition of auxiliary losses trains the sub-models directly, enabling them to recognize speech; however, their output is not utilized during testing.

Recent studies indicated that using intermediate prediction trained with intermediate loss can accurately recognize speech. In [24], the intermediate prediction was employed to enhance the performance through iterative decoding. [25] achieved fast decoding by pruning intermediate layers of the trained model. However, they were unable to avoid a decrease in performance or an increase in decoding time due to repeated decoding or pruning processes.

To address this issue, we propose a simple yet effective training method called Intra-ensemble. Intra-ensemble involves mixing intermediate outputs without increasing the decoding time or causing performance degradation. Intra-ensemble combines the sub-model outputs by performing a weighted sum using learnable parameters. This weighted sum feature, called contextualized representation, is used not only during train time but also during test time. Unlike previous studies that applied their methods to only one ASR structure; CTC models, we demonstrated that Intra-ensemble can enhance the performance for all three structures - CTC models, AED, and transducer. For the CTC models, the Intra-ensemble demonstrated an average improvement of 9.3% for the TEDLIUM2 dataset and 13.5% for the LibriSpeech dataset. Furthermore, when applied to the AED models, it exhibited an average performance improvement of 3.0% for the LibriSpeech dataset. When applied to the conformer-transducer models, it exhibited an average performance improvement of 4.1%. Moreover, through several analytical experiments described in Section 4, we confirmed that the sub-models significantly contributed to performance improvement.

2. Proposed method

2.1. Intermediate CTC

Intermediate CTC (InterCTC) is an auxiliary loss designed for CTC models for regularization [10]. Let us consider a multi-layered transformer-based architecture that employs a CTC loss function. In this structure, the encoder comprises L layers for

10.21437/Interspeech.2023-1255

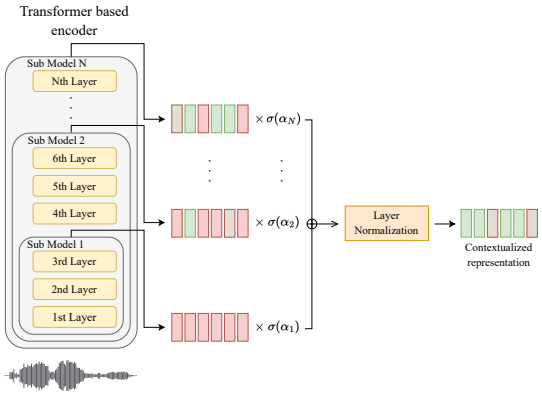

Figure 1: *Overview diagram of the proposed method. The contextualized representation is generated by the weighted sum of intermediate layer outputs using a learnable parameter α. The learnable parameter α is trained alongside the model during training and fixed at its learned value during test time.*

a given input $\mathbf{X}_0 \in \mathbb{R}^{T \times D}$ of length T and dimension D. The CTC then calculates the probability of the target sequence $\mathbf{y}$ by considering all the possible alignments for the label and input length T. The likelihood of the target sequence $\mathbf{y}$ and encoder output $\mathbf{X}_L$ is defined as:

$$\mathbf{Z}_L = Softmax(Linear_{D \to |V'|}(\mathbf{X}_L)) \tag{1}$$

$$P_{CTC}(\mathbf{y}|\mathbf{Z}_L) = \sum_{\pi \in \beta^{-1}(\mathbf{y})} P(\pi|\mathbf{Z}_L) \tag{2}$$

where $\beta^{-1}(\mathbf{y})$ denotes the set of alignments π of length T which are compatible with the target sequence $\mathbf{y}$ including the special blank token and $|V^{'}|$ denotes the output token vocabulary size, including the blank token. $Softmax(\cdot)$ denotes a softmax function, and $Linear(\cdot)$ denotes a linear layer that maps a D-dimensional vector into a $|V^{'}|$-dimensional vector. During training, we minimized the negative log-likelihood induced by the CTC using $P_{CTC}(\mathbf{y}|\mathbf{Z}_L)$ in Eq. (2):

$$\mathcal{L}_{CTC} = -logP_{CTC}(\mathbf{y}|\mathbf{Z}_L) \tag{3}$$

The intermediate loss is attached to the intermediate layer of the encoder at the layer indices set $\mathcal{K}$ and is defined as follows:

$$\mathcal{L}_{InterCTC} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} -logP_{CTC}(\mathbf{y}|\mathbf{Z}_k) \tag{4}$$

where k denotes a layer index and $\mathbf{Z}_k$ denotes k_{th} intermediate prediction. Note that the linear layer used to obtain $\mathbf{Z}_k$ is shared with the one used to obtain $\mathbf{Z}_L$. Subsequently, the training objective is defined by combining Eqs. (3) and (4):

$$\mathcal{L} = (1-w)L_{CTC} + wL_{InterCTC} \tag{5}$$

with the hyperparameter w. Note that intermediate predictions are only used during training and not at test time.

2.2. Self-conditioned CTC

InterCTC introduced an auxiliary loss from the intermediate layers of the encoder. On the other hand, the Self-conditioned CTC (SelfCond) further utilizes the intermediate predictions as a condition for the next encoder input:

$$\mathbf{X}_l^{cond} = \begin{cases} \mathbf{X}_l + Linear_{|V'| \to D}(\mathbf{Z}_l) & (l \in \mathcal{K}) \\ \mathbf{X}_l & (l \notin \mathcal{K}) \end{cases} \tag{6}$$

where $Linear_{|V'| \to D}(\cdot)$ maps a $|V^{'}|$-dimensional vector into a D-dimensional vector for each element in the input sequence and $\mathcal{K}$ denotes the set of intermediate layer indices. Note that this linear layer is shared among the $|\mathcal{K}|$ intermediate layers. During training, SelfCond utilizes an additional loss, the same as InterCTC in Eq. (5). The intermediate prediction is utilized to condition the next layer. However, Similar to InterCTC, only the output of the last layer is used for the final prediction.

2.3. Intra-ensemble

To directly utilize the predictions of the intermediate layer, we propose Intra-ensemble. Our primary concerns behind Intra-ensemble are determining which intermediate layers to use and how to combine the intermediate layer outputs into a single representation. The intermediate layer's output can be viewed as the output of a sub-model, which should be sufficiently accurate to recognize speech. We chose the same choice for the intermediate layer as SelfCond because they hypothesized that the intermediate prediction is sufficient to recognize speech on their intermediate layer set $\mathcal{K}$. The most straightforward approach for combining intermediate outputs is to average the outputs. However, averaging fails to capture the importance of each layer. Thus, we adopt the learnable parameter α to address this issue. The α contains $|\mathcal{K}|$ parameters that correspond to the $|\mathcal{K}|$ intermediate layers and is trained alongside the model. We utilize the sigmoid function for alpha in order to introduce nonlinearity. We provide a detailed explanation of the importance of α in Section 4.2.3. The representation obtained through the selected intermediate outputs and the learnable parameter α is as follows:

$$\mathbf{X}^{context} = LayerNorm(\sum_{k \in \mathcal{K}} \sigma(\alpha_k)\mathbf{X}_k) \tag{7}$$

where $\sigma(\cdot)$ denotes a sigmoid function. As described in Figure. 1, this representation is referred to as contextualized representation, which we employ for both training and inference. As the contextualized representation simply involves the weighted sum of the intermediate layer outputs that naturally arises during the computation of the complete model, the training time is slightly increased, without any increase the in decoding time.

2.4. Related work

In data2vec [27, 28] utilized intermediate layer outputs, trained the model using the contextualized representation as a self-target. However, the contextualized representation is obtained as the mean of all layers and was used for self-supervised learning manner, not for the ASR task. Gtrans [29] proposed the use of intermediate layer outputs to train deep networks in a seq2seq structure for machine translation tasks. The Intra-ensemble method in contrast, works not only for AED structure but also for CTC and Transducer models.

Our work can also be viewed as an extension of studies that enhance the performance of the encoder. [6–8] introduced an additional network to enhance the performance of the encoder for CTC. [6, 7] improved the performance of the encoder by masking its output and using an additional decoder to predict the masked portion. [8] enhanced performance by utilizing

Table 1: *WER on the TEDLIUM2, LibriSpeech 100h/960h datasets for CTC models. Above the dashed line, the results for TEDLIUM2 and LibriSpeech 100h were obtained from [26], while the results for LibriSpeech 960h were derived from [5]. In all experiments, greedy decoding was used during the decoding process. Highlighted in bolds indicate the best performance achieved.*

Method	TEDLIUM2			LibriSpeech-100h					LibriSpeech-960h				
	dev	test	RERR	dev-clean	dev-other	test-clean	test-other	RERR	dev-clean	dev-other	test-clean	test-other	RERR
Previous work													
CTC	8.9	8.6	-	7.4	20.5	7.8	20.8	-	3.8	9.8	3.9	9.9	-
InterCTC	8.5	8.3	4.0 %	6.9	19.7	7.1	20.2	4.6 %	3.1	8.5	3.3	8.4	15.1 %
SelfCond	8.7	8.0	4.6 %	6.6	19.4	6.9	19.7	6.9 %	**2.9**	7.9	**3.1**	7.9	**20.7** %
Proposed work													
CTC	9.4	8.7	-	7.6	20.6	7.7	20.8	-	3.6	9.1	3.8	9.1	-
+ Intra-ensemble	9.1	8.3	3.4 %	7.2	19.6	7.3	19.7	5.2 %	3.3	8.7	3.5	8.6	5.2 %
InterCTC	8.7	8.3	6.1 %	7.0	19.2	7.2	19.5	6.9 %	3.2	8.7	3.3	8.5	7.1 %
+ Intra-ensemble	**8.4**	8.0	9.1 %	6.6	19.1	6.8	19.3	8.6 %	3.2	8.2	3.3	8.2	10.4 %
SelfCond	8.8	8.1	6.7 %	6.9	19.0	6.9	19.6	7.7 %	3.2	8.2	3.3	8.2	10.2 %
+ Intra-ensemble	8.5	**7.9**	**9.3 %**	**6.6**	**18.8**	**6.7**	**18.9**	**10.1 %**	3.1	**7.9**	3.2	**7.9**	13.5 %

Table 2: *WER on full LibriSpeech for AED models. CTC and 1-best decoding were used during the decoding process.*

Method	LibriSpeech-960h				
	dev-clean	dev-other	test-clean	test-other	RERR
CTC model	3.6	9.1	3.8	9.1	-
CTC-decoding					
18 enc - 6 dec	3.2	7.6	3.2	8.0	-
+ Intra-ensemble	2.9	7.5	3.1	7.5	5.2 %
1-best					
18 enc - 6 dec	2.8	6.7	3.1	6.8	-
+ Intra-ensemble	2.6	6.4	3.1	6.7	3.0 %

the prediction error of the decoder as a loss to train intermediate layers. However, they require additional networks, and the training time is also increased. Furthermore, these studies proposed various methodologies to improve the performance of the encoder, but at test time, only the output of the final layer was used.

3. Results

We evaluated the performance of Intra-ensemble on the two corpora: TEDLIUM2 [30] (English, 207 hours) and LibriSpeech [31] (English, 960 hours). We utilized icefall [32] for all experiments and utilized 80-dimensional log-mel features with 3-dimensional pitch features as inputs, and SpecAugment [33] for data augmentation, in addition to applying speed perturbation [34]. We tokenized label sentences as subwords with sentencepeice [35]. For TEDLIUM2, the model was trained for 60 epochs. For LibriSpeech 100h subset, the model was trained for 70 epochs and 40 epochs for the full LibriSpeech dataset. After training, the model parameter was obtained by averaging models from the last 10 epochs. All experiments were performed on 4 NVIDIA A100 40GB GPUs.

3.1. CTC models

We first investigated our approach for CTC models. For the hyperparameters of the experiments, we mainly followed the icefall recipes. All models comprised an 18-layer conformer encoder with 4 attention heads and 256 hidden dimensions, and 15 convolution kernel sizes. For InterCTC, we selected the 9th layer as the intermediate layer, using a value of 0.3 for w as described in [10]. For SelfCond, we utilized $\mathcal{K} = \{3, 6, 9, 12, 15\}$, and a value of 0.5 for w, as mentioned in [11]. For our method, as we calculated the loss for contextualized representation instead of the output of the final layer, we utilized $\mathcal{K} = \{3, 6, 9, 12, 15, 18\}$, including the final layer when applying an intermediate loss. The results of the CTC models are presented in Table 1. For TEDLIUM2, when our proposed method was applied to InterCTC and SelfCond, it demonstrated an average performance improvement of 9.1% and 9.3%, respectively. Even when applied to a standard CTC model, which does not directly impose a loss on intermediate layers, it still demonstrated

Table 3: *WER on full LibriSpeech for Conformer-T models. No loss other than RNN-T was used. Experiments were conducted for each of the different model sizes.*

Method	# of params	LibriSpeech-960h				
		dev-clean	dev-other	test-clean	test-other	RERR
Conformer-T	30.5 M	2.8	7.2	3.0	7.1	-
	78.6 M	2.5	6.4	2.7	6.4	-
	116.5 M	2.3	5.8	2.5	5.8	-
Intra-ensemble	30.5 M	2.6	6.9	2.8	6.9	4.1 %
	78.6 M	2.4	6.2	2.4	6.0	3.5 %
	116.5 M	2.2	5.6	2.4	5.6	3.1 %

Table 4: *A WER comparison based on the number of epochs on LibriSpeech test sets. The model used in this experiment was trained on LibriSpeech 960h.*

Epoch	10 epoch	20 epoch	30 epoch	40 epoch
WER(%)	5.5	4.2	3.8	3.6

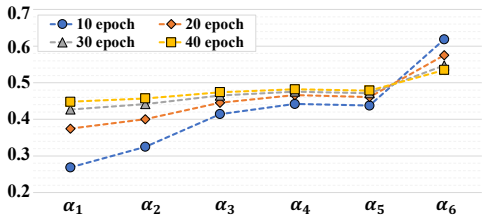

Figure 2: *Visualization of learnable parameter alpha on the different number of epochs.*

an average performance improvement of 3.4%. For the LibriSpeech 100h subset, our method demonstrated a 10.1% performance improvement, exhibiting a similar trend to TEDLIUM2. For LibriSpeech 960h, our method achieved an average performance improvement of 13.5% compared to the standard CTC model. Note that for LibriSpeech 960h, our method showed a lower average performance improvement compared to previous studies, likely due to the superior performance of the standard CTC model.

3.2. AED

We then explored our method for AED models. All models comprised a 6-layer transformer decoder with 4 attention heads and 256 hidden dimensions and utilized the same encoder architecture as the CTC models. Table 2 presents the results of our method for LibriSpeech 960h. Since AED models employ CTC loss for the encoder, we first performed $CTC - decoding$ using only the encoder to verify the performance of the encoder itself. Similar to the CTC models, we achieved an average performance improvement of 5.2% when applying our method. Additionally, even though the encoder has same parameters as the CTC models, we confirmed a performance improvement of

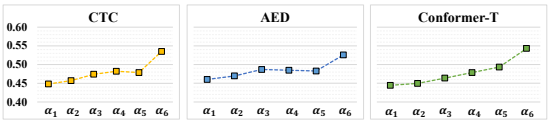

Figure 3: *Visualization of learnable parameter α on different ASR architecture.*

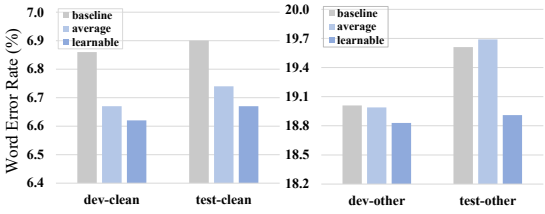

Figure 4: *A WER comparison of two different methods for generating contextualized representations on LibriSpeech dev/test sets.*

20.9% compared with the existing CTC model when using the decoder for training. When applying our method to $1-best$ decoding using the decoder, we achieved a performance improvement of 3.0%. Through this experiment, we confirmed that the proposed method can improve the performance of AED models.

3.3. Transducer

We investigated our approach for the conformer-transducer (conformer-T) which uses a conformer as the encoder. For conformer-T, we did not use any loss other than the RNN-T loss to investigate the effects of our approach. We utilized the pruned RNN-T loss [36] owing to its fast and memory-efficient training. In addition, in order to demonstrate that our method can be applied to other model sizes, we conducted experiments on three different model. As described in Table 3, the model with 30.5 M parameters utilized the same CTC model for the encoder, the prediction network with 512 hidden units, and the joint network with a dimension of 512. The Intra-ensemble demonstrated an average performance improvement of 4.5% compared to the baseline. For the model with an encoder containing 512 hidden units and 12 layers (denotes 78.6 M parameters), we chose $\mathcal{K} = \{3, 6, 9, 12\}$. Our method exhibited an average performance improvement of 3.5%. The model with an encoder containing 512 hidden units and 18 layers (denotes 116.5 M parameters), it exhibited an average performance improvement of 3.1%, demonstrating that the Intra-ensemble can improve the performance of models with a large number of parameters.

4. Analysis

4.1. Importance of sub-models

In Table 4 and Figure 2, we visualized the performance and α values by epoch. We trained the standard CTC model with Intra-ensemble on the LibriSpeech 960h dataset and evaluated it on the test-clean subset. We utilized $\mathcal{K} = \{3, 6, 9, 12, 15, 18\}$ to obtain intermediate outputs. As α passes through the sigmoid function, we visualized $\sigma(\alpha)$. Since $\sigma(\alpha)$ serves as the weight for intermediate outputs, it can also be seen as indicating the importance of each layer. In Figure 2, we can observe that the weight for the final layer decreases, and the weight for sub-models increases as training progresses, resulting in improved performance. Therefore, the output of sub-models played a significant role in improving performance.

Table 5: *A WER comparison based on the number of intermediate layers.*

# of α	LibriSpeech-100h				
	dev-clean	dev-other	test-clean	test-other	RERR
baseline	7.6	20.6	7.7	20.8	-
2	7.4	19.9	7.5	20.4	2.6 %
3	7.3	19.7	7.5	20.4	3.2 %
6	**7.2**	**19.6**	**7.3**	**19.7**	**5.2 %**
9	7.3	19.9	7.9	20.0	3.6 %
18	7.6	20.6	7.8	21.0	-0.2 %

4.2. α visualization

In Figure 3, we visualized α for different ASR architectures when Intra-ensemble is applied. All encoders have 18 layers and 256 hidden units and we utilized the standard models with our method added. As with Figure 2, we visualized α passed through the sigmoid function. From Figure 3, we can observe that although the weight for the final layer is relatively high, the weight for sub-models is not insignificant, indicating that the output of all sub-models is appropriately utilized.

4.3. Ablation study: Importance of learnable parameter α

To investigate the effect of the learnable parameter α, we performed Intra-ensemble without α. Since there is no learnable parameter α, we averaged the intermediate outputs to obtain the contextualized representation. We added the Intra-ensemble to SelfCond and trained on LibriSpeech 100h subset. As shown in Figure 4, since the intermediate layers of SelfCond recognize speech well, we achieved performance improvement compared to the baseline simply by averaging the intermediate outputs. However, since the average of the intermediate outputs does not consider the weight of each layer during training, adding α resulted in a more significant performance improvement.

4.4. Study on intermediate layer selection

We evaluated whether the intermediate output we selected was appropriate by investigating performance based on sub-model size. To do this, we trained on Librispeech 100h with Intra-ensemble applied to a standard CTC model containing 18 layers and 256 hidden units. We compared results by ensembling intermediate outputs for all factors of the encoder layer count. As shown in Table 5, the performance was improved compared to the baseline if the sub-model is large enough to achieve accurate recognition. However, if the size of the sub-model is too large, the performance improvement is limited due to insufficient ensembling. According to the experimental results, the appropriate size for the sub-model was found to be three layers.

5. Conclusions

We proposed the Intra-ensemble that generates contextualized representation using learnable parameter α and uses it for training and inference, improving performance without increasing decoding time. Our algorithm is easy to implement and can be applied to CTC, AED, and transducer structures. In addition, we experimentally demonstrated that it can be applied orthogonally to previous research and can further improve performance. Our algorithm can also be applied to other tasks that use transformer which we leave as the future work.

Acknowledgement This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University))

6. References

[1] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2016, pp. 4945–4949.

[2] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs RNN in speech applications," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop* (ASRU), 2019, pp. 449–456.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. International Conference on Machine Learning* (ICML), 2006, pp. 369–376.

[4] N. Chen, S. Watanabe, J. Villalba, P. Żelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 121–125, 2020.

[5] Y. Higuchi, K. Karube, T. Ogawa, and T. Kobayashi, "Hierarchical conditional end-to-end asr with ctc and multi-granular subword units," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2022, pp. 7797–7801.

[6] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *arXiv preprint arXiv:2005.08700*, 2020.

[7] Y. Higuchi, H. Inaguma, S. Watanabe, T. Ogawa, and T. Kobayashi, "Improved mask-ctc for non-autoregressive end-to-end asr," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2021, pp. 8363–8367.

[8] T. Komatsu and Y. Fujita, "Interdecoder: using attention decoders as intermediate regularization for ctc-based speech recognition," in *Proc. IEEE Spoken Language Technology Workshop* (SLT), 2023, pp. 46–51.

[9] A. Tjandra, C. Liu, F. Zhang, X. Zhang, Y. Wang, G. Synnaeve, S. Nakamura, and G. Zweig, "Deja-vu: Double feature presentation and iterated loss in deep transformer networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2020, pp. 6899–6903.

[10] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2021, pp. 6224–6228.

[11] J. Nozaki and T. Komatsu, "Relaxing the conditional independence assumption of ctc-based asr by conditioning on intermediate predictions," *arXiv preprint arXiv:2104.02724*, 2021.

[12] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," vol. 28, 2015.

[13] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2016, pp. 4960–4964.

[14] T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *Proc. Interspeech*, 2019.

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

[16] R. Botros, T. N. Sainath, R. David, E. Guzman, W. Li, and Y. He, "Tied & reduced RNN-T decoder," *arXiv preprint arXiv:2109.07513*, 2021.

[17] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop* (ASRU), 2019, pp. 114–121.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems* (NeurIPS), vol. 30, 2017.

[19] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2018, pp. 5884–5888.

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

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

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Springer European Conference on Computer Vision* (ECCV), 2016, pp. 646–661.

[24] T. Komatsu, Y. Fujita, J. Lee, L. Lee, S. Watanabe, and Y. Kida, "Better intermediates improve ctc inference," *arXiv preprint arXiv:2204.00176*, 2022.

[25] J. Lee, J. Kang, and S. Watanabe, "Layer pruning on demand with intermediate ctc," *arXiv preprint arXiv:2106.09216*, 2021.

[26] Y. Higuchi, N. Chen, Y. Fujita, H. Inaguma, T. Komatsu, J. Lee, J. Nozaki, T. Wang, and S. Watanabe, "A comparative study on non-autoregressive modelings for speech-to-text generation," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop* (ASRU), 2021, pp. 47–54.

[27] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *Proc. International Conference on Machine Learning* (ICML), 2022, pp. 1298–1312.

[28] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, "Efficient self-supervised learning with contextualized target representations for vision, speech and language," *arXiv preprint arXiv:2212.07525*, 2022.

[29] J. Yang, Y. Yin, L. Yang, S. Ma, H. Huang, D. Zhang, F. Wei, and Z. Li, "Gtrans: Grouping and fusing transformer layers for neural machine translation."

[30] A. Rousseau, P. Deléglise, Y. Esteve *et al.*, "Enhancing the ted-lium corpus with selected data for language modeling and more ted talks." in *Proc. HLT International Conference on Language Resources and Evaluation* (LREC), 2014, pp. 3935–3939.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2015, pp. 5206–5210.

[32] "icefall," https://github.com/k2-fsa/icefall.

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

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

[35] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[36] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned RNN-T for fast, memory-efficient asr training," *arXiv preprint arXiv:2206.13236*, 2022.