IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 1, 2020, accepted December 17, 2020, date of publication December 28, 2020,
date of current version January 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3047799

Selective Fine-Tuning on a Classifier Ensemble: Realizing Adaptive Neural Networks With a Diversified Multi-Exit Architecture

KAZUTOSHI HIROSE[1], SHINYA TAKAMAEDA-YAMAZAKI[2], (Member, IEEE), JAEHOON YU[3], (Member, IEEE), AND MASATO MOTOMURA[3], (Senior Member, IEEE)

[1]School of Engineering, Tokyo Institute of Technology, Kanagawa 226-8502, Japan
[2]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo 113-8654, Japan
[3]Institute of Innovation Research, Tokyo Institute of Technology, Kanagawa 226-8502, Japan

Corresponding author: Kazutoshi Hirose (hirose.kazutoshi@artic.iir.titech.ac.jp)

This work was supported by JSPS KAKENHI Grant Numbers JP19J20473,JP18H05288.

ABSTRACT Adaptive neural networks that provide a trade-off between computing costs and inference performance can be a crucial solution for edge artificial intelligence (AI) computing where resource and energy consumption are significantly constrained. Edge AIs require a fine-tuning technique to achieve target accuracy with less computation for pre-trained models on the cloud. However, a multi-exit network, which realizes adaptive inference costs, requires significant training costs because it has many classifiers that need to be fine-tuned. In this study, we propose a novel fine-tuning method for an ensemble of classifiers that efficiently retrain the multi-exit network. The proposed fine-tuning method exploits individualities by assembling the output of the intermediate classifiers trained with distinct preprocessed data. The evaluation results show that the proposed method achieved 0.2%-5.8%, 0.2%-4.6% higher accuracy with only 77%-93%, 73%-84% training computation compared to the entire fine-tuning of classifiers on the pre-modified CIFAR-100 and Imagenet, respectively, although it depends on assumed edge environments.

INDEX TERMS Ensemble, fine-tuning, neural networks.

I. INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable breakthroughs in computer vision applications because of their outstanding recognition accuracy, while the edge devices still struggle to use them in practical situations. Therefore, for utilizing the benefit of DNNs in edge devices such as mobile phones, autonomous vehicles, and edge servers, the key challenges are computational cost and memory footprint. Because each edge device lies in various situations, achieving high accuracy with a single DNN for every scene requires massive operations and parameters.

To tackle this issue, we can use either one or both of the following approaches: i) Improving the efficiency of DNN processing. ii) Adjusting the computation cost of the DNN according to each situation. To improve the efficiency of DNNs, many researchers have proposed multiple approaches in the past decade: pruning [1], quantization [2], efficient convolution network [3], and network architecture search [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Wenming Cao.

Recently, for flexibly adjusting the trade-off between computation cost and inference performance, several studies have been presented using a model with multiple classifiers in the middle layers, which enable the model to terminate its inference process at the early stage [5]–[7]. This article mainly focuses on the latter.

The multi-exit networks mentioned above allow edge devices to select an affordable trade-off between the computation cost and accuracy of an inference process. This advantage can broadly extend the applications of sophisticated DNNs on edge devices. However, a model pre-trained on a cloud hardly shows its full potential in each device equally because of the difference between its training data and the data the device has to handle. The fine-tuning process can complement the gap and improve inference performance in edge environments that have rich diversity. The remaining problem is that the fine-tuning of a multi-exit network requires major computation costs compared to that of a single-exit network such as VGG [8] and ResNet [9] because all exits require retraining for maintaining the accuracy.

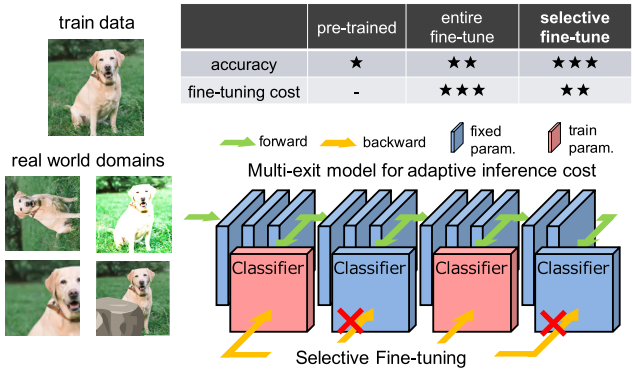

FIGURE 1. Selective fine-tuning on classifier ensemble.

This article proposes a novel fine-tuning method for multi-exit networks, which improves the inference performance of an ensemble of multi-exit classifiers with a lower computational requirement by using partial fine-tuning with differently preprocessed data for each classifier, as shown in Figure 1. Here, preprocessing is introduced to represent data distortions or variations that are expected to exist in edge environments. The major contributions of this article are summarized as follows:

1) We proposed a selective fine-tuning method for an ensemble of intermediate classifiers in the multi-exit neural network.
2) The evaluation results show that the proposed method achieved 0.2%-6.5%, 0.2%-4.6% higher accuracy with only 73%-93%, 61%-84% training computation compared with the entire fine-tuning of classifiers on the premodified CIFAR-100 and Imagenet, although it depends on assumed edge environments.
3) Furthermore, the proposed method's accuracy can surpass that of the base case even for original (i.e., not preprocessed) test data in a particular case for training a baseline model.

The rest of this article is organized as follows. Section II introduces related studies. Section III describes the proposed fine-tuning method and ensemble classifiers of a multi-exit network. Section IV shows the evaluation results of the proposed method using the CIFAR-10, CIFAR-100, and ImageNet datasets. Finally, Section V concludes this article.

II. RELATED WORK

This section describes three categories of related studies: efficient learning, adaptive inference, and ensemble learning. The efficient learning studies are co-operable approaches with the proposed method to improve training efficiency. The studies of adaptive inference are network architectures that enable a trade-off between computation cost and inference performance, including the multi-exit network, which is the base model used in this study. Finally, the studies of ensemble learning provide previous ensemble methods on multi-exit networks.

A. EFFICIENT LEARNING

[2], [10], [11] proposed quantization methods of gradient values to reduce computation complexity in backpropagation. In addition, [12]–[14] proposed a simpler training algorithm that removes complicated operations in the backpropagation, such as transposition of weights and differentiation of the activation function. These studies are useful for reducing the computation during training; however, it is not a solution for suppressing the linearly increasing computational cost of a multi-exit architecture, which is proportional to the number of intermediate classifiers. Thus, our selective fine-tuning method provides a direct solution to this problem. Combining our method with an efficient learning approach, we can further improve its efficiency.

B. ADAPTIVE INFERENCE

Several studies have proposed network architectures that adaptively adjust the computational cost of inference. [15], [16] proposed a structure to skip forward paths dynamically depending on input data. [5], [7], [17], [18] have multiple classifiers attached to the middle layers of their model, terminating inference at the early stage when necessary. The difficulty is that the computation cost of fine-tuning increases with the number of exits, which are also called intermediate classifiers. We aim to alleviate the burden of fine-tuning without an accuracy drop. Therefore, we designed and evaluated the selective fine-tuning method on the baseline model proposed in [5] and adopted distillation loss [19], [20] for the training process.

C. ENSEMBLE LEARNING

Ensemble techniques can improve the accuracy of neural networks [21]–[24]. Even if every model learned from the same dataset, the advantage gained by averaging the models allows us to obtain robust and stable solution points. Differences caused by random initialization, random mini-batch selection, hyperparameters, and implementation details generate the diversity of models composing the ensemble model. The ensemble of models is a reliable method for reducing a generalization error. The proposed method improves the diversity by retraining each intermediate classifier of a multi-exit architecture with a distinct preprocessed dataset.

III. SELECTIVE FINE-TUNING ON CLASSIFIER ENSEMBLE

We propose a selective fine-tuning method for an ensemble of multi-exit neural network models. To improve the accuracy of an ensemble, each source of the ensemble must be distinct. Our proposed method enables us to efficiently train the network based on the selective classifiers to be fine-tuned with distinct preprocessed training data to enhance the diversified classifiers.

A. NOTIONS

Let the training dataset on a cloud be $\mathcal{D}_T$ and the dataset on the inference environment be $\mathcal{D}_I$. The elements of each

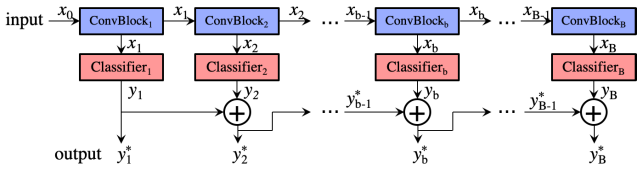

FIGURE 2. Ensemble multi-exit network: The blue blocks have pre-trained parameters, and the red blocks are intermediate classifiers retrained by a fine-tuning method. Ensemble outputs were obtained from these intermediate classifiers. We proposed a fine-tuning strategy to increase the generalizability of the classifier ensemble.

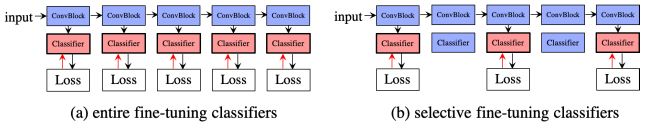

FIGURE 3. Two fine-tuning approaches in a classifier ensemble of a multi-exit network: Ensemble method is the same as Figure 2. In the entire fine-tuning process, all intermediate classifiers (represented as red blocks) are updated via a backpropagation process. In contrast, the parameters of the remaining layers (represented as blue blocks) were fixed. In selective fine-tuning, only selected intermediate classifiers are updated using dedicated preprocessed training data. Each intermediate classifier finally has a different characteristic according to the applied training data with distinct preprocessing.

dataset are defined as $z_T = (\boldsymbol{x}_T, y_T) \in \mathcal{D}_T$ and $z_I = (\boldsymbol{x}_I, y_I) \in \mathcal{D}_I$, which are pairs of input feature x and label y. In this study, we assume that a baseline model learned $\mathcal{D}_T$ on a cloud, and the fine-tuning process uses $\mathcal{D}_I$ an edge device. Environmental changes on the edge device can easily affect the inference performance. As it is challenging to gather datasets on the actual environment, we prepared $\mathcal{D}_I$ by applying several preprocesses to $\mathcal{D}_T$.

B. ENSEMBLE OF INTERMEDIATE CLASSIFIERS

We build the ensemble on a multi-exit neural network by accumulating the output value of each intermediate classifier, as shown in Figure 2. A multi-exit network comprises several convolution blocks (ConvBlocks), followed by a classifier. A ConvBlock can consist of a convolution layer, batch normalization layer, activation function, and pooling layer. We denote this as $ConvBlock_{\{1..b..B\}}$ on the network blocksize B. Let this output be $\boldsymbol{x}_b = ConvBlock_b(\boldsymbol{x}_{b-1})$. A classifier consists of a fully connected layer, in addition to the same elements as ConvBlocks. We represent the output of each classifier as $y_b = Classifier_b(\boldsymbol{x}_b)$. Finally, an ensemble of classifiers is obtained from the aggregation of multiple classifiers by a simple accumulation as

$$y_b^* = \frac{1}{b}\sum_{1}^{b} y_b. \tag{1}$$

Note that y_b is the value before applying the softmax function. As y_b^* can be computed in sequence from a former block, this ensemble method is a family of adaptive inference [7], [20].

C. ENTIRE FINE-TUNING METHOD

An entire fine-tuning process retrains all intermediate classifier parameters, as shown in Figure 3(a).

As $\mathcal{D}_T$ and $\mathcal{D}_I$ are distinct, it is necessary to recover the accuracy by retraining the intermediate classifier using $\mathcal{D}_I$. The goal of the entire fine-tuning process is to obtain the optimal parameter $\hat{\boldsymbol{\theta}}$ for $\mathcal{D}_I$ as

$$\hat{\boldsymbol{\theta}} = arg\ min_{\boldsymbol{\theta} \subset Classifier} \sum_{z_I \in \mathcal{D}_I} \mathcal{L}(z_I, \boldsymbol{\theta}). \tag{2}$$

The parameters are updated via the optimization problem for the entire fine-tuning.

D. SELECTIVE FINE-TUNING METHOD

We propose selective fine-tuning, which applies to only selected intermediate classifiers with dedicated training data with distinct preprocessing. This method aims to obtain higher accuracy of the classifier ensemble at low additional computation cost, as shown in Figure 3(b). Compared to the ordinary entire fine-tuning shown in Figure 3(a), the proposed selective fine-tuning picks one or more intermediate classifiers to be fine-tuned with data in $\mathcal{D}_I$, while the unselected classifiers are fixed. Each intermediate classifier trained using the dedicated preprocessed data can obtain different characteristics.

Let $\mathcal{T} \subset T = \{1, .., B\}$ be the index set of intermediate classifiers. The parameters of subset $\mathcal{T}$ are fine-tuned via the optimization problem as

$$\hat{\boldsymbol{\theta}} = arg\ min_{\boldsymbol{\theta} \subset Classifier_{\mathcal{T}}} \sum_{z_I \in \mathcal{D}_I} \mathcal{L}(z_I, \boldsymbol{\theta}). \tag{3}$$

Selective fine-tuning requires less computation cost than the entire fine-tuning as only the partial classifiers are updated.

We must choose the subset $\mathcal{T}$ carefully. We could identify the highest accuracy subset by fine-tuning all the classifiers and exploring all the combinations of classifiers to be fine-tuned. However, this approach does not reduce the number of training operations. Thus, we propose a method for predicting a subset to be fine-tuned to achieve high accuracy with a low computational cost. The method utilizes the training parameters of fine-tuning classifiers based on the experimental results that intermediate accuracy is closely related to the final one, as shown in Section IV-C2. The changes between $\mathcal{D}_T$ and $\mathcal{D}_I$ are similar across classes, although it depends on the preprocessing. Therefore, classifiers can learn the changes without data input from all classes; it is thus possible to predict the superior $\mathcal{T}$ in small step fine-tuning.

Let θ_0 be the parameter before fine-tuning, and θ_s the parameter after fine-tuning s steps on the preprocessed dataset $\mathcal{D}_I$. We can predict $\mathcal{T}$ to achieve high accuracy by computing y^* for all subsets as

$$y_{B,\mathcal{T}}^* = \sum_{b}^{\mathcal{T}} y_b^{\theta_s} + \sum_{b}^{\bar{\mathcal{T}}} y_b^{\theta_0}, \tag{4}$$

$$\hat{\mathcal{T}} = arg\ max_{t \in 2^T} Accuracy(y_{B,t}^*, y). \tag{5}$$

TABLE 1. Customized VGG 11 Architecture (3 blocks) for CIFAR-10 Dataset.

	Layer(-dim)	Kernel	Stride
ConvBlock1	conv-128	3	1
	conv-128	3	1
	maxpool	2	2
Classifier1	conv-128	3	2
	conv-128	3	2
	fc-10	-	-
ConvBlock2	conv-256	3	1
	conv-256	3	1
	maxpool	2	2
Classifier2	conv-128	3	2
	conv-128	3	1
	fc-10	-	-
ConvBlock3	conv-512	3	1
	conv-512	3	1
	maxpool	2	2
Classifier3	fc-1024	-	-
	fc-10	-	-

This method selects the subset $\mathcal{T}$ that has the highest intermediate accuracy of y^*. After determining the subset, $\mathcal{T}$ classifiers are fine-tuned, and the remaining $\bar{\mathcal{T}}$ classifiers are reset to θ_0 and fixed. In this article, we adopt single-epoch step size s. In general, if a subset continues to achieve the highest accuracy, you can select it.

The memory requirement for storing the parameter θ_0 of each intermediate classifier is small compared with the entire model. The computation amount of predicting the subset is also small compared with a single step of fine-tuning.

IV. EXPERIMENTS

We evaluate the validity of the proposed method on a multi-exit network. This section describes the basic settings for training and presents the performance of selective fine-tuning. We will show the effectiveness of exploiting the individuality of each classifier in a multi-exit network through the experimental results.

A. BASELINE MODEL TRAINING AND DATASET

The baseline models are a customized VGG-like model and two types of multi-scale dense networks (MSDNet) [5] implemented with PyTorch. We adopt MSDNet as it can achieve higher accuracy than the multi-exit ResNet described as [5].

1) CUSTOMIZED VGG MODEL

In the VGG architecture, we adopted an 11-layers model, which has three exits, as presented in Table 1. The training dataset is CIFAR-10 [25], which contains 50 000 training images and 10 000 test images. We used 500 training images as validation data. The training process adopts the standard data augmentation techniques described in [9]. For training the baseline model, we used the following constant hyperparameters: the mini-batch size of Momentum SGD is 256; the training epoch is 200; the learning rate starts from 0.1 and is reduced to 0.01 and 0.001 at 100 and 150 epochs, respectively. For fine-tuning, we use the same hyperparameters.

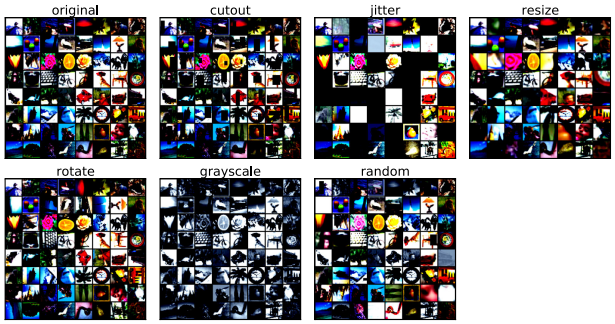

FIGURE 4. The preprocessed dataset of CIFAR-100.

2) MULTI-SCALE DENSE NETWORK

In MSDNet, we adopted the seven-exit and five-exit models for CIFAR-100 [26] and ImageNet [27], respectively. The network architectures are similar to the original model [5].

CIFAR-100 contains 50 000 training images and 10 000 test images. The training process adopts standard data augmentation techniques described in [9] and uses 500 of the training images as validation data. For the initial training of the baseline model, the loss function of the classifier at each exit includes both regular loss and distillation loss [19]. For the training baseline seven-exit model, we used Optuna [28] to optimize hyperparameters: learning rate, distillation loss rate, distillation temperature multiplier rate τ_*, and upper bound for the desired teacher confidence μ. For fine-tuning, the hyperparameters are constant as follows: The mini-batch size of Momentum SGD is 64 without distillation loss; the training epoch is 300; the learning rate starts from 0.1 and is reduced to 0.01 and 0.001 at 150 and 225 epochs, respectively.

ImageNet contains 1.2 million training images and 100 000 test images. The training process adopts the standard data augmentation techniques described in [9] and uses 50 000 of the training images as validation data. We used the pre-trained model [5] as a baseline five-exit model. For fine-tuning, the hyperparameters are constant as follows: the mini-batch size of Momentum SGD is 256, the training epoch is 90, the learning rate starts from 0.1, and is reduced to 0.01 and 0.001 at 45 and 67 epochs.

B. PREPROCESSING TYPES

Preprocessing was introduced to represent data distortions or variations that are expected to exist in edge environments. We applied preprocessing methods to the original training dataset $\mathcal{D}_T$ to obtain $\mathcal{D}_I$ mentioned in Section III. In this study, we examined six different preprocessing methods. Figure 4 shows an example of each preprocessing:

1) *Cutout* erases a random rectangle region of images. We assume a scene where the object is obscured by an obstacle.
2) *Jitter* randomly changes the brightness, contrast, and saturation of images. We assume a scene where the light intensity on the object changes.

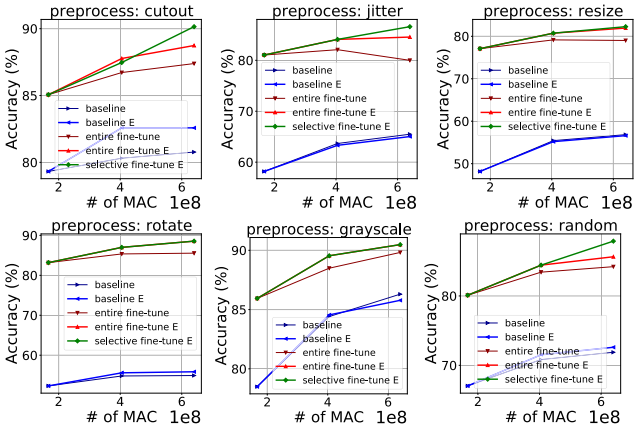

FIGURE 5. Top-1 accuracy on customized VGG11 (3 blocks) using the preprocessed CIFAR-10 test dataset: We fine-tuned the classifiers using the preprocessed training dataset. Each graph shows the inference accuracy(y-axis)-computation cost(x-axis) trade-off for the corresponding preprocess. Here, "E" indicates ensemble.

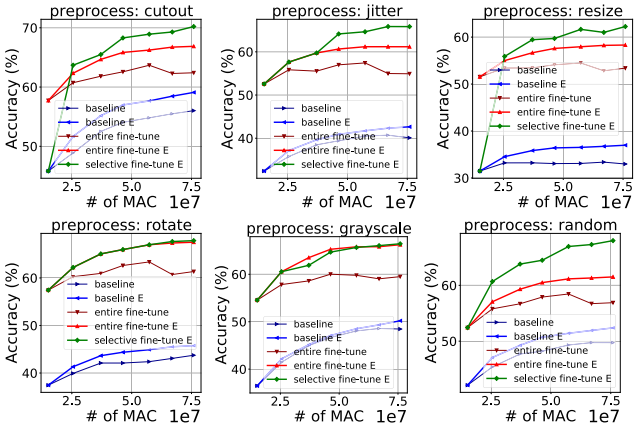

FIGURE 6. Top-1 accuracy on MSDNet (7 blocks) using the preprocessed CIFAR-100 test dataset: We fine-tuned the classifiers using the preprocessed training dataset. Each graph shows the inference accuracy(y-axis)-computation cost(x-axis) trade-off for the corresponding preprocess. Here, "E" indicates ensemble.

3) *Resize* crops images to random size and aspect ratio. We assume a scene where the distance between the object and the camera changes.
4) *Rotate* rotates images by an angle of 0, 90, 180, or 270 degrees. We assume a scene where the camera angle changes.
5) *Grayscale* converts images to monochrome. Although we do not assume a specific scene, we use grayscale as one of the various preprocessing steps.
6) *Random* selects a preprocessing listed above randomly.

C. SELECTIVE FINE-TUNING EVALUATION USING THE PREPROCESSED TEST DATA

We evaluate the ensemble of intermediate classifiers tuned by the entire fine-tuning described in Section III-C and the selective fine-tuning described in Section III-D. It fine-tunes the intermediate classifiers with the preprocessed training dataset and validates the preprocessed test dataset. We also compare its computation/accuracy trade-off with a baseline model.

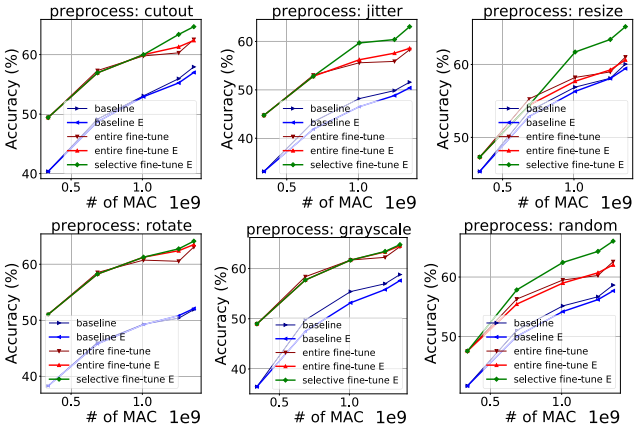

FIGURE 7. Top-1 accuracy on MSDNet (5 blocks) using the preprocessed ImageNet test dataset: We fine-tuned the classifiers using the preprocessed training dataset. Each graph shows the inference accuracy(y-axis)-computation cost(x-axis) trade-off for the corresponding preprocess. Here, "E" indicates ensemble.

1) ACCURACY OF EACH CLASSIFIER

Figures 5, 6, and 7 show the trade-off between inference computation and top-1 accuracy of each model against each preprocessed test dataset.

For CIFAR-10, selective fine-tuning outperforms the baseline model and the entire fine-tuning in cutout, jitter, resize, and random. However, in rotate and grayscale, the selective fine-tuning trains all classifiers; hence, its accuracy is the same as that of the entire fine-tuning. The small number of exits suggests that it is challenging to take advantage of the diversity with an ensemble.

For CIFAR-100 and ImageNet, the selective fine-tuning accuracy of the last block is higher than the entire fine-tuning accuracy in each preprocess. The results show that the selective fine-tuning method utilizes the diversity among the classifiers.

2) PREDICTION METHOD OF CLASSIFIERS TO BE FINE-TUNED

We evaluate the prediction method using CIFAR-100 and the MSDNet (7 blocks). Figure 8 shows the relevance between the accuracy after fine-tuning one step and the accuracy of having completed fine-tuning. The highest accuracy subset at step 1 is higher than the entire fine-tuning in cutout, jitter, resize, and random. We then present the accuracy of the following steps in jitter and grayscale as a sample of strong and weak predictions in Figure 9. This shows that we can predict a superior subset as the training steps progress and find a subset that achieves accuracy similar to the best accuracy within one epoch (= 762 steps). Therefore, we adopted s steps equal to one epoch considering the training cost.

TABLE 2. Selective Fine-Tuning Accuracy and Training Operations Relative to the Entire Fine-Tuning Using the Preprocessed Test Dataset: We Fine-Tuned the Classifiers Using the Preprocessed Training Dataset. Accuracy Refers to the Value of the Last Block.

Dataset	Preprocess	best subset		predict@epoch1	
		Accuracy(%)	Training Ops.(%)	Accuracy(%)	Training Ops.(%)
CIFAR-10	cutout	+0.98	-38	+0.67	-4
	jitter	+2.39	-38	+2.39	-38
	resize	+0.62	-38	+0.62	-38
	rotate	0.00	0	0.00	0
	grayscale	0.00	0	0.00	0
	random	+2.38	-38	+2.11	-4
CIFAR-100	cutout	+2.96	-21	+2.09	-39
	jitter	+4.62	-17	+4.09	-23
	resize	+4.22	-23	+3.21	-26
	rotate	+0.56	-7	+0.06	-14
	grayscale	+0.18	-7	-0.41	-17
	random	+5.79	-21	+4.05	-31
ImageNet	cutout	+2.30	-37	+2.24	-33
	jitter	+4.47	-33	+4.41	-37
	resize	+4.61	-34	+4.21	-61
	rotate	+0.52	-16	-1.09	-33
	grayscale	+0.23	-16	-0.52	-29
	random	+4.08	-33	+4.06	-34

TABLE 3. Best Accuracy of Selective Fine-Tuning in the Last Block Using the Original Test Dataset: We Fine-Tuned the Classifiers Using the Preprocessed Training Dataset. Here, "E" Indicates Ensemble. Δ is Calculated as *selective finetuning E acc.* − *max*(*baseline acc.*, *baseline E acc.*). The Patterns Consisting of B or F Show the Selection of Baseline or Fine-Tuning in the Blocks. From the Left, the First Letter Represents the First Block, the Second to Seventh Letters Represent the Second to Seventh Blocks.

Dataset	Model	Baseline acc.	Baseline E acc.	Selective fine-tuning of E acc.	Δ	Preprocessing	Pattern
CIFAR-10	customized VGG11 (3 blocks)	93.59	93.40	93.82	+0.42	jitter	FFB
CIFAR-100	MSDNet (7 blocks)	74.21	76.50	77.21	+2.22	jitter	BFBFFBB
ImageNet	MSDNet (5 blocks)	71.33	70.69	70.93	-0.40	rotate	BBFBB

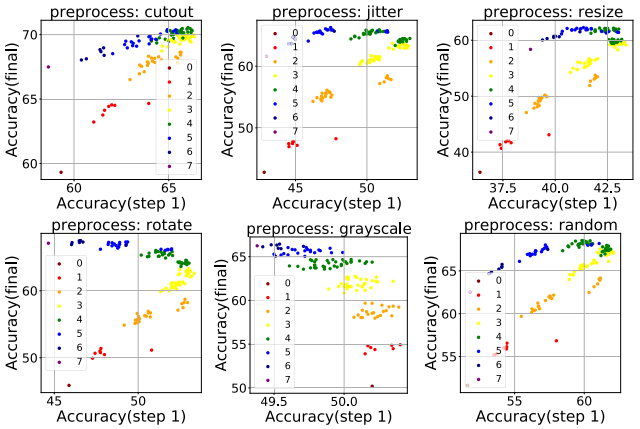

FIGURE 8. Top-1 accuracy of the last block on MSDNet (7 blocks) using the original CIFAR-100 test dataset: The X-axis indicates accuracy after fine-tuning one step, and the Y-axis indicates accuracy after having completed fine-tuning. The legend shows how many blocks have been fine-tuned. Here,"0" and "7" correspond to the baseline model and the entire fine-tuned model, respectively.

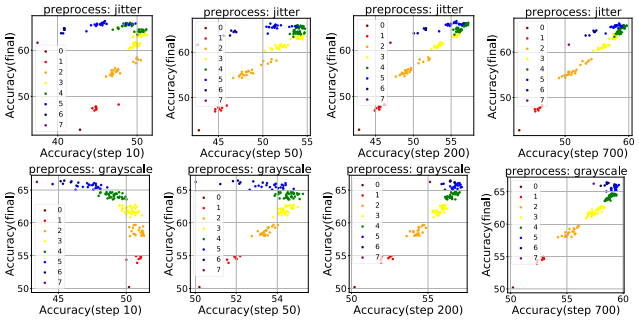

FIGURE 9. Top-1 accuracy on following steps of Figure 8:: The X-axis indicates accuracy after fine-tuning 10, 50, 200, and 700 step, and the Y-axis indicates accuracy after having completed fine-tuning. The legend shows how many blocks have been fine-tuned. Here,"0" and "7" correspond to the baseline and the entire fine-tuned models, respectively.

3) ACCURACY AND TRAINING OPERATIONS

Figures 10, 11, and 12 show the last classifier accuracy of all combinations to fine-tune classifiers and its training operations for fine-tuning. We summarized these results in Table 2. Based on this result, we can confirm the effectiveness of the selective fine-tuning method as it requires much less computation and achieves higher accuracy than the entire fine-tuning, except for the VGG model, which has a small number of intermediate classifiers. In addition, the prediction method achieves higher accuracy than the entire fine-tuning in most cases. Although some accuracies are below the entire fine-tuning, it achieves a higher accuracy than picking a subset at random. Note that training operations include feedforward, backward, and weight update.

D. SELECTIVE FINE-TUNING EVALUATION USING THE ORIGINAL TEST DATA

Thus far, we have evaluated the ensemble of intermediate classifiers fine-tuned with the preprocessed training dataset on the preprocessed test dataset. We do not know the inference performance on the original test dataset after fine-tuning

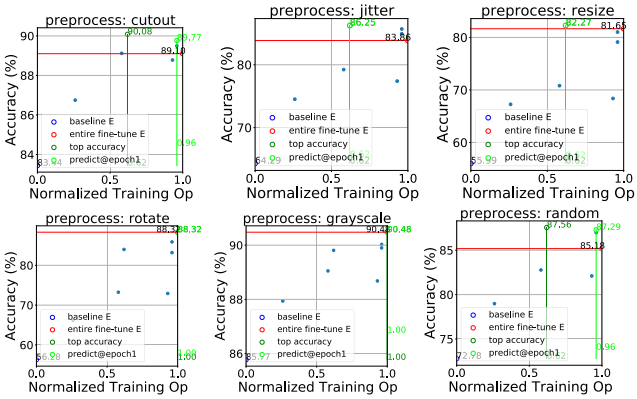

FIGURE 10. Top-1 accuracy of the last block and its training operations on customized VGG11 (3 blocks) using the preprocessed CIFAR-10 test dataset: Each graph shows the inference accuracy of the last block(y-axis)-training computation cost(x-axis) trade-off for the corresponding preprocess. A dot indicates a subset to be fine-tuned.

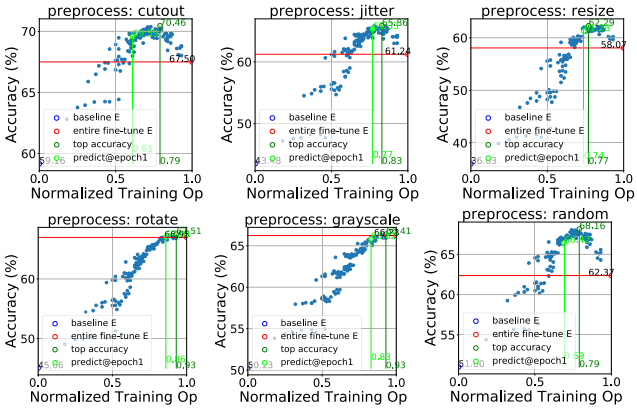

FIGURE 11. Top-1 accuracy of the last block and its training operations on MSDNet (7 blocks) using the preprocessed CIFAR-100 test dataset: Each graph shows the inference accuracy of the last block(y-axis)-training computation cost(x-axis) trade-off for the corresponding preprocess. A dot indicates a subset to be fine-tuned.

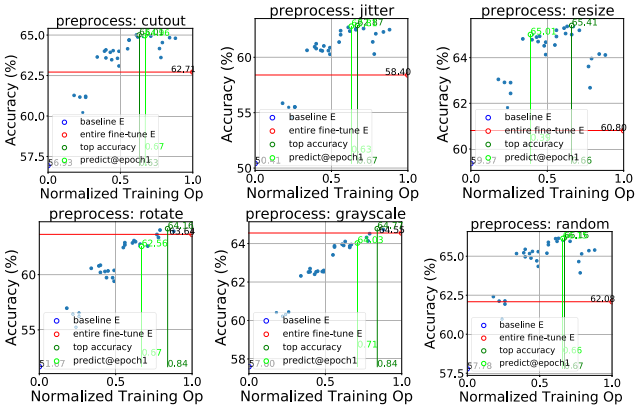

FIGURE 12. Top-1 accuracy of the last block and its training operations on MSDNet (5 blocks) using the preprocessed ImageNet test dataset: Each graph shows the inference accuracy of the last block(y-axis)-training computation cost(x-axis) trade-off for the corresponding preprocess. A dot indicates a subset to be fine-tuned.

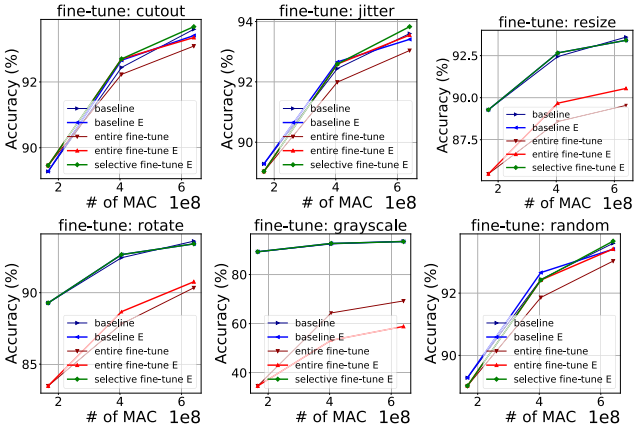

FIGURE 13. Top-1 accuracy on customized VGG11 (3 blocks) using the original CIFAR-10 test dataset: We fine-tuned the classifiers using the preprocessed training dataset.

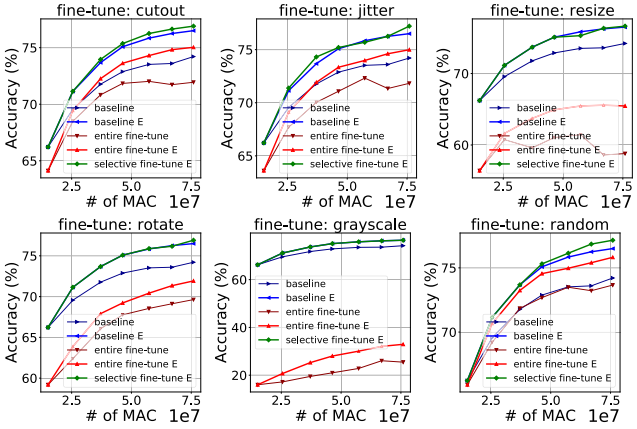

FIGURE 14. Top-1 accuracy on MSDNet (7 blocks) using the original CIFAR-100 test dataset: We fine-tuned the classifiers using the preprocessed training dataset.

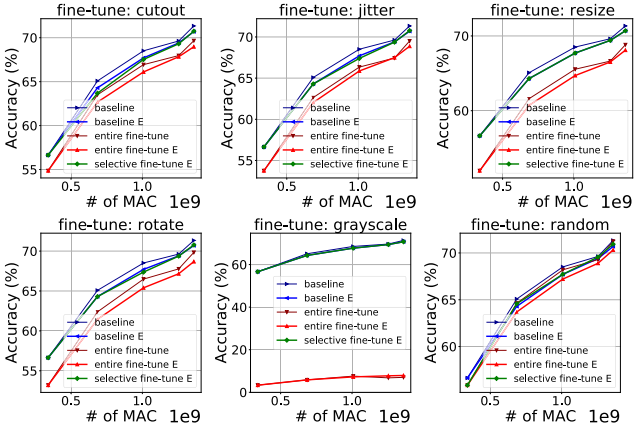

FIGURE 15. Top-1 accuracy on MSDNet (5 blocks) using the original ImageNet test dataset: We fine-tuned the classifiers using the preprocessed training dataset.

with the preprocessed training dataset. Figures 13, 14, and 15 answer to our question. It shows the inference accuracy of each method on the original test dataset, and Table 3 summarizes the best accuracies in the last block. Despite the difference between the original data and the preprocessed

data, the selective fine-tuning achieved a 2.2% higher accuracy than the accuracy of the baseline model on CIFAR-100. This experiment shows that the ensemble of classifiers trained with different preprocessing may be useful even for the original dataset in the case of baseline model training.

V. CONCLUSION

This article proposed a selective fine-tuning method for an ensemble of intermediate classifiers in a multi-exit neural network. The proposed method diversifies selected classifiers by fine-tuning on preprocessed datasets with different attributes from the original training dataset. With this approach, the proposed method exploits the individuality of each intermediate classifier of multi-exit neural networks. The experiment showed that the diversity intentionally generated by the proposed method improved the inference accuracy of multi-exit neural networks against the preprocessed test dataset. It also showed that the proposed selective fine-tuning achieved better accuracy with a smaller computation amount than the entire fine-tuning. Our future work will include determining appropriate preprocesses, and developing a lightweight model for actual edge environments.

REFERENCES

[1] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1. Cambridge, MA, USA: MIT Press, 2015, pp. 1135–1143.

[2] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, Feb. 2018, pp. 1–14.

[3] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Mach. Learn. Res.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA: PMLR, Jun. 2019, pp. 6105–6114.

[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: http://arxiv.org/abs/1611.01578

[5] G. Huang, D. Chen, T. Li, F. Wu, L. V. D. Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[6] A. Ruiz and J. Verbeek, "Adaptative inference cost with convolutional neural mixture models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1872–1881.

[7] A. Ruiz and J. Verbeek, "Distilled hierarchical neural ensembles with adaptive inference cost," Tech. Rep., 2020. [Online]. Available: https://hal.inria.fr/hal-02500660

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[10] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*. [Online]. Available: http://arxiv.org/abs/1606.06160

[11] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. New York, NY, USA: Curran Associates, 2017, pp. 1509–1519.

[12] M. Yazdani, "Linear backprop in non-linear networks," in *Proc. NIPS Workshop Compact Deep Neural Netw. Ind. Appl. (CDNNRIA)*, 2017, pp. 1–6.

[13] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. NIPS*, Red Hook, NY, USA. Curran Associates Inc, 2016, pp. 1045–1053.

[14] D. Han and H. J. Yoo, "Efficient convolutional neural network training with direct feedback alignment," 2019, *arXiv:1901.01986*. [Online]. Available: https://arxiv.org/abs/1901.01986

[15] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1790–1799.

[16] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and E. J. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 409–424.

[17] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2464–2469.

[18] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: International Convention Centre, Aug. 2017, pp. 527–536. PMLR.

[19] M. Phuong and C. Lampert, "Distillation-based training for multi-exit architectures," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1355–1364.

[20] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3712–3721.

[21] J. Xie, B. Xu, and Z. Chuang, "Horizontal and vertical ensemble with deep representation for classification," 2013, *arXiv:1306.2759*. [Online]. Available: http://arxiv.org/abs/1306.2759

[22] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why m heads are better than one: Training a diverse ensemble of deep networks," 2015, *arXiv:1511.06314*. [Online]. Available: http://arxiv.org/abs/1511.06314

[23] X. lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, Eds. New York, NY, USA: Curran Associates, Inc., 2018, pp. 7517–7527.

[24] J. Lee and S.-Y. Chung, "Robust training with ensemble consensus," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–15.

[25] A. Krizhevsky, V. Nair, and G. Hinton. (Sep. 2018). *CIFAR-10 (Canadian Institute for Advanced Research)*. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009, pp. 1 and 6.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[28] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.

KAZUTOSHI HIROSE received the B.E. and M.E. degrees in electronics from Hokkaido University, Japan, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Tokyo Institute of Technology. His current research interests include deep neural networks and its hardware-aware algorithms. He received the Young Researcher Award from the IPSJ Special Interest Group on System Architecture and the Best Student Presentation Award from the Technical Committee on Computer Systems of IEICE, Japan, in 2017, and the Research Fellowship for Young Scientists from JSPS, in 2019.

SHINYA TAKAMAEDA-YAMAZAKI (Member, IEEE) received the B.E., M.E., and D.E. degrees from the Tokyo Institute of Technology, Japan, in 2009, 2011, and 2014, respectively. From 2011 to 2014, he was a JSPS Research Fellow with DC1. From 2014 to 2016, he was also an Assistant Professor of the Nara Institute of Science and Technology, Japan. From 2016 to 2019, he was also an Associate Professor of Hokkaido University, Japan. Since 2018, he has been a Researcher of JST PRESTO. Since 2019, he has also been an Associate Professor of The University of Tokyo, Japan. His research interests include computer architecture, high-level synthesis, and machine learning acceleration. He is a member of IEICE, IPSJ, and JSAI.

JAEHOON YU (Member, IEEE) received the B.E. degree in electrical and electronic engineering and the M.S. degree in communications and computer engineering from Kyoto University, Kyoto, Japan, in 2005 and 2007, respectively, and the Ph.D. degree in information systems engineering from Osaka University, Osaka, Japan, in 2013. From 2013 to 2019, he was an Assistant Professor with Osaka University. He is currently an Associate Professor with the Tokyo Institute of Technology, Japan. His research interests include computer vision, machine learning, and system-level design. He is a member of IEICE and IPSJ.

MASATO MOTOMURA (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1985, 1987, and 1996, respectively. In 1987, he joined NEC Central Research Laboratories, Kawasaki, Japan, where he worked on various hardware architectures including string search engines, multi-threaded on-chip parallel processors, embedded DRAM-field-programmable gate array (FPGA) hybrid systems, memory-based processors, and reconfigurable systems. From 2001 to 2008, he was with NEC Electronics, Kawasaki, Japan, where he led research and business development of dynamically reconfigurable processor (DRP) that he invented. He was also a Visiting Researcher with the MIT Laboratory for Computer Science, Cambridge, MA, USA, from 1991 to 1992. Since 2011, he has been a Professor of Hokkaido University, Sapporo, Japan. Since 2019, he has also been a Professor of the Tokyo Institute of Technology, where he is also leading the Artificially Intelligent Computing (ArtIC) Research Unit. He is actively working on reconfigurable and parallel architectures for deep neural networks, machine learning, annealing machines, and intelligent computing in general. He is a member of IEICE, IPSJ, and EAJ. He was a recipient of the IEEE JSSC Annual Best Paper Award, in 1992, the IPSJ Annual Best Paper Award, in 1999, the IEICE Achievement Award, in 2011, and the ISSCC Silkroad Award as the Corresponding Author, in 2018.