Block Refinement Learning for Improving Early Exit in Autoregressive ASR

Naotaka Kawata*†, Shota Orihashi*, Satoshi Suzuki*, Tomohiro Tanaka*, Mana Ihori*, Naoki Maikishima*, Taiga Yamane*, and Ryo Masumura*
* NTT Human Informatics Laboratories, NTT Corporation, Japan
E-mail: naotaka.kawata@ntt.com
† NTT TechnoCross Corporation, Japan

***Abstract*—While the autoregressive transformer models of automatic speech recognition (ASR) are highly accurate, the inference time is long because of their sequential decoding. Early exit is a technique that aims to speed up the inference process by terminating it early on the basis of output from intermediate decoder layers. When an inference result with a high-confidence token is obtained in low intermediate layers, inference can be terminated at that point and the computational complexity is reduced compared to not terminating. In order to terminate inference early, it is necessary to improve the accuracy of the low intermediate layers. However, early exit often focuses on improving the accuracy of high intermediate layers because they determine the upper accuracy limit of ASR. As a result, it is difficult to terminate inference in low intermediate layers because the confidence of those layers is low. To solve this problem, we propose block refinement learning (BRL), which is a re-training method of existing early-exiting models. BRL trains the low intermediate layers while maintaining the overall accuracy of the model by considering gradients of not only low intermediate layers but also high intermediate layers. In this way, low intermediate layers can be trained while maintaining the accuracy of the high intermediate layers. We demonstrated the effectiveness of BRL on Japanese discourse ASR tasks.**

I. Introduction

Reducing inference time is important for automatic speech recognition (ASR). Previous studies have reduced the inference time by reducing the model parameters and computational complexity through the use of knowledge distillation [1]–[3], quantization [4]–[6], and pruning [7]–[9].

On the other hand, early exit [10], [11], which aims to speed up the inference process of ASR, has attracted attention in recent years due to its high affinity with ASR models that have multi-layer decoder. Early exit trains a model that has output layer associated with each of its intermediate decoder layers as well as the final decoder layer. In this paper, this model is referred to as the "early-exiting model". Also, the pair of a decoder layer and an output layer attached to it is referred to as the "intermediate layer". The early-exiting model can reduce the inference time by dynamically selecting inference paths for each input on the basis of the outputs of the intermediate layers. For instance, by terminating inference in low intermediate layers when inference result in low intermediate layers is high confidence and continuing inference until high intermediate layers when inference result in low intermediate layers is low confidence.

In training the early-exiting model, it often focuses on the accuracy of the high intermediate layers because it defines the upper limit of the ASR accuracy. However, the training policy that focuses on the accuracy of the high intermediate layers neglects the accuracy of low intermediate layers. As a result, the effect of reducing the inference time is smaller because it becomes hard for early exit to terminate inference in low intermediate layers. Given this background, a method is needed to improve the accuracy of low intermediate layers while maintaining the accuracy of high intermediate layers.

In this paper, we propose block refinement learning (BRL), which is a training method of the early-exiting model, to improve the ASR accuracy of low intermediate layers. An overview of BRL is shown in Figure 1. The key advantage of BRL is that it improves the ASR accuracy of low intermediate layers while maintaining the ASR accuracy of high intermediate layers in the early-exiting model. To this end, it trains the parameters of low intermediate layers, while keeping the parameters of all other layers fixed. At the same time, BRL trains low intermediate layers by exploiting not only the gradients for the corresponding attached output layers, but also the gradients for all output layers including the high intermediate ones. We demonstrated the effectiveness of BRL on Japanese discourse ASR tasks.

Our contributions are summarized as follows:

- We established a method to improve the ASR accuracy of low intermediate layers while maintaining the ASR accuracy of high intermediate layers in the early-exiting model.
- By our method, we experimentally demonstrated faster inference while maintaining ASR performance.

II. Related work

This section details previous studies on early exit. The previous studies used two different training methods to train the early-exiting models which has multiple output layers: two stage training [12]–[14], in which only the added output layers are trained, and joint training [15]–[17], in which the entire early-exiting model is trained in a single training. With two-stage training, only the added output layers are trained, while the parameters of the other component are fixed. Therefore, it is possible to maintain the performance of the final output

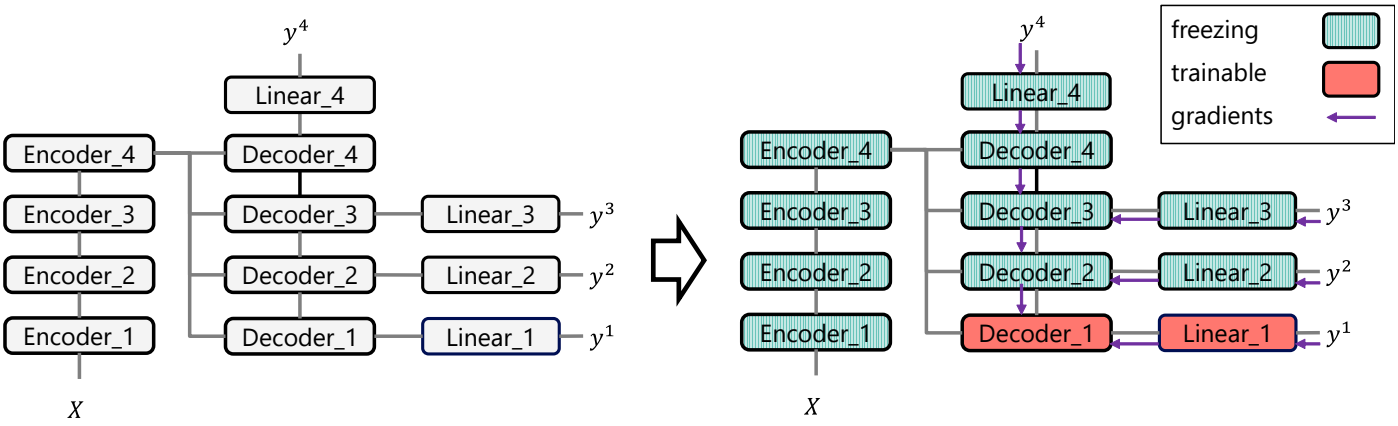

Fig. 1: Overview of the proposed method. The left side shows the early-exiting model, and the right side shows the training of BRL for the early-exiting model. As an example, this figure shows the early-exiting model with four encoder layers and four decoder layers, where the first layer is an intermediate layer trained by BRL.

layer. On the other hand, with joint training, all parameters included in the early-exiting model are trained in a single training. It has been pointed out that the accuracy of low intermediate layers is poor in two-stage training because it trains only the added output layers [18]. Also, the accuracy of low intermediate layers is high in joint training because joint training trains all layers, not just the added ones. In this paper, we focus on joint training for its advantage of reducing the inference time by ensuring the high accuracy of the low intermediate layers.

In inference, the early-exiting model achieves a speedup by terminating the inference early based on exit criteria. The possible exit criteria include entropy-based [19], probability-based [16], [20], and patience-based methods [21]. We investigated the effectiveness of BRL by using probability-based and patience-based exit criteria as representative examples.

III. The Transformer-based Early-exiting Model

A. Transformer-based ASR model

This section details the ASR model based on transformer [22]. This model estimates a text $\boldsymbol{Y} = (y_1, \dots, y_T)$, that is the inference result from acoustic features $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$. y_t is the t-th token of the text, and T is the length of output sequence. $\boldsymbol{x}_n$ is the n-th frame of the features, and N is the frame length. The ASR model estimates the following probability P of generating $\boldsymbol{Y}$ for $\boldsymbol{X}$:

$$P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}) = \prod_{t=1}^{T} P(y_t|y_{1:t-1}, \boldsymbol{X};\boldsymbol{\Theta}), \tag{1}$$

where $y_{1:t-1} = (y_1, \dots, y_{t-1})$ and $\boldsymbol{\Theta}$ denotes the ASR model parameter. The probability P obtained by the transformer-based ASR model is as follows:

$$\boldsymbol{h} = \mathrm{TransformerEnc}(\boldsymbol{X};\boldsymbol{\theta}_{\mathrm{enc}}), \tag{2}$$

$$\boldsymbol{e}_{t-1} = \mathrm{Embedding}(y_{t-1};\boldsymbol{\theta}_{\mathrm{emb}}), \tag{3}$$

$$\boldsymbol{c}^l_{t-1} = \begin{cases} \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{e}_{1:t-1};\boldsymbol{\theta}^l_{\mathrm{dec}}) & \text{if } l = 1, \\ \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{c}^{l-1}_{1:t-1};\boldsymbol{\theta}^l_{\mathrm{dec}}) & \text{otherwise,} \end{cases} \tag{4}$$

$$P(y_t|y_{1:t-1}, \boldsymbol{X};\boldsymbol{\Theta}) = \mathrm{Softmax}(\boldsymbol{c}^L_{t-1};\boldsymbol{\theta}^L_{\mathrm{linear}}), \tag{5}$$

where $\mathrm{TransformerEnc}(\cdot)$ is a transformer encoder that consists of an embedding layer, positional encoding layer, and multiple transformer encoder layers. The transformer encoder layer consists of a multi-head self-attention layer and a feed-forward layer. $\boldsymbol{\theta}_{\mathrm{enc}}$ denotes its parameters. $\mathrm{Embedding}(\cdot)$ is an embedding layer that consists of an embedding layer and a positional encoding layer. $\boldsymbol{\theta}_{\mathrm{emb}}$ denotes its parameters. $\mathrm{TransformerDec}(\cdot)$ is a transformer decoder that consists of a masked-multi-head attention layer, encoder-decoder multi-head-attention layer, and feed-forward layer. $\boldsymbol{\theta}^l_{\mathrm{dec}}$ denotes the parameter of the l-th transformer decoder, where $l \in [1, L]$. L is the number of decoder layers. $\mathrm{Softmax}(\cdot)$ is a softmax layer with a linear transformation. $\boldsymbol{\theta}^l_{\mathrm{linear}}$ denotes the parameter of the l-th softmax layer connected to the l-th transformer decoder. Note that the softmax layer is only connected to the last transformer decoder in the basic ASR model. The parameter $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_{\mathrm{enc}}, \boldsymbol{\theta}_{\mathrm{emb}}, \boldsymbol{\theta}^1_{\mathrm{dec}}, \dots, \boldsymbol{\theta}^L_{\mathrm{dec}}, \boldsymbol{\theta}^L_{\mathrm{linear}}\}$ is optimized with a cross-entropy loss function $\mathcal{L}$, defined as

$$\mathcal{L}(\boldsymbol{\Theta}) = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \sum_{t=1}^{T} \log P(y_t|y_{1:t-1}, \boldsymbol{X};\boldsymbol{\Theta}). \tag{6}$$

$\mathcal{D}$ denotes training data containing multiple $\boldsymbol{X}$ and $\boldsymbol{Y}$ pair data.

B. The early-exiting model

This section details the transformer-based early-exiting ASR model. The model can obtain probabilities P^l from l-th intermediate decoder layer. The probabilities P^l are obtained as follows:

$$P^l(y_t|y_{1:t-1}, \boldsymbol{X};\boldsymbol{\Theta}_E) = \mathrm{Softmax}(\boldsymbol{c}^l_{t-1};\boldsymbol{\theta}^l_{\mathrm{linear}}). \tag{7}$$

Algorithm 1 Inference with probability-based exit criteria (Input: $\boldsymbol{X}$, Output: $\boldsymbol{Y}$)

$\boldsymbol{h} \leftarrow \mathrm{TransformerEnc}(\boldsymbol{X}; \boldsymbol{\theta}_{\mathrm{enc}})$
for $t = 1$ to T **do**
  $\boldsymbol{e}_{t-1} = \mathrm{Embedding}(y_{1:t-1}; \boldsymbol{\theta}_{\mathrm{emb}})$
  for $l = 1$ to L **do**
    if $l = 1$ **then**
      $\boldsymbol{c}^l_{t-1} \leftarrow \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{e}_{1:t-1}; \boldsymbol{\theta}^l_{\mathrm{dec}})$
    else
      $\boldsymbol{c}^l_{t-1} \leftarrow \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{c}^{l-1}_{1:t-1}; \boldsymbol{\theta}^l_{\mathrm{dec}})$
    end if
    $P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}}) \leftarrow \mathrm{Softmax}(\boldsymbol{c}^l_{t-1}; \boldsymbol{\theta}^l_{\mathrm{linear}})$
    $s^l_t \leftarrow \mathrm{Score}(P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}}))$
    if $s^l_t \geq \lambda^l$ **then**
      $y_t \leftarrow \arg\max_{y_t} P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}})$
      break
    end if
  end for
end for

In the early-exiting model, one $\mathrm{Softmax}(\cdot)$ is connected to each $\mathrm{TransformerDec}(\cdot)$. The parameter $\boldsymbol{\Theta}_E = \{\boldsymbol{\theta}_{\mathrm{enc}}, \boldsymbol{\theta}_{\mathrm{emb}}, \boldsymbol{\theta}^1_{\mathrm{dec}}, \ldots, \boldsymbol{\theta}^L_{\mathrm{dec}}, \boldsymbol{\theta}^1_{\mathrm{linear}}, \ldots, \boldsymbol{\theta}^L_{\mathrm{linear}}\}$ is optimized with a cross-entropy loss function $\mathcal{L}_E$, defined as

$$\mathcal{L}_E(\boldsymbol{\Theta}_E) = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \sum_{t=1}^{T} \sum_{l=1}^{L} w^l \log P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_E). \tag{8}$$

The weights w^l set in Schuster et al. [23] favor higher layers, as follows:

$$w^l = \frac{l}{\sum_{k=1}^{L} k}. \tag{9}$$

C. Inference of the early-exiting model

This section details two exit criteria of early exit, probability-based [20] and patience-based (PABEE) [21]. The probability-based exit criteria terminates inference when the early-exiting model outputs high-confidence token from its intermediate layers. Specifically, terminating inference occurs when the confidence score $s^l_t \in [0, 1]$ exceeds the thresholds $\lambda^l \in [0, 1]$. The confidence score $s^l_t = \mathrm{Score}(\cdot)$ is defined by the difference between the top two values of $\mathrm{Softmax}(\cdot)$. The thresholds λ^l are determined by setting arbitrary objective functions and solving an optimization problem. The flow of inference with the probability-based exit criteria is shown in Algorithm 1.

PABEE terminates inference when the early-exiting model outputs same token continuously from its intermediate layers. It uses a patience counter p to store the number of times that outputs do not change and the threshold $\hat{p} \in [1, L-1]$. Terminating inference occurs when p is equal to $\hat{p}$. The flow of inference with the patience-based exit criteria is shown in Algorithm 2.

Algorithm 2 Inference with patience-based exit criteria (Input: $\boldsymbol{X}$, Output: $\boldsymbol{Y}$)

$\boldsymbol{h} \leftarrow \mathrm{TransformerEnc}(\boldsymbol{X}; \boldsymbol{\theta}_{\mathrm{enc}})$
for $t = 1$ to T **do**
  $p \leftarrow 0$
  $\boldsymbol{e}_{t-1} = \mathrm{Embedding}(y_{1:t-1}; \boldsymbol{\theta}_{\mathrm{emb}})$
  for $l = 1$ to L **do**
    if $l = 1$ **then**
      $\boldsymbol{c}^l_{t-1} \leftarrow \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{e}_{1:t-1}; \boldsymbol{\theta}^l_{\mathrm{dec}})$
    else
      $\boldsymbol{c}^l_{t-1} \leftarrow \mathrm{TransformerDec}(\boldsymbol{h}, \boldsymbol{c}^{l-1}_{1:t-1}; \boldsymbol{\theta}^l_{\mathrm{dec}})$
    end if
    $P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}}) \leftarrow \mathrm{Softmax}(\boldsymbol{c}^l_{t-1}; \boldsymbol{\theta}^l_{\mathrm{linear}})$
    if $\arg\max_{y_t} P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}})$
    $= \arg\max_{y_t} P^{l-1}(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}})$ **then**
      $p \leftarrow p + 1$
    else
      $p \leftarrow 0$
    end if
    if $p = \hat{p}$ **then**
      $y_t \leftarrow \arg\max_{y_t} P^l(y_t|y_{1:t-1}, \boldsymbol{X}; \boldsymbol{\Theta}_{\mathrm{E}})$
      break
    end if
  end for
end for

Note that due to the structure of the transformer self-attention mechanism, all previous hidden states $\boldsymbol{c}^{l-1}_{1:t-1}$ are essential to the calculation of $\boldsymbol{c}^l_{t-1}$. Therefore, when terminating inference occurred in l-th layer in the estimation of t-th token, hidden states $\{c^{l+1}_{t-1}, ..., c^L_{t-1}\}$ are complemented by copying c^l_{t-1}.

IV. Proposed method

This section details BRL, an effective method of training the early-exiting model. The key advantage of BRL is that it improves the ASR accuracy of specified low intermediate layers, while maintaining the ASR accuracy of high intermediate layers in the early-exiting model. BRL has two features:

- Fixing the parameters of high intermediate layers and training the parameters of low intermediate layers.
- Calculating gradients from not only low intermediate layers, but also high intermediate layers.

As shown in Figure 1, the initial parameter of BRL is the existing early-exiting model. In BRL, only the weights $\{\boldsymbol{\theta}^1_{\mathrm{dec}}, \ldots, \boldsymbol{\theta}^j_{\mathrm{dec}}, \boldsymbol{\theta}^1_{\mathrm{linear}}, \ldots, \boldsymbol{\theta}^j_{\mathrm{linear}}\}$ in $\boldsymbol{\Theta}_E$ are updated using Equation (8). $j \in [1, L]$ denotes the number of specified low intermediate layers to be trained.

TABLE I: Dataset details.

	Data size (hours)	Number of utterances	Number of characters
Train	512.6	413,240	13,349,780
Valid	1.9	1,292	47,970
Test	1.3	1,385	32,089

BRL sets uniform weights for w^l as follows:

$$w^l = \frac{1}{L}. \tag{10}$$

BRL sets uniform weights w^l because BRL focuses on improving the ASR accuracy of low intermediate layers, while maintaining the ASR accuracy of high intermediate layers.

V. Experiment

A. Dataset

We evaluated the effectiveness of BRL on Japanese discourse ASR tasks by using the corpus of spontaneous Japanese (CSJ) [24]. We used CSJ as the training set (Train), validation set (Valid), and test set (Test). Valid and Test are Eval2 and Eval3 provided by CSJ, respectively. Details of these datasets are given in Table I. We used 80 log mel-scale filterbank coefficients as acoustic features and text segmented into character units. To exclude speech data that was too long, we removed from the dataset any data that contained text with more than 300 tokens or more than $1,500$ audio frames.

B. Settings

The early-exiting model is a transformer-based encoder-decoder model with intermediate layers. We used two early-exiting models with different numbers of decoder layers: EE-base and EE-deep. EE-base has 8 encoder layers and 6 decoder layers. Every intermediate decoder layer in EE-base has a corresponding output layer. EE-deep has 8 encoder layers and 12 decoder layers, i.e., more decoder layers than in EE-base. EE-deep has an output layer for each even-numbered intermediate decoder layer. EE-base and EE-deep were created under the following conditions: the dimensions of the input and output continuous representations were set to 512, the dimensions of the inner outputs in the position-wise feed-forward networks were set to $2,048$, and the number of heads in the multi-head attention was set to 4. To stabilize training, the dropout rate in the transformer encoder and decoder layers was set to 0.1. For the activation function, we used swish activation layer [25].

BRL was applied to EE-base and EE-deep to yield BRL-base and BRL-deep. BRL-base was trained from the first through the third of its intermediate layers ($j = 3$) by utilizing BRL. Also, BRL-deep was trained from the first through the sixth of its intermediate layers ($j = 6$).

In the training, our models were optimized by using RAdam[26]. We set a mini-batch size of 64 and the learning rate of 0.0001. We used label smoothing [27] for regularization. For data augmentation, we applied SpecAugment with frequency masking and time masking [28], where the number of frequency masks and time-step masks were each set to 2, the frequency-masking width was randomly chosen from 0 to 27 frequency bins, and the time-masking width was randomly chosen from 0 to 100 frames. Early stopping was applied if no best model was found in the validation set after 10 epochs. We trained all models with one NVIDIA A6000 GPU.

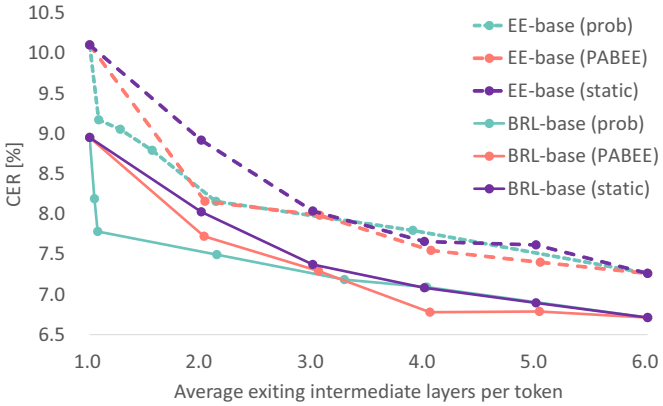

Fig. 2: CER and average exiting layer of models trained by BRL in EE-base.

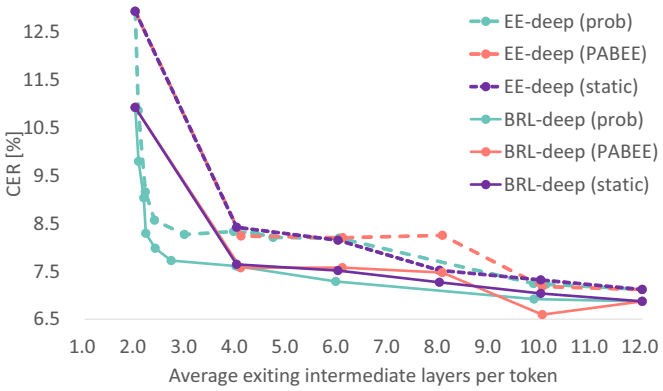

Fig. 3: CER and average exiting layer of models trained by BRL in EE-deep.

In the decoding, we used a beam search with a beam size of 4. In this experiment, we tested three exit criteria: probability-based (prob), patience-based (PABEE), and static (static). In prob, we set an optimal threshold λ^l that simultaneously minimized the character error rate (CER) and the average exiting intermediate layer per token in the validation set. To find the optimal threshold λ^l, we used NSGA-II [29], which is a multi-objective genetic algorithm for solving multi-objective optimization problems. We searched for the optimal λ^l in 300 epochs. In PABEE, we set a patience counter $\hat{p}$ from 1 to $L-1$. Static is the results of forced termination of the inference in each intermediate layer, which are presented to validate the training effect of BRL. We used the CPU, which was INTEL XEON Silver 4310, for the decoding and batch size was 1. The Intel oneAPI Math Kernel Library [30] was enabled for the computations, and 24 threads were used to perform parallel

TABLE II: The impact of early exit on actual inference speed.

	↓ CER[%]	↑ Speedup
EE-base (static by final output)	7.26	×1.00
EE-base (prob)	7.80	×1.38
	8.16	×2.27
	9.17	×3.30
BRL-base (prob)	7.09	×1.39
	7.49	×2.56
	7.78	×3.43

computations.

The accuracy of the early-exiting model was measured by CER. Reduction in inference time was measured by the average exiting intermediate layers per token.

C. Results

Figure 2 shows the trade-off between inference speed and CER in EE-base. From a comparison of EE-base (static) and BRL-base (static), we can find that BRL improved the ASR accuracy of the individual low intermediate layers. From a comparison of EE-base (prob) and BRL-base (prob), we can also find that BRL-base (prob) achieved a comparable CER to that of EE-base (prob) with fewer average exiting intermediate layers. Furthermore, we can find the same trend even when the exit criteria were changed to PABEE. Figure 3 shows the trade-off between inference speed and CER in EE-large. It confirms that BRL has a similar effect in EE-large as in EE-base. These results indicate that BRL can improve the ASR accuracy to train a low intermediate layer while maintaining the ASR accuracy of high intermediate layers.

To show the actual effect of BRL, we evaluated it in terms of computation speed and CER. Table II shows the impact of early exit on the actual inference speed. EE-base (static by final output) shows the inference speed and CER when always exiting at the final decoder layer. Speedup is a ratio based on EE-base (static by final output). The effect of speedup can be compared by Speedup under the same CER. By comparing EE-base (prob) and BRL-base (prob) in Table II, we can find that the same CER is achieved by Speedup of $\times 3.43$ in BRL-base (prob), while $\times 1.38$ in EE-base (prob). In other words, BRL-base (prob) achieves 2.49 times faster compared to EE-base (prob) at the same CER.

Further, we investigated the effect of changing the number of trained low intermediate layers in BRL. We adjusted j from 1 to 6 in BRL. Figure 4 shows the result. Even when the number of intermediate layers to be trained changed, BRL remained effective under all experimental conditions. $j = 6$ is the result of re-training all decoder layers including the final one. $j = 6$ and EE-base (prob) have different settings for loss weights. The loss weights of $j = 6$ obey Equation (9) and the loss weights of EE-base (prob) obey Equation (10). The result that $j = 6$ exceeds the ASR performance of EE-base (prob) indicate that BRL can also be used for the early-exiting model with insufficient training in the intermediate layers.

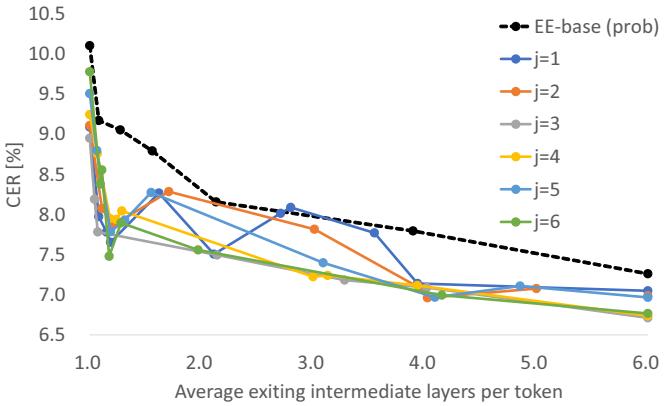

Fig. 4: All model used probability-based exit criteria. Result of varying the number of low intermediate layers. $j = 3$ indicates BRL-base (prob).

VI. Conclusion

We presented a block refinement learning (BRL) that achieves faster inference for the early-exiting model in automatic speech recognition (ASR). The key advantage of BRL is that it improves the ASR accuracy of low intermediate layers while maintaining the ASR accuracy of high intermediate layers in the existing early-exiting model. To this end, BRL trains the parameters of the low intermediate layers, while keeping the parameters of all other layers fixed. In addition, it exploits the gradients from not only the corresponding attached output layers, but also from all output layers including those associated with the high intermediate layers. We experimentally showed that BRL can improve the ASR accuracy of low intermediate layers while maintaining the ASR accuracy of high intermediate layers in the early-exiting model.

References

[1] M. A. Haidar, C. Xing, and M. Rezagholizadeh, "Transformer-based asr incorporating time-reduction layer and fine-tuning with self-knowledge distillation," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2021, pp. 2102–2106.

[2] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning*, 2015.

[3] L. Li, Y. Lin, D. Chen, *et al.*, "Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021, pp. 475–486.

[4] S. Kim, A. Gholami, Z. Yao, *et al.*, "Integer-only zero-shot quantization for efficient speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 4288–4292.

[5] S. Shen, Z. Dong, J. Ye, *et al.*, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 8815–8821.

[6] W. Zhang, L. Hou, Y. Yin, *et al.*, "TernaryBERT: Distillation-aware ultra-low bit BERT," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 509–521.

[7] C.-I. J. Lai, Y. Zhang, A. H. Liu, *et al.*, "Parp: Prune, adjust and re-prune for self-supervised speech recognition," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[8] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[9] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 1135–1143.

[10] R. Tang, K. Kumar, J. Xin, *et al.*, "Temporal early exiting for streaming speech commands recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7567–7571.

[11] D. Berrebbi, B. Yan, and S. Watanabe, "Avoid overthinking in self-supervised models for speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[12] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, Sep. 2020.

[13] J. Xin, R. Tang, Y. Yu, and J. Lin, "BERxiT: Early exiting for BERT with better fine-tuning and extension to regression," in *Porc. European Chapter of the Association for Computational Linguistics (EACL)*, 2021, pp. 91–104.

[14] G. A. Wright, U. Cappellazzo, S. Zaiem, *et al.*, "Training early-exit architectures for automatic speech recognition: Fine-tuning pre-trained models or training from scratch," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7567–7571.

[15] S. Bae, J. Ko, H. Song, and S.-Y. Yun, "Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp. 5910–5924.

[16] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.

[17] Y. Ji, J. Wang, J. Li, Q. Chen, W. Chen, and M. Zhang, "Early exit with disentangled representation and equiangular tight frame," in *Proc. Association for Computational Linguistics (ACL)*, 2023.

[18] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," in *Proc. Association for Computational Linguistics (ACL)*, 2020, pp. 2246–2251.

[19] S. Geng, P. Gao, Z. Fu, and Y. Zhang, "Romebert: Robust training of multi-exit bert," *arXiv preprint arXiv:2101.09755*, 2021.

[20] T. Schuster, A. Fisch, J. Gupta, *et al.*, "Confident adaptive language modeling," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022, pp. 17 456–17 472.

[21] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[22] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[23] Z. Liu, Z. Xu, H.-J. Wang, T. Darrell, and E. Shelhamer, "Anytime dense prediction with confidence adaptivity," in *Proc. International Conference on Learning Representations (ICLR)*, 2022.

[24] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," in *Proc. International Conference on Language Resources and Evaluation (LREC)*, 2000.

[25] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[26] L. Liu, H. Jiang, P. He, *et al.*, "On the variance of the adaptive learning rate and beyond," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[28] D. S. Park, W. Chan, Y. Zhang, *et al.*, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 2613–2617.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, pp. 182–197, 2002.

[30] M. Krainiuk, M. Goli, and V. R. Pascuzzi, "Oneapi open-source math library interface," in *Proc. International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, 2021, pp. 22–32.