IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 December 2023, accepted 29 December 2023, date of publication 8 January 2024, date of current version 17 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3351308

RESEARCH ARTICLE

Toward Improving Ensemble-Based Collaborative Inference at the Edge

SHUNGO KUMAZAWA, (Graduate Student Member, IEEE), JAEHOON YU, (Member, IEEE), KAZUSHI KAWAMURA, (Member, IEEE), THIEM VAN CHU, (Member, IEEE), AND MASATO MOTOMURA, (Fellow, IEEE)

Tokyo Institute of Technology, Midori-ku, Yokohama, Kanagawa 226-8502, Japan

Corresponding author: Shungo Kumazawa (kumazawa.shungo@artic.iir.titech.ac.jp)

This work was supported in part by JST CREST, Japan, under Grant JPMJCR18K3; and in part by KAKENHI, Japan, under Grant 23H05489 and Grant 22KJ1345.

ABSTRACT Ensemble-based collaborative inference systems, Edge Ensembles, are deep learning edge inference systems that enhance accuracy by aggregating predictions from models deployed on each device. They offer several advantages, including scalability based on task complexity and decentralized functionality without dependency on centralized servers. In general, ensemble methods effectively improve the accuracy of deep learning, and conventional research uses several model integration techniques for deep learning ensembles. Some of these existing integration methods are more effective than those used in previous Edge Ensembles. However, it remains uncertain whether these methods can be directly applied in the context of cooperative inference systems involving multiple edge devices. This study investigates the effectiveness of conventional model integration techniques, including cascade, weighted averaging, and test-time augmentation (TTA), when applied to Edge Ensembles to enhance their performance. Furthermore, we propose enhancements of these techniques tailored for Edge Ensembles. The cascade reduces the number of models required for inference but worsens latency by sequential inference processing. To address this latency issue, we propose m-parallel cascade, which adjusts the number of models processed simultaneously to m. We also propose learning TTA policies and weights for weighted averaging using ensemble prediction labels instead of ground truth labels. In the experiments, we verified the effectiveness of each technique for Edge Ensembles. The proposed m-parallel cascade achieved a 2.8 times reduction in latency compared to the conventional cascade, even with a 1.06 times increase in computational costs. Additionally, the ensemble label-based learning demonstrated comparable effectiveness to the approach using ground truth labels.

INDEX TERMS Ensemble, edge computing, collaborative inference, neural networks, cascade, test time augmentation.

I. INTRODUCTION

Edge-based inference of deep neural networks (DNNs) has garnered attention due to their ability to address challenges related to real-time performance, data privacy, and scalability, which are often associated with traditional cloud-based DNN inference [1]. In recent years, collaborative inference, involving the cooperative use of multiple edge nodes with individually limited computational resources to function as a larger computational resource, has been actively researched [2]. This approach overcomes the challenge of scarce computational resources in single nodes by leveraging the collective power of multiple nodes, resulting in more effective and accurate inference. This study focuses on the collaborative inference system among edge devices to efficiently process the expanding data volume resulting from the growing IoT ecosystem. The conventional collaborative inference architecture partitions a single DNN model and distributes its components across multiple edge devices [1], [2], [3]. While this architecture allows establishing large

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

networks on the edge, it has certain drawbacks, such as the difficulty in conducting inference on a single device, the strong dependency on availability of neighboring devices with required DNN partitions, and the need for complex control of repeated device-to-device communications [4].

To address these challenges, ensemble-based collaborative inference systems, Edge Ensembles, have been proposed [5], [6], [7], [8]. Edge Ensembles utilize ensemble methods to aggregate the inference results from respective models deployed on each edge device, achieving higher accuracy than single-device inference. This approach offers several distinct advantages that underscore its significance within the edge environment: 1) **Low dependence on other devices**: Edge Ensemble allows inference even when communication is disrupted because each device has a complete local model. 2) **Scalability**: The adaptability of Edge Ensembles allows for the adjustment of the number of devices involved based on the complexity of the task, enhancing scalability as demands fluctuate. 3) **Decentralization**: Operating within a decentralized system, Edge Ensembles reduce reliance on centralized servers, enhancing the robustness and reliability of edge-based inference. 4) **Resource Optimization**: Edge Ensembles make efficient use of computational resources by leveraging unused models on certain devices for inference, thus optimizing the utilization of available computing power. In particular, the last advantage aligns well with the explosive growth of IoT devices in the future. As IoT devices continue to increase, efficiently utilizing unused computational resources will become challenging. Edge ensembles offer a straightforward and efficient way to use these computational resources without the need for complex tasks such as network partitioning. Edge Ensembles share similarities with Federated Learning (FL) [9], regarding collaborative processing systems across multiple devices. However, it is important to note that Edge Ensemble is a collaborative inference system, whereas FL is a collaborative training system.

In previous Edge Ensembles, they have utilized conventional model integration methods such as majority voting [8], averaging the output of each model [5], [8], and weighted averaging based on the importance of each model [6], [7]. In addition to these methods, various effective model integration techniques have been used in previous DNN ensembles [10], [11], [12], [13]. However, since DNN ensembles require high computational costs due to the processing of multiple DNN models, most of these techniques were not originally designed for edge inference systems. It is uncertain whether these previous integration techniques can be directly applied in the context of Edge Ensembles, where there are constraints on computational resources and uncertainty regarding the availability of other devices' models due to limited communication bandwidth.

In this study, we examine the model integration part for improving Edge Ensembles. The model integration techniques commonly employed in DNN ensembles can be categorized into two approaches. The first approach involves handling the entire process, from training base learners to integrating them in inference. The second approach focuses on integrating the individual models already trained in each method. An example of the former approach is the Mixture of Experts (MoE) [14], which consists of task-specific expert models and a gating function that selects these experts based on the type of input. In recent years, the MoE models have achieved significant high accuracy in computer vision and natural language processing [15], [16], [17]. However, the former approaches, like MoE, which train base learners with the assumption of ensembling, do not meet the requirements of Edge Ensembles, where each base learner can conduct inference independently.

Therefore, this work considers improving the model integration in edge ensembles based on the latter approach and selects three integration techniques used in previous DNN ensembles: cascade, weighted averaging, and test-time augmentation (TTA). We apply these techniques to models designed for Edge Ensembles and evaluate their effectiveness. Additionally, we propose improvements and extensions to these methods for better alignment with Edge Ensemble requirements. Cascades are techniques that infer models in sequence and terminate the inference process midway, which is less computationally expensive but has higher latency than simple ensembles. We propose a method to improve the latency, a challenge for use at the edge. For TTA and weighted averaging, in addition to investigating the effects of these methods on Edge Ensembles, we propose learning TTA policies and the weights of weighted averaging using ensemble inference labels instead of ground truth labels. It is important to note that these methods focus on adapting the ensemble integration part during inference in edge environments with limited labeled data, rather than training the base learners themselves. The contributions of this paper are summarized as follows:

1) We analyze the effectiveness of Edge Ensembles with three integrated methods: cascade, TTA, and weighted averaging, in terms of accuracy, computational costs, and latency.
2) We propose advancements from existing methods, including the m-parallel cascade, which allows for the adjustment of latency and computational complexity while maintaining accuracy, and ensemble label-based learning, which learn ensemble weights and TTA policies without ground truth labels.
3) The m-parallel cascade achieve a reduction in computational costs and latency while maintaining accuracy compared to the conventional Edge Ensemble.

The rest of this paper is organized as follows. Section II describes Edge Ensembles and the ensemble methods used in this study. Section III describes the proposed methods, including m-parallel cascade and ensemble label-based learning. Section IV sets up the experiments and analyzes the hyperparameters of the proposed methods, and Section V presents experimental results and analysis. Finally, Section VI concludes this paper.

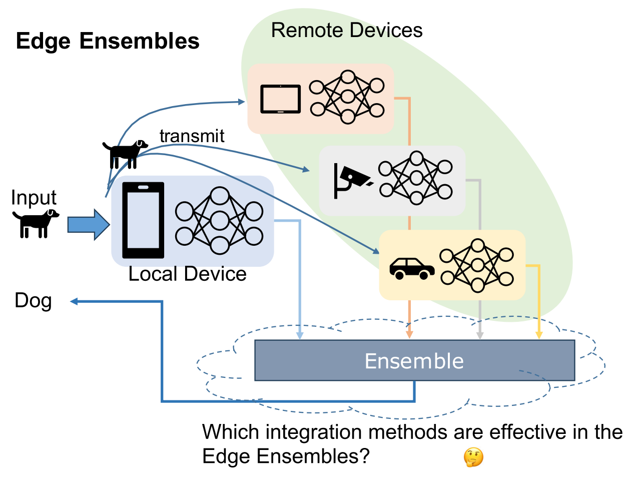

Edge Ensembles
Remote Devices
transmit
Input
Local Device
Dog
Ensemble
Which integration methods are effective in the Edge Ensembles? 🤔

FIGURE 1. An overview of Edge Ensembles. In this system, the local device collects data and sends it to nearby devices for inference. After inference on each device, the local device gathers and integrates all results, improving accuracy compared to a single prediction. This study examined model integration strategies to improve Edge Ensembles.

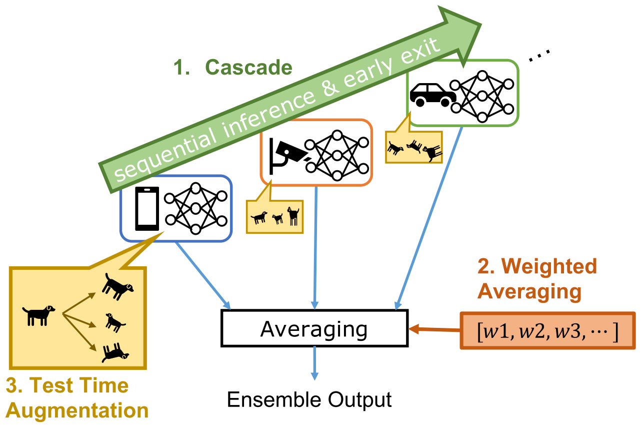


FIGURE 2. An overview of this study. This study investigates the effectiveness of three existing ensemble methods, namely 1. cascade, 2. weighted averaging, and 3. TTA as improvement techniques for Edge Ensembles. We examine their impact on Edge Ensembles and proposed enhancement methods specifically tailored for Edge Ensembles.

II. RELATED WORK

In this section, we first explain ensemble-based collaborative inference, Edge Ensembles, which is the focus of this study. Then, we describe conventional model integration techniques for improving ensemble methods and discuss how to use them in the Edge Ensembles context.

A. EDGE ENSEMBLES

Edge Ensembles are collaborative inference systems that improve accuracy by aggregating individual models' predictions on each edge device. FIGURE 1 represents an overview of the system. The figure illustrates the collaborative inference flow in Edge Ensembles. In this process, the local device shares the collected data with other nearby devices. Each device conducts inference using the shared data and sends the results back to the local device. Finally, the local device aggregates all the inference results, including its own output.

Edge Ensembles offer several advantages in the following aspects: they enable standalone inference using a local model, provide scalability by adjusting the number of devices involved based on task complexity, and operate as decentralized systems without dependency on a central server. In Edge Ensembles, communication with other devices incurs additional latency. However, the advantage of ensemble methods, which allow parallel processing of each device, enables significantly lower latency than sequential approaches that involve splitting a large model and running it on multiple devices. Furthermore, Malka et al. [6] proposed quantizing the test data before transmitting it to mitigate network overhead. However, if there are slower devices in the ensemble group, it can significantly reduce the overall throughput. Therefore, Feng et al. [7] proposed network search algorithms and knowledge distillation-based training methods in an Edge Ensemble composed of heterogeneous devices with performance disparities. These approaches aim to ensure that each device meets the constrained inference processing time. In addition, privacy concerns arise in Edge Ensembles due to data transmission to other devices. Yilmaz et al. [8] proposed the security-enhanced method where each device adds noise to the inference results to prevent leakage of information about the model on that device. This approach is similar to FL, where models train collaboratively while keeping their individual information confidential. Similarly, in this method, models collaboratively conduct inference while preserving the confidentiality of their information.

This study focuses on the integration of models for improving Edge Ensembles. In previous Edge Ensembles, inference results from each model are aggregated by majority voting or averaging over noisy outputs [8], averaging the output of each model [5], or weighted averaging based on the F1 score [7] or accuracy [6] as the importance of each model. The following section explores other model integration methods that can further enhance Edge Ensembles.

B. ENSEMBLE INTEGRATION METHODS

There are various existing ensemble methods. Traditional ensemble methods include learning strategies of base learners, such as bagging and boosting, as well as integration techniques for combining the outputs of base learners, like majority voting and weighted averaging [10], [11]. There are also methods covering the entire process, from learning to integrating base learners, such as MoE [14]. In the context of DNNs, implicit ensemble techniques such as dropout [18] and stochastic depth [19] exist. In this study, we focus on integrating individually trained models using standard training methods without relying on ensemble learning strategies to preserve the advantage of Edge Ensembles, which allows single inference even when communication with another device is lost. We consider methods of integrating models to improve Edge Ensembles from the following three approaches:

1) INTEGRATION METHOD WITH HIGH EFFICIENCY

One of the primary techniques for efficient ensemble integration is ensemble selection [10], [12], [20]. Ensemble selection methods involve selectively reducing the number of base learners used in inference. However, in Edge Ensembles, ensemble selection methods may not always be feasible or suitable due to the potential unavailability of the desired models or the concentration of access on specific devices with outstanding models. On the other hand, cascade is a method that does not involve base learner selection but dynamically terminates inference based on the confidence of the inference results. As a result, the cascade can be regarded as one of the approaches for dynamic ensemble selection. This study employs cascade as an integration method to enhance efficiency because of its simplicity. It can dynamically reduce the number of models used in inference just by setting the inference terminating threshold in advance.

2) INTEGRATION METHOD TO IMPROVE ACCURACY

Integrating each output with weighted averaging, also used in previous Edge Ensembles, is generally effective in terms of accuracy. Comparing various integration techniques, such as majority voting and unweighted averaging in ensembling convolutional neural networks (CNNs), the weighted averaging method based on the Super Learner (SL) achieves the highest accuracy [13]. This study examines weighting methods other than those used in previous Edge Ensembles to improve accuracy.

3) INTEGRATION OF BASE LEARNERS WITH IMPROVED ACCURACY

In general, improving the accuracy of each base learner leads to an enhancement in the accuracy of ensembles. This study employs TTA as a method to improve the accuracy of a single base learner at inference. TTA is an ensemble method conducted on a single model, which aggregates the inference results for each augmented test data.

FIGURE 2 represents the integration techniques used in this study to improve Edge Ensembles: cascade, weighted averaging, and TTA. The figure indicates the parts of the ensemble where each method is applied. The following sections describe each method.

a: CASCADES

Cascades are techniques that reduce computational complexity by sequentially processing the inferences of base learners and terminating the process at intermediate stages. The original cascade [21] does not aggregate the inference results for each stage, while this study employs a technique similar to Soft Cascade [22], which propagates information to the next stage, to use in the context of ensemble. FIGURE 3 compares the inference process by the cascade and simple ensemble.

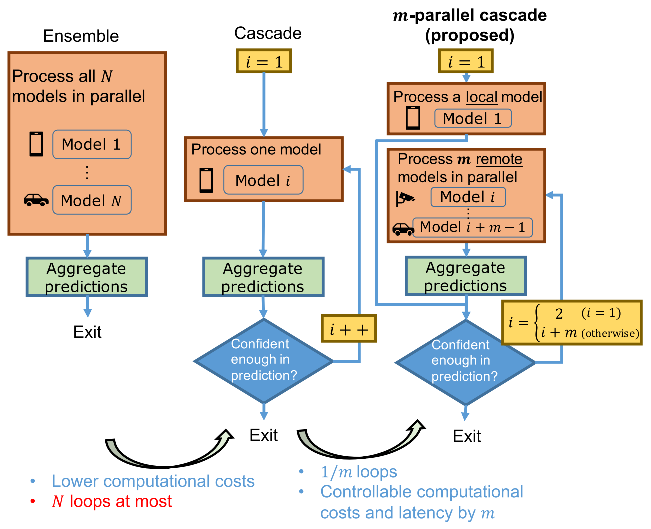

Ensemble
Process all N models in parallel
Model 1
Model N
Aggregate predictions
Exit
Cascade
$i = 1$
Process one model
Model i
Aggregate predictions
$i++$
Confident enough in prediction?
Exit
m-parallel cascade (proposed)
$i = 1$
Process a local model
Model 1
Process m remote models in parallel
Model i
Model $i+m-1$
Aggregate predictions
$i = \begin{cases} 2 & (i=1) \\ i+m & \text{(otherwise)} \end{cases}$
Confident enough in prediction?
Exit
Lower computational costs
N loops at most
$1/m$ loops
Controllable computational costs and latency by m

FIGURE 3. Flows of inference for three methods: simple ensemble (left), cascade (middle), and *m*-parallel cascade (right, proposed method). The cascade reduces computational costs by conducting sequential inferences and stopping midway, but it cannot parallelize each inference processing, which may lead to increased latency. The proposed method, *m*-parallel cascade, mitigates latency increase by parallelizing inference for *m* models.

The left side of the figure represents the ensemble, which conducts inferences of all base learners simultaneously and aggregates their results. The center of the figure represents the cascade, which sequentially processes the inference of each base learner and aggregates their results. If the cascade determines that the aggregated result exceeds a certain confidence threshold, it terminates the inference at that point. Compared to the ensemble, the cascade allows for inference with fewer models, especially simpler data. As a result, the average computational complexity decreases.

Cascades are techniques that have been used since before the advancements of DNNs. In the context of DNNs, the cascade has been applied to multiple EfficientNet models in recent years, resulting in a 5.5x speedup for online processing at 5.4x fewer FLOPs while maintaining accuracy [23]. The cascade in Edge Ensembles decreases the number of models required, thereby increasing the number of unused devices. By utilizing these unused devices in another ensemble group, the overall average accuracy of the system can be improved as a result. However, there is a concern about latency degradation when inferring models sequentially. In this study, in addition to evaluating conventional cascade, we propose a method to mitigate cascade latency.

b: WEIGHTED AVERAGING

The weighted averaging assigns weights to each base learner when averaging their output class probabilities. While unweighted averaging is a reasonable approach when the performance of each DNN model is roughly the same, it is not optimal when the base learners consist of heterogeneous models [11]. There are several existing weighting methods for base learners including approaches based on Bayesian theory [24], [25], [26], the performance of each base learner [6], [7], [27], [28], and the utilization of a meta-learner [29], [30].

This study employs two approaches for Edge Ensembles. The first approach utilizes SL, a meta-learning technique, which has demonstrated superior accuracy in CNN ensemble [13]. The second approach exploits the performance of each base learner, inspired by [7], the conventional Edge Ensemble that uses F1 scores as a performance measure. The following paragraphs explain the weighting methods based on SL and F1 scores.

I) SUPER LEARNER-BASED WEIGHTING

SL [30] is a meta-learning technique that learns the weighted combination of each base learner based on their outputs. SL learns the optimal weights for each base learner via V-fold cross-validation. For the classification in DNN ensembles, the loss function of SL defined in [13] is represented as follows:

$$Loss_{SL}(\vec{w}) = \sum_{v=1}^{V} \sum_{i \in val(v)} l\left(y_i, \sum_{j=1}^{N} w_j \vec{z_{i,j}}\right), \tag{1}$$

where N is the number of base learners, $\vec{w} = [w_1, w_2, \ldots w_N]$ is the weight vector, y_i is the class of the i-th data, $val(v)$ is the set of indices of the data in the v-th fold, $\vec{z_{i,j}}$ is the output vector before softmax from the j-th base learner of the i-th data, and l is the classification loss such as cross-entropy loss. SL learns the weight vector that minimizes the loss. In the inference phase, SL gets the class for the i-th data by

$$\text{class}_i = \arg\max_k \sum_{j=1}^{N} w_j z_{i,j,k}. \tag{2}$$

where $z_{i,j,k}$ is the output value of k-th class from the j-th base learner for the i-th data.

Performing cross-validation multiple times and constructing models at each iteration can be time-consuming. However, it has been shown that SL is effective in finding effective weights even when using a single validation set [13], [31]. This study applied the same method in [13] to model constructions assuming Edge Ensembles.

II) F1 SCORE-BASED WEIGHTING

Previous Edge Ensembles utilize the F1 score-based weighting method [7]. This method uses confidence scores for each class of each model as the F1 score:

$$c_{j,k} = \frac{2\,\text{Precision}_{j,k}\,\text{Recall}_{j,k}}{\text{Precision}_{j,k} + \text{Recall}_{j,k}}, \tag{3}$$

where $c_{j,k}$ is the confidence score of the j-th base learner for the k-th class, and $\text{Precision}_{j,k}$ and $\text{Recall}_{j,k}$ are the precision and recall of the k-th class on the j-th base learner for the validation set. Then, the prediction class is calculated by

$$\text{class}_i = \arg\max_k \sum_{j=1}^{N} c_{j,k}\,\sigma\left(\vec{z_{i,j}}\right)_k, \tag{4}$$

where σ represents softmax function, and thus $\sigma\left(\vec{z_{i,j}}\right)_k$ means the output probability of k-th class from the j-th base learner for the i-th data.

This approach is effective when there is a bias in performance among classes within a model. However, it may cause overfitting if the number of validation sets is insufficient or if the number of classes is too large. For example, when using the CIFAR-100 dataset, which has 100 classes, and choosing 5,000 images as the validation set, which is half of the test set, there are only 50 images per class. This is not sufficient to determine the performance for each class. In this study, taking inspiration from this approach, we propose a weighting method based on the accuracy of whole classes in a model as the general-purpose method not limited to data sets.

c: TEST TIME AUGMENTATION

TTA is a technique applying data augmentation to the test data during inference and aggregating inference results for each augmented data to improve accuracy. TTA can be considered as an ensemble of inference data in a single model. Although less significant than the accuracy improvement of TTA on a single model, executing TTA on multiple models and aggregating all their results improves accuracy compared to ensembles without TTA [32]. In this research, we investigate the utility of TTA in Edge Ensembles where the available number of devices varies.

Additionally, there are researches focused on finding effective TTA transformation policies [33], [34], [35]. One approach is the Greedy Policy Search (GPS) [33], which employs a greedy search strategy. Other methods involve learning policy weights through gradient descent [34] or using a separate network for policy predictions [35].

This study uses GPS, which is the least computationally expensive, to explore the TTA policy for each model and then investigate whether edge ensembles using TTA are effective.

III. PROPOSED METHODS

This study examines the efficacy of three existing ensemble methods, namely cascade, TTA, and weighted averaging, for Edge Ensembles. Additionally, we propose enhancements and extensions to these methods specifically tailored for Edge Ensembles. This section describes these proposed methods. We propose m-parallel cascade with tunable latency and computational complexity for the cascade. For TTA and weighted averaging, we offer learning methods using ensemble prediction labels, i.e., labels from final ensemble results, instead of ground truth labels. These methods aim to adjust the integration part in real-world field scenarios where obtaining labeled data is challenging, such as in edge environments. We also propose a weighting method based on the accuracy of each base learner for weighted averaging. In the following sections, we describe each of the proposed methods.

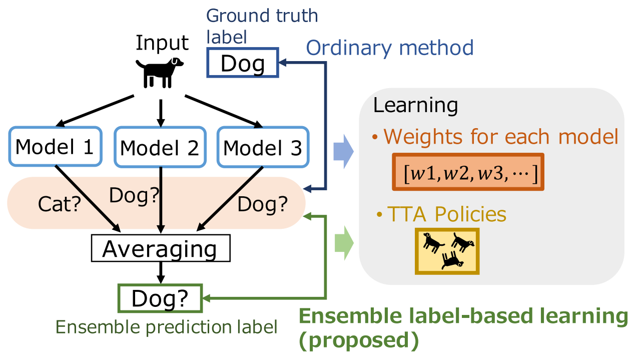


FIGURE 4. **Learning methods for weighted averaging weights and TTA policy. In the conventional approach, each learning process uses ground truth labels, whereas the proposed method utilizes labels from ensemble predictions. We devised this learning approach for edge environments where getting ground truth labels is challenging.**

A. *m*-PARALLEL CASCADE

The cascade reduces the average number of models in inference compared to simple ensembles but requires sequential inference processing. It reduces computational complexity and latency compared to the ensemble in environments that can only process models sequentially, such as a single server. However, in Edge Ensembles, where multiple devices can execute in parallel, the cascade's latency is higher than the ensemble's. The increased latency can be problematic since edge inference systems often require response time.

Therefore, we propose m-parallel cascade, which mitigates latency for Edge Ensembles. The right side of FIGURE 3 shows the flow of the m-parallel cascade. The m-parallel cascade processes multiple m models on remote devices simultaneously, whereas original cascades process one model at a time. This method is just between ensemble and cascade. Note that we don't parallel the inference in the local device, which is the first iteration of the cascade process. This is because cascades can terminate the inference process with only fast local inference for simple data, eliminating latency in data transfer. Since m-parallel cascade infer m models at once, the average number of models used increases, but the latency decreases compared to the original cascade. By adjusting the parallelism hyperparameter m, it is possible to reduce the number of models in inference while satisfying the required response time. In Section IV, we analyze the accuracy, computational complexity, and latency for varying m.

B. ENSEMBLE LABEL-BASED LEARNING

This section introduces ensemble label-based learning for TTA policies and weights in weighted averaging. This approach is specifically designed for Edge Ensembles, collaborative inference systems deployed in edge environments where obtaining labeled data is challenging. Instead of relying on ground truth labels, the method utilizes ensemble prediction labels for adapting the model integration part. The inspiration for this method comes from *Dynamic Deep Ensemble Management* used in [7], which adjusts the participation probability of each device in the inference based on its contribution, as measured by comparing each device's inference result with the final ensemble result. FIGURE 4 compares the usual method, which uses ground truth labels, and the proposed method, which uses ensemble prediction labels. While the usual learning method is based on the loss computed with the ground truth labels of the training data and the predictions of each base learner, the proposed method uses ensemble prediction labels instead of ground truth labels. The following two paragraphs detail learning TTA policies and weighting for each base learner with ensemble prediction labels.

1) GPS WITH ENSEMBLE PREDICTION LABELS

This study uses GPS, a greedy algorithm-based lightweight technique, as a policy search method for TTA. The proposed method uses the prediction labels from ensemble results without TTA instead of the ground truth labels as the training data in GPS. Each base learner learns a policy that makes its inference result closer to the ensemble prediction result, which would improve the accuracy of a single model. On the other hand, since this method relies on ensemble results, it is uncertain whether ensembling all results from each base learner using TTA based on this approach would actually lead to an improvement in accuracy. In Section V, we analyze the effectiveness of this method through experiments.

2) WEIGHTING METHODS WITH ENSEMBLE PREDICTION LABELS

As in the GPS case above, this method uses ensemble prediction labels instead of ground truth labels for learning weights in weighted averaging. Weighting methods based on each base learner's performance, such as F1 score or accuracy, can dynamically update the weights during inference by comparing each base learner's prediction with ensemble prediction labels. However, it is uncertain whether learning weights using ensemble prediction labels leads to effective weighting. Therefore, we experimentally validate the effectiveness of this approach.

C. ACCURACY-BASED WEIGHTING METHOD

This study employs two types of weighting methods, one based on SL and the other based on the performance of each base learner. The former method follows the same approach as [13]. In the latter method, previous research [7] had employed class-specific F1-scores for the performance metric as indicated in Equation 4. However, there is a concern that this approach may not function effectively when there is insufficient data to calculate F1 scores for each class. FIGURE 9 compares the accuracy of the CIFAR-100 dataset between scenarios with no weighting, weighting based on class-specific F1 scores, and weighting based on the macro average of F1 scores across all classes. When using class-specific F1 score-based weights, the accuracy decreased compared to not using weights. However, when

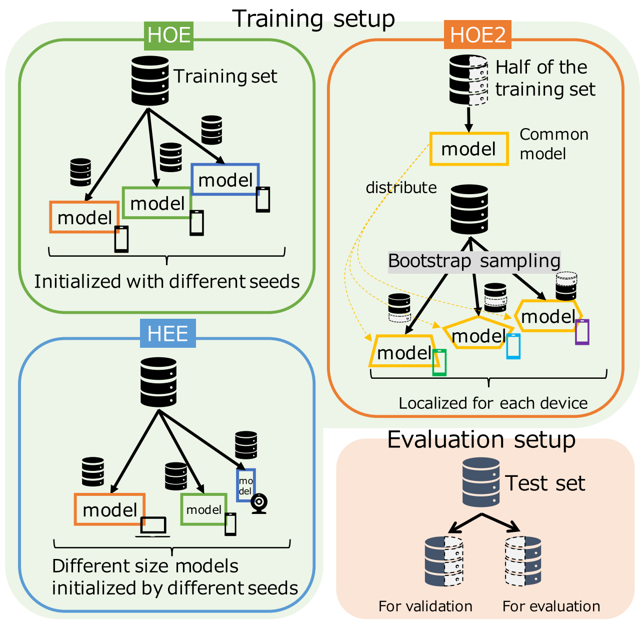


FIGURE 5. Schematic diagram of the experimental setup. HOE assumes each device has its own separately generated models, and HOE2 assumes a common model is localized on each device. HEE takes into account the varying model size requirements of each device. In the evaluation, we divide the test set into two halves, one for validation and the other for evaluation.

using the macro-averaged F1 score for weighting, accuracy was slightly improved compared to not using weights. These results suggest that overfitting of the weights occurred when calculating class-specific F1 scores.

In this study, we use accuracy-based (ACC-based) weighting, which simplifies the implementation without significantly altering the functionality compared to using average class-specific F1 scores for weighting. Furthermore, we do not use accuracy as weights directly to improve performance but rather employ a function to get the appropriate weights. We define the weight w_i for the i-th base learner as an ACC-based method using softmax with temperature as follows:

$$w_i = \frac{\exp\left(\frac{acc_i}{T}\right)}{\sum_{j=1}^{N} \exp\left(\frac{acc_j}{T}\right)}, \tag{5}$$

where acc_i is the accuracy of the i-th base learner, and T is the temperature in softmax. We adjust the softmax temperature T to obtain the optimal weights for each Edge Ensemble model configuration. In Section IV, we examine the accuracy for varying T.

IV. SETUP AND PRELIMINARY EXPERIMENTS

In preparation for the evaluation in Section V, this section first sets up the experiments and then analyzes the proposed method's hyperparameters.

A. SETUP

In this study, we focus on how to combine models in Edge Ensembles rather than training individual models. We prepare the sets of models assuming Edge Ensembles in advance and evaluate the inference performance for each integration methods. In this section, we first present the model configurations designed for Edge Ensembles and the datasets used for evaluation. Then, we describe the settings of the ensemble methods used in this study.

1) MODELS AND DATASETS

We consider three model configurations of Edge Ensembles: 1. Homogeneous Ensemble (HOE), 2. HOE2, and 3. Heterogeneous Ensemble (HEE). FIGURE 5 shows the schematic of the training strategies for each model configuration. The HOE setting consists of 16 ResNet18 models, each of which trains with the entire training data from different initial weights generated by each seed. The HOE2 setting also consists of 16 ResNet18 models. However, each model is based on a single ResNet18 trained for half of the total epochs using half of the training data. Then, each model trains for the remaining half of the total epochs using bootstrap-sampled data from a whole training set. This configuration assumes a scenario where a common model trained on the server is localized on the edge. The HEE setting consists of lightweight variants of ResNet18 models based on five different configurations. We reduce the channel sizes of each layer in ResNet18 by multiplying a parameter x, which is a reduction factor relative to the original channel size. We set x to 0.2, 0.4, 0.6, 0.8, and 1.0 (original ResNet18) and create 2, 3, 6, 3, and 2 models for each x, resulting in 16 models. Each model trains using the entire training data from different initial weights. This model configuration accounts for the scenario where the Edge Ensemble comprises edge devices with varying model size requirements.

For evaluation, we utilize CINIC-10, CIFAR-100, and Tiny ImageNet datasets. CINIC-10 is a 10-class dataset that combines samples from CIFAR-10 and ImageNet. Tiny ImageNet is a 200-class dataset that contains extracted data from ImageNet. For all datasets, we train each model for 100 epochs. The optimizer is SGD with a momentum of 0.9. A weight decay of 5×10^{-4} is applied. The learning rate is set to 0.1 with a cosine annealing schedule. We train each model using all the training sets and divide the test sets in half, using one half for evaluation and the other half as a validation set. The validation set is used to adjust various hyperparameters and learn TTA policies and weights for weighted averaging.

2) SETUP OF ENSEMBLE METHODS

This paper uses the simple ensemble with unweighted averaging for each base learner's output probability as the baseline for evaluation, and we conveniently call this method the vanilla ensemble. Following paragraphs describe the implementation and parameter settings in ensembles with the cascade, TTA, and weighted averaging used in this study.

For the cascade, it is necessary to set up a confidence measure function and an inference termination threshold. We use the confidence measure based on the maximum probability in the prediction, whose effectiveness has been

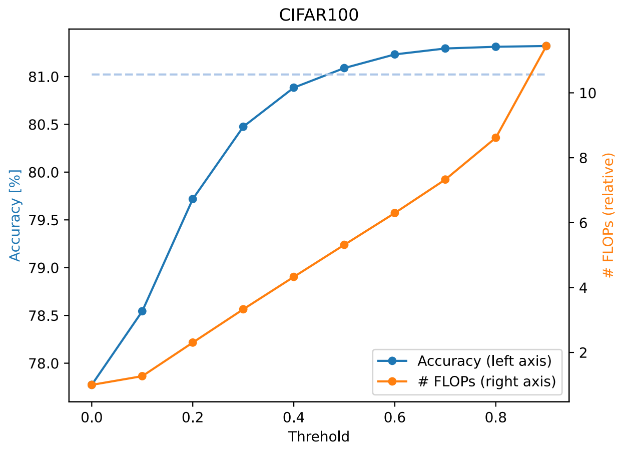


FIGURE 6. Relationship between cascade termination threshold, accuracy, and computational costs. The left axis and the blue line represent accuracy, while the right axis and the orange line represent relative computational costs. For this paper, we select a threshold that reduces the computational costs the most within a 0.3% accuracy reduction (indicated by the dotted line) from the maximum accuracy.

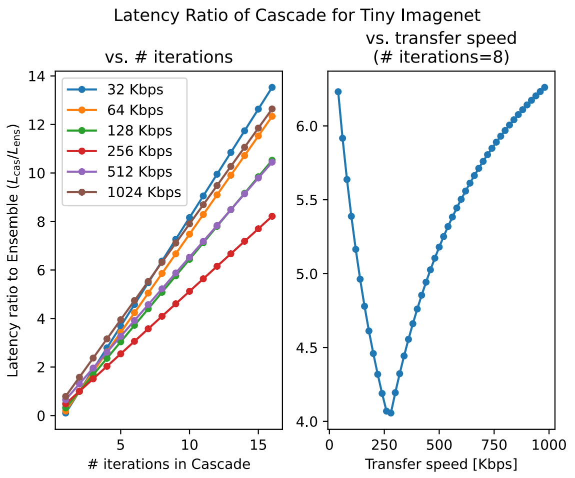


FIGURE 7. Latency ratio of cascade compared to a regular edge ensemble for various transfer speeds and cascade iteration counts. When data transfer time and inference time are equal, the parallel execution of these processes is most effective, resulting in the minimal latency ratio. When the number of iterations in the cascade is 1, the cascade ends with local device inference only. Therefore, the latency is lower than that of the original Edge Ensemble.

demonstrated in [32]. Since it is not always possible to infer a specific device order in Edge Ensembles, we set the threshold to be constant within a series of the cascade. FIGURE 6 shows the accuracy and computational complexity of the HOE for the CIFAR-100 dataset when varying the threshold in cascade. Lowering the threshold leads to a decrease in accuracy but reduces computational complexity. The appropriate threshold depends on the required accuracy of the target task. For this experiment, we set a tolerance range for accuracy degradation within a range of 0.3%, which is sufficiently smaller than the accuracy improvement effect achieved by the ensemble. We select the threshold for cascade to minimize computational costs while ensuring that the accuracy decrease on the validation set is limited to within 0.3% of the vanilla ensemble.

For TTA, we use GPS for each model to find valid policies from the candidate policy set before inference. We utilize a candidate policy set consisting of 276 different transformations. This set includes the 23 transformations proposed by RandAugment [36] with a transformation magnitude of 9, which consider the sign of the magnitude, as well as an additional transformation of horizontal flip. By combining two transformations from these 24 options, we generate ${}_{24}C_2 = 276$ unique transformations. Note that we do not consider the order of the combinations to prevent an excessive number of candidate policies. In the inference phase, we applied the top 10 transformation policies found by GPS for each model.

In weighted averaging, we use SL-based and ACC-based weighting methods. For SL-based weighting, we employ a 1×1 convolutional layer, similar to the approach used in [13], to assign a single weight to each model. We train the SL using back-propagation with the validation set. For the ACC-based weighting, we divide the validation set in half. We use one half to calculate the accuracies of each base learner and the other half to find the optimal T. Once we obtain T, we assign weights based on the accuracy of the entire validation set.

In order to avoid the complexity of the search space, each method has only one change from the vanilla ensemble. For example, in the case of the cascade, only the part that integrates base learners sequentially is changed from the vanilla ensemble, and the part that uses unweighted averaging to aggregate inference results is the same.

3) SETUP OF EVALUATION METRICS

As evaluation metrics, we assess the accuracy of all the methods through experiments. Furthermore, we estimate how computational costs and latency change compared to the original Edge Ensemble.

First, regarding computational costs, weighted averaging involves multiplying each model's predictions by their respective weights, so there is little change compared to the original Edge Ensemble. In the case of TTA, the computational costs increase by the number of TTA transformation policies applied. In our experiment, we applied 10 transformation policies, resulting in a 10-fold increase in computational costs. As for the cascade, the computational costs vary for each data point, depending on the number of models used for inference. In the experiment, we estimate the average change in the number of FLOPs (Floating-Point Operations) for cascade compared to the original Edge Ensemble.

Next, regarding latency, for weighted averaging, as with the computational cost, it is almost equivalent to the original Edge Ensemble. For TTA and cascade, latency changes compared to the original Edge Ensemble. To estimate them, first, we briefly estimate the latency in the original Edge Ensemble. A detailed latency analysis is conducted in [6], but here, to provide a more accessible estimate of how the method employed in this study impacts the Edge Ensemble's latency, we utilize a simplified latency model. Assuming that data transfer times and inference times are consistent across all devices, the latency of the Edge Ensemble L_{ens} is represented as follows:

$$L_{\text{ens}} = T_{\text{data}} + \tau, \tag{6}$$

where T_{data} represents the transfer time of the inference data to other devices, and τ represents the inference time of each device. This latency equation represents the time when a local device sends data to other remote devices simultaneously, and remote devices perform inference simultaneously. We ignore the time required to return the inference results in the equation because the data size of the inference results is considerably smaller than the inference data. In the case of TTA, the τ increases by the number of TTA transformation policies applied. Therefore, it can be expressed by the following equation:

$$L_{\text{TTA}} = T_{\text{data}} + k\tau, \tag{7}$$

where k represents the number of TTA policies applied. The increase in L_{TTA} compared to L_{ens} depends on the ratio of τ to T_{data}, but once the values are determined, it can be statically calculated. If τ is dominant, it increases by up to a maximum of k times. In the case of cascade, the latency varies depending on how data transfer and inference processing are controlled sequentially. Here, we assume that the inference of data on a certain device and the data transmission to the next inference device are carried out in parallel simultaneously. Then, the latency is represented by the following equation:

$$L_{\text{cas}} = \begin{cases} I\tau + (T_{\text{data}} - \tau)(I - 1) & (T_{\text{data}} - \tau \geq 0) \\ I\tau & (T_{\text{data}} - \tau < 0), \end{cases} \tag{8}$$

where I represents the number of iterations in the cascade. By performing inference and data transmission simultaneously, it is possible to hide the time taken for data transmission. FIGURE 7 summarizes latency changes from the original Edge Ensemble for various transfer speeds using Tiny ImageNet dataset. We estimate τ using a typical model inference speed on edge devices of 700 milliseconds for an image sized 224 × 224, based on measurements in [37]. We then convert this time for the Tiny ImageNet size of 64 × 64. Furthermore, we assume the transfer speeds range from tens to hundreds of kbps, based on common communication methods used in edge environments, as used in [38]. We estimate the data size using the JPEG images in the Tiny ImageNet dataset. The latency of the cascade varies depending on the number of cascade iterations. Therefore, in the experiments, we measure the number of cascade iterations and evaluate the latency by averaging the latency ratios for the six data transfer speeds used in the left figure of FIGURE 7.

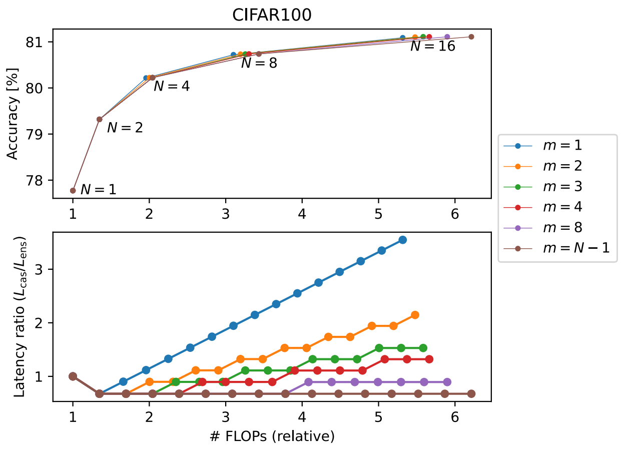


FIGURE 8. Latency and accuracy of *m*-parallel cascade for varying *m* in the case of the HOE setting with the CIFAR-100 dataset. *N* represents the total number of models in the cascade. Each dot in the bottom figure corresponds to an increasing *N* from 1 to 16.When the number of models is 1, both cascade and ensemble have the same latency because they use only a local model for inference. Depending on the number of models, cascade has an advantage over the ensemble regarding latency because cascade can terminate inference with only alocal model for simple data.

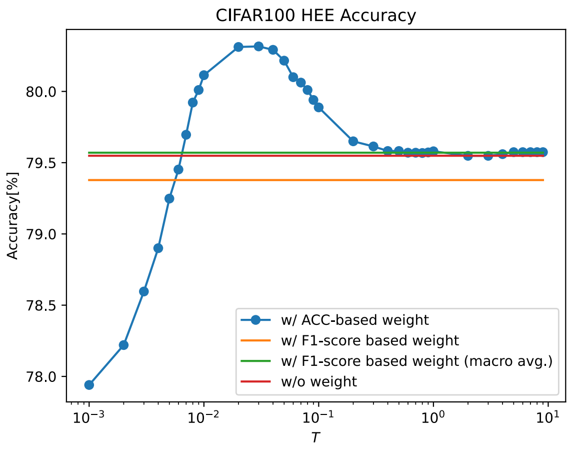


FIGURE 9. Accuracy using ACC-based weighting for varying *T* in Equation 5 in the case of the HEE model configuration with the CIFAR-100 dataset. The blue line shows the accuracy of the ACC-based weighted ensemble with different values of *T*, and the red line represents the accuracy of the vanilla ensemble as a baseline. The orange line represents the accuracy with the class-specific F1 score-based weighting method, while the green line represents the accuracy with weighting based on the averaged F1 score across all classes.

B. HYPERPARAMETER ANALYSIS

This section analyzes two hyperparameters of the proposed method, the parallelism m in the m-parallel cascade and the temperature T in the ACC-based weighting method.

1) PARALLELISM IN *M*-PARALLEL CASCADE

FIGURE 8 shows the accuracy and latency of the proposed method for m-parallel cascade for varying the parallelism parameter m. The top part of the figure shows the accuracy, and the bottom part shows the latency. The horizontal axis denotes the average number of models participating in inference, expressed as relative FLOPs. Because we obtained similar trends regardless of the dataset and model configuration, the figure only shows the data for the

TABLE 1. Accuracy of the single model and vanilla ensemble with 16 models for the baseline.

Dataset	Accuracy[%](diff with single)					
	HOE		HOE2		HEE	
	single	Ensemble	single	Ensemble	single	Ensemble
CINIC-10	87.1	89.5 (+2.4)	84.4	87.0 (+2.6)	85.3	88.7 (+3.4)
CIFAR-100	77.3	81.0 (+3.7)	72.4	76.0 (+3.6)	73.6	79.6 (+6.0)
Tiny ImageNet	65.1	72.2 (+7.1)	60.2	66.7 (+6.5)	60.9	69.4 (+8.5)

TABLE 2. Accuracy of each model composing the HEE setting.

Dataset	Accuracy[%]				
	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$
CINIC-10	81.0	84.5	85.9	86.5	87.2
CIFAR-100	66.9	71.1	74.6	76.1	77.4
Tiny ImageNet	56.3	59.1	60.8	63.4	65.1

CIFAR-100 in the HOE setting. We observed that ensemble accuracy for the same number of models remains almost the same as the change in m while the computational complexity increases and latency decreases. We also confirmed that the latency is most efficient when m is a divisor of the number of remote devices used for inference. Section V evaluates the trade-off between computational complexity and latency for cases of $m = 1, 2, 4$, and 8 for the cascade with 16 models.

2) TEMPERATURE IN ACC-BASED WEIGHTING METHODS

FIGURE 9 shows the accuracy for varying temperature T from 0.001 to 9 for the ACC-based weighting function in Equation 5. We obtained similar trends regardless of the dataset. However, for model configurations, the HOE and HOE2 settings had little effect on ACC-based weight assignment (details will be discussed in Section V). Therefore, we show only HEE for CIFAR-100 in the figure. For HEE on all datasets, too low values for T significantly degraded accuracy, and too high values for T had no effect, resulting in a T peak between 0.01 and 0.1. In the evaluation of Section V, we use the value of T that yielded the highest accuracy on the validation set for each dataset and model configuration.

V. EVALUATION

In this section, we conduct experiments to evaluate the effectiveness of the Edge Ensembles when using cascade, weighted averaging, and TTA for model integration. We also experimentally validate the effectiveness of the proposed methods of m-parallel cascade, ensemble label-based learning, and the ACC-based weighting method. TABLE 1 presents the accuracy of the single model and vanilla ensemble consisting of 16 models as baselines for each model configuration and dataset. The column of the single model represents the average accuracy of all the models composing its model configurations. TABLE 2 offers the accuracy of the single model composing the HEE setting for each x.

A. CASCADE

TABLE 3 shows the accuracy, FLOPs, and latency of the ensemble with cascade for each dataset and model configuration using 16 models. In the table, the values in parentheses for accuracy represent the difference compared to the vanilla ensemble. The FLOPs and latency values indicate the ratio when compared to the values of the vanilla ensemble. For the HOE and HOE2 settings, we calculate the FLOPs and latency based on the average number of models participating in inference. For the HEE setting, we employ the cascades of ascending and descending order based on model size criteria due to its composition of models with different structures. In addition, we calculate the FLOPs of each model for each x. However, we calculate the latency to be constant regardless of x for simplicity because the latency of data communication is independent of x, and the inference time depends on the device performance in addition to x.

The accuracy in all model configurations and datasets remained nearly unchanged while reducing the FLOPs. However, the latency increased. When comparing each model configuration, we observes that the HOE and HOE2 settings yielded similar results. In the case of HEE, the descending order cascade outperformed the ascending order cascade. This is due to the thresholds used to minimize accuracy drops in the cascade process. In the ascending order cascade, to maintain accuracy, the inference must continue until the later strong models if the earlier weak models make incorrect class predictions. Therefore, a higher termination threshold is necessary, and terminating the inference process becomes more challenging, even for simpler data.

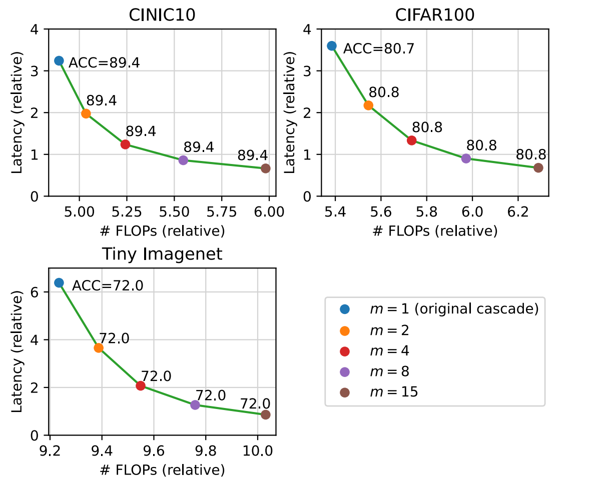


FIGURE 10. Computational cost and latency of the m-parallel cascade in the HOE setting for each dataset. The vertical axis, representing latency, shows the latency ratio compared to a regular ensemble. The horizontal axis, representing FLOPs, indicates the relative computational cost of the cascade compared to a single model, where the computational cost of a single model is normalized to 1. These values were measured based on the average number of models used in the cascade. While there are minor variations in accuracy based on the value of m, the method preserves overall accuracy and allows for tuning latency and FLOPs.

Furthermore, we applied m-parallel cascade to the HOE setting and examined the changes in latency and computational complexity. FIGURE 10 shows the variation in latency and FLOPs of the m-parallel cascade for each dataset. By varying the value of m, we achieved a trade-off between

TABLE 3. **Accuracy, FLOPs, latency in cascade with 16 models.**

Dataset	Accuracy [%] (diff from vanilla ensemble)				FLOPs ratio to vanilla ensemble				Latency ratio to vanilla ensemble			
	HOE	HOE2	HEE		HOE	HOE2	HEE		HOE	HOE2	HEE	
			asc	desc			asc	desc			asc	desc
CINIC-10	89.4 (−0.1)	86.9 (−0.1)	88.6 (−0.1)	88.6 (−0.1)	0.31×	0.34×	0.50×	0.26×	3.24×	3.69×	6.02×	1.77×
CIFAR-100	80.7 (−0.3)	75.9 (−0.1)	79.2 (−0.4)	79.3 (−0.3)	0.34×	0.41×	0.54×	0.35×	3.60×	4.48×	6.42×	2.85×
Tiny ImageNet	72.0 (−0.2)	66.4 (−0.3)	69.3 (−0.1)	69.4 (−0.0)	0.58×	0.56×	0.69×	0.63×	6.38×	6.23×	8.01×	6.38×

TABLE 4. **Accuracy of ensemble with weighted averaging.**

Dataset	Weighted Averaging Accuracy [%] (diff from the baseline)					
	HOE		HOE2		HEE	
	ACC-based	SL-based	ACC-based	SL-based	ACC-based	SL-based
CINIC-10	89.5 (+0.0)	89.5 (−0.0)	87.0 (+0.0)	87.0 (+0.0)	89.0 (+0.3)	88.9 (+0.2)
CIFAR-100	81.1 (+0.1)	81.1 (+0.1)	76.1 (+0.1)	75.9 (−0.1)	80.3 (+0.7)	79.9 (+0.3)
Tiny ImageNet	72.1 (−0.1)	72.0 (−0.2)	66.7 (+0.0)	66.6 (−0.1)	70.2 (+0.8)	70.5 (+1.1)

TABLE 5. **Accuracy of ensemble with TTA.**

Dataset	TTA Accuracy [%] (diff from the baseline)					
	HOE		HOE2		HEE	
	single	Ensemble	single	Ensemble	single	Ensemble
CINIC-10	87.9 (+0.8)	89.5 (+0.0)	85.2 (+0.8)	87.1 (+0.1)	86.4 (+1.1)	88.7 (+0.0)
CIFAR-100	78.0 (+0.7)	81.1 (+0.1)	73.1 (+0.7)	76.2 (+0.2)	75.1 (+1.5)	79.6 (+0.0)
Tiny ImageNet	66.0 (+0.9)	72.2 (+0.0)	61.1 (+0.9)	67.2 (+0.5)	62.6 (+1.7)	69.9 (+0.5)

latency and computational complexity while maintaining nearly the same level of accuracy. For instance, when m was 4, the latency was about 2.8 times lower than the original cascade while increasing FLOPs by 1.06 times on average accross all datasets. Compared to the vanilla ensemble, the average amount of computation was 0.43 times, while the increase in latency was controlled to just 1.55 times. Furthermore, when m is greater than or equal to 8, it reduces latency and computation costs while maintaining accuracy compared to the vanilla ensemble. In practical applications, selecting an appropriate value of m based on the acceptable latency, it is possible to reduce computational complexity while maintaining accuracy.

B. WEIGHTED AVERAGING

TABLE 4 shows the accuracy in the ensemble with 16 models when using the SL-based and ACC-based weighted averaging for each dataset and model configuration. In the HOE and HOE2 settings, both ACC-based and SL-based weighted averaging did not yield significant improvements. However, in the HEE setting, both methods proved to be effective. This can be attributed to the minimal variations between the performances of models within the HOE and HOE2 settings, which renders the weighting meaningless.

In the HEE setting, when comparing ACC-based and SL-based weighting methods, ACC-based outperformed SL-based in CIFAR-100, while SL-based was superior in Tiny ImageNet. CIFAR-100 is a simpler dataset than Tiny ImageNet. It is considered that ACC-based weighting, which simply prioritizes each prediction based on accuracy, was more effective than performing complex weight learning using SL for simpler data. Furthermore, in the case of CINIC-10, the weighting methods did not yield significant effects. We can attribute this to the fact that CINIC-10 is the easiest dataset and the more minor performance differences between base learners within the HEE setting, as shown in TABLE 1. In the case of HEE, where there were significant effects, we analyzed 1) the accuracy changes for the number of base learners and 2) the learning of weights using the proposed training method with ensemble prediction labels.

1) EFFECTIVENESS FOR VARYING NUMBER OF BASE LEARNERS

In Edge Ensemble scenarios, not all models on each device are always available due to communication constraints. Therefore, it is essential to verify the effectiveness of weights learned using all models in training when only a subset of models is available in inference. FIGURE 11 shows the average accuracy in the HEE setting for the change in the number of base learners for the ensemble with weighted averaging. We trained weights from all 16 models in the training phase and randomly selected subsets of models in the inference phase. We confirmed that conducting weighted inference using randomly selecting a subset of models is as effective as using all 16 models. This observation validates the effectiveness of the weighting approach for Edge Ensemble scenarios.

2) LEARNING WITH ENSEMBLE PREDICTION LABELS

FIGURE 11, compares the accuracy of learning with ground truth labels and ensemble prediction labels for ACC-based and SL-based weights. The weighting method of using ensemble prediction labels yielded results that were almost equivalent to using ground truth labels. This experimental confirmation demonstrates the effectiveness of ensemble label-based weighting in situations where weighted averaging is beneficial.

C. TTA

TABLE 5 shows the accuracy of a single model and an ensemble with 16 models when applying TTA for each dataset and model configuration. Applying TTA to single models improved accuracy consistently across all model configurations. However, in most cases, ensembling did not show significant improvements. Although there was some improvement in ensembling for the HOE2 and HEE settings on Tiny ImageNet, it was still lower than the accuracy improvement achieved by TTA on a single model. These results suggest that ensembling and TTA share some redundant functionalities.

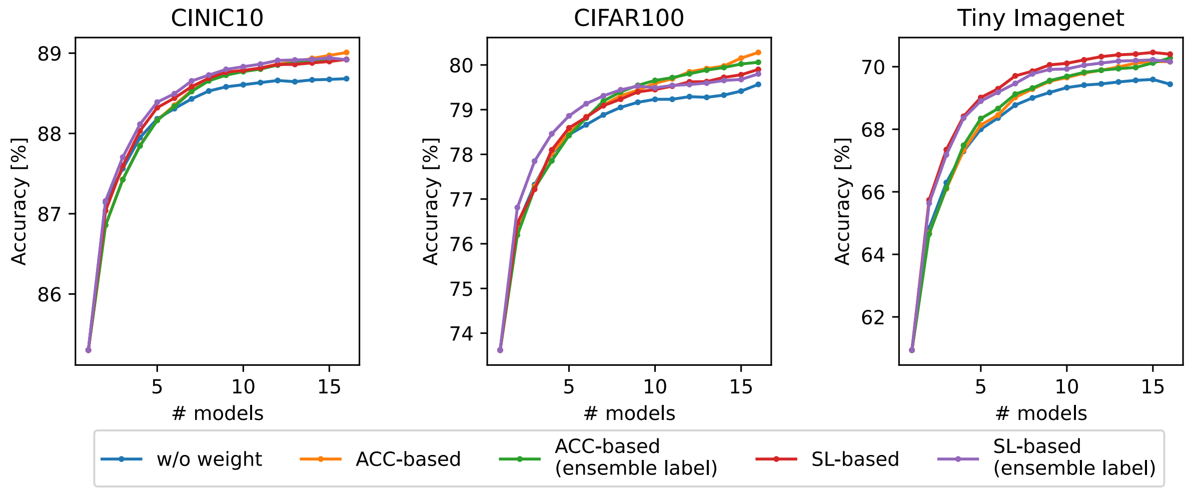


FIGURE 11. Comparison of ensemble accuracies with and without weighted averaging in the HEE setting for varying numbers of base learners. The blue line represents the vanilla ensemble, while the other colored lines depict ensembles with weighted averaging using each weighting method. For the green and purple lines, ensemble prediction labels are used instead of ground truth labels for weighting. We randomly sampled models to measure accuracy after training the weights using all 16 models. While there was some variation in effectiveness among the weighting methods, weighted averaging showed efficacy even with varying the number of base learners. The method using ensemble prediction labels demonstrated nearly equivalent performance to using ground truth labels.

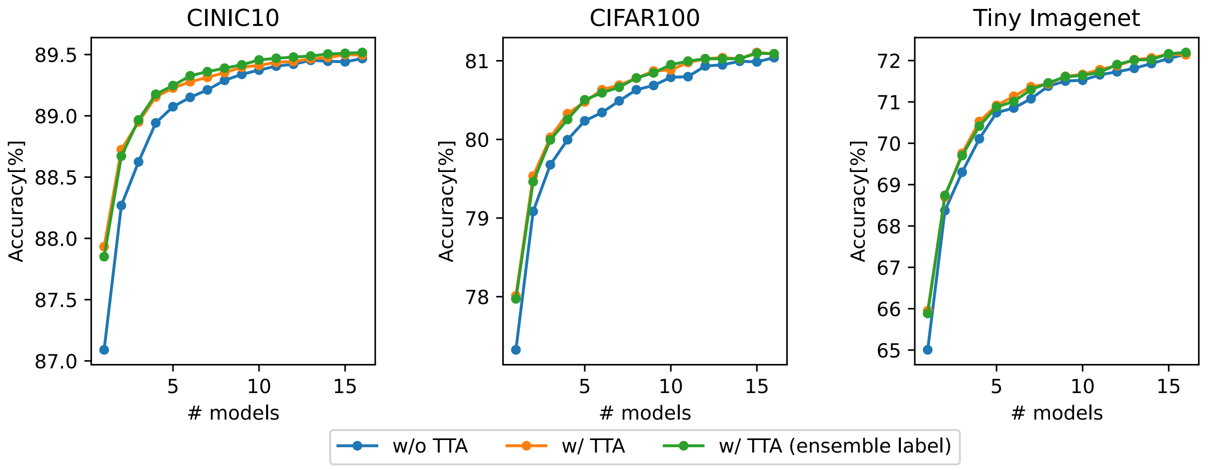


FIGURE 12. Comparison of ensemble accuracies with and without TTA in the HOE setting for varying numbers of base learners. The blue line represents the vanilla ensemble, the orange line represents the ensemble with TTA, and the green line represents the ensemble with TTA using policies learned from ensemble prediction labels instead of ground truth labels. As the number of base learners increased, the difference in accuracy between using TTA and not using TTA decreased. Furthermore, the effect of TTA was almost equivalent regardless of the difference in the labels used during policy learning.

FIGURE 12 shows the effectiveness of TTA for each total number of base learners in the ensemble for the HOE setting, which had the lowest efficacy of TTA in the ensemble among all model configurations. TTA demonstrated a notable impact when the ensemble had a small number of models, but the effect diminished as the number of models increased. Similar trends were observed for other model configurations and datasets not shown in the figure. These results indicate that TTA can be an effective method to improve accuracy in Edge Ensembles further when the number of available models is limited.

FIGURE 12 also shows the accuracy of the proposed ensemble label-based learning, which learn the policy for TTA by GPS using ensemble prediction labels instead of ground truth labels. Although the accuracy is slightly lower for the proposed method compared to the case using ground truth labels, the overall effectiveness of TTA is still evident. This indicates that even in Edge Ensemble scenarios where ground truth labels are unavailable, treating ensemble prediction labels as pseudo-ground truth allows for effective TTA policy learning.

D. EVALUATION OF COMBINATIONS

In this section, we evaluate the performance when combining these methods. Cascade, TTA, and weighted averaging are three independent methods that can be used simultaneously, but each has distinct characteristics. Cascade is a method that reduces accuracy but decreases computational complexity,

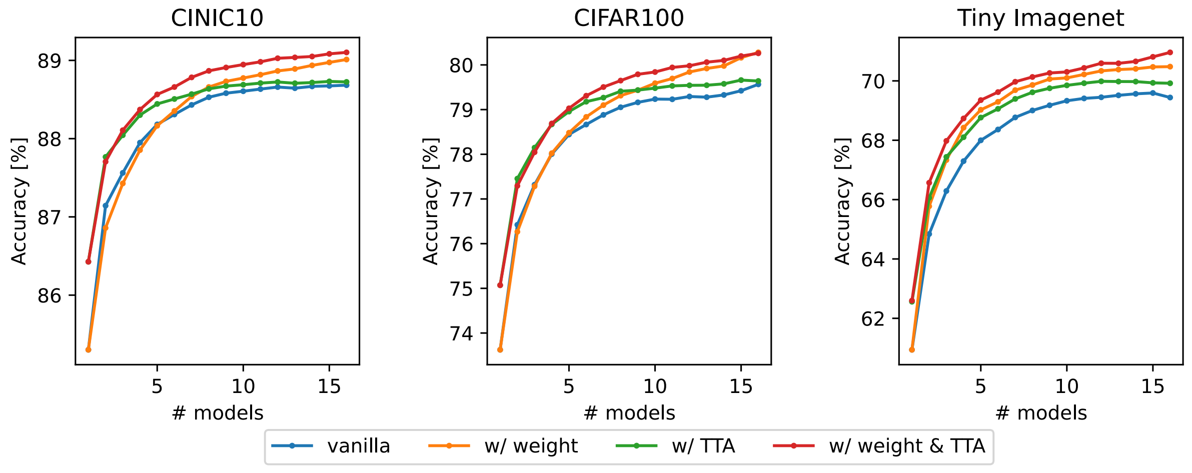


FIGURE 13. Comparison of ensemble accuracies with TTA and weighted averaging and their combination in the HEE setting for varying numbers of base learners. The combination of TTA and weighted averaging resulted in the simultaneous effects of both techniques.

TABLE 6. Performance of a combination of cascade and weighted averaging.

Dataset	HEE w/ cascade & weighted averaging		
	Accuracy [%] (diff from cascade/weighting/vanilla)	FLOPs ratio (cascade alone)	Latency ratio (cascade alone)
CINIC-10	$88.7\ (+0.1/-0.3/-0.0)$	$0.26\times\ (0.26\times)$	$1.84\times\ (1.77\times)$
CIFAR-100	$79.8\ (+0.5/-0.5/+0.2)$	$0.35\times\ (0.35\times)$	$2.93\times\ (2.85\times)$
Tiny ImageNet	$70.4\ (+1.0/-0.1/+1.0)$	$0.60\times\ (0.63\times)$	$5.95\times\ (6.38\times)$

while TTA is a technique that increases accuracy at the expense of computational complexity. These two methods have contrasting features, and there are no advantages to using them simultaneously. Therefore, we do not consider scenarios where they are used together. Consequently, in this section, we evaluate two combinations: the combination of cascade and weighted averaging, and the combination of TTA and weighted averaging. In evaluating each combination, we assess the HEE setting in which weighted averaging was the most effective among the three model configurations.

1) COMBINATION OF CASCADE AND WEIGHTED AVERAGING

For the combination of cascade and weighted averaging, we use cascade with descending order and employed weighting methods that yielded higher accuracy for each dataset. We use the same evaluation metrics as in the case of using cascade alone. TABLE 6 presents accuracy, FLOPs ratio, and latency ratio of HEE settings with cascade and weighted averaging. By combining cascade and weighted averaging, FLOPs and latency remained nearly the same level as in the case of cascade alone, while the accuracy matched or exceeded that of the vanilla ensemble.

2) COMBINATION OF TTA AND WEIGHTED AVERAGING

In evaluating the combination of TTA and weighted averaging, we assess the accuracy by varying the number of models, similar to the evaluation of TTA alone. FIGURE 13 compares the accuracy achieved when combining weighted averaging and TTA with the accuracy obtained when using each method individually and when using none of them. By combining TTA, which is effective when the number of models is small, with weighted averaging, which is more effective when the number of models is large, we consistently improved accuracy compared to the vanilla ensemble across all numbers of models.

E. DISCUSSION

Thus far, we have assumed the context of Edge Ensembles and evaluated the performance by applying cascade, weighted averaging, and TTA to the model integration part through experiments.

For cascades, previous studies have reported that cascades reduce FLOPs and improve inference speed compared to simple ensembles [23]. However, our study assumes an Edge Ensemble where models on each device conduct inference in parallel. As a result, while the cascade reduced computational costs, it increased latency in our context. Furthermore, the m-parallel cascade mitigated the increased latency based on the parallelism factor m, allowing for the adjustment of the degree of computational costs reduction and the latency increase in the cascade.

Weighted averaging was the most effective when applied with the HEE configuration, similar to previous studies [13]. However, in this study, we went one step further by evaluating performance, assuming variations in the availability of other devices in Edge Ensembles. We observed that weighted averaging is effective even when varying the number of base learners, demonstrating its effectiveness in the context of Edge Ensembles.

In the case of TTA, similar to previous studies [32] that combined conventional DNN ensembles with TTA, we observed that TTA on a single model was the most effective, and its benefits diminished when used in ensembles. In our study, we analyzed accuracy with TTA for different numbers of models to assess its utility in the context of Edge Ensembles. We found that TTA can be used as an option to improve the accuracy of Edge Ensembles when only a few nearby devices are available for collaborative inference.

In addition, we validated the effectiveness of using ensemble prediction labels for learning TTA policies and weighting methods. We found that the results were generally comparable to using ground truth labels. This demonstrates the feasibility of using ensemble prediction labels alone to improve the accuracy of the ensemble itself without involving complex model training. However, it is important to note that this approach heavily relies on the accuracy of the ensemble prediction. For instance, if the Edge Ensembles contain numerous devices that produce random answers, the ensemble prediction result would be close to random, making effective learning impossible. Moreover, malicious models within the ensemble could lead to intentional misdirection of the predictions. Authors in [7] proposed removing devices with high error rates as a solution, but this approach also depends on the ensemble prediction results, remaining similar challenges. Addressing adversarial models in Edge Ensembles is a crucial future research direction.

VI. CONCLUSION

This paper leverages three approaches to enhance Edge Ensembles: cascade as an integration method with high efficiency, weighted averaging as an integration method to improve accuracy, and TTA as a method to integrate each base learner with improved accuracy. The experimental results support the effectiveness of these methods in improving Edge Ensembles' performance. The proposed m-parallel cascade addresses the latency issues present in conventional cascades for Edge Ensembles while reducing computational costs compared to simple ensembles. Weighted averaging improved accuracy, especially in the HEE setting, and enhanced accuracy in Edge Ensembles where the availability of neighboring devices varies. TTA significantly enhances the accuracy of single models compared to ensemble cases. We also validate its effectiveness in scenarios with limited availability of neighboring devices for Edge Ensembles. Furthermore, our demonstration highlighted the effectiveness of ensemble label-based learning, enabling Edge Ensembles to adapt their model integration process on the edge side.

In conclusion, the collaborative inference system, which efficiently processes inference by leveraging the collective power of multiple devices, is expected to gain significant importance in the era of IoT. In this study, we validated fundamental ensemble-based approaches in the context of collaborative inference while also introducing new ideas and assessing their effectiveness. However, critical security issues related to test data privacy and adversarial handling still remain for collaborative inference, necessitating further research in future work.

REFERENCES

[1] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[2] W.-Q. Ren, Y.-B. Qu, C. Dong, Y.-Q. Jing, H. Sun, Q.-H. Wu, and S. Guo, "A survey on collaborative DNN inference for edge intelligence," *Mach. Intell. Res.*, vol. 20, no. 3, pp. 370–395, Jun. 2023.

[3] P. Joshi, M. Hasanuzzaman, C. Thapa, H. Afli, and T. Scully, "Enabling all in-edge deep learning: A literature review," *IEEE Access*, vol. 11, pp. 3431–3460, 2023.

[4] N. Shlezinger and I. V. Bajic, "Collaborative inference for AI-empowered IoT devices," *IEEE Internet Things Mag.*, vol. 5, no. 4, pp. 92–98, Dec. 2022.

[5] N. Shlezinger, E. Farhan, H. Morgenstern, and Y. C. Eldar, "Collaborative inference via ensembles on the edge," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 8478–8482.

[6] M. Malka, E. Farhan, H. Morgenstern, and N. Shlezinger, "Decentralized low-latency collaborative inference via ensembles on the edge," 2022, *arXiv:2206.03165*.

[7] X. Feng, C. Luo, B. Wei, J. Zhang, J. Li, H. Wang, W. Xu, M. C. Chan, and V. C. M. Leung, "Time-constrained ensemble sensing with heterogeneous IoT devices in intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12949–12960, Nov. 2023.

[8] S. F. Yilmaz, B. Hasircioglu, and D. Gündüz, "Over-the-air ensemble inference with model privacy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 1265–1270.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 54, 2017, pp. 1273–1282.

[10] Y. Yang, H. Lv, and N. Chen, "A survey on ensemble learning under the era of deep learning," *Artif. Intell. Rev.*, vol. 56, no. 6, pp. 5545–5589, Jun. 2023.

[11] M. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan, "Ensemble deep learning: A review," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105151.

[12] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, p. e1249, Jul. 2018.

[13] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *J. Appl. Statist.*, vol. 45, no. 15, pp. 2800–2818, Nov. 2018.

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[15] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 8583–8595.

[16] N. Du et al., "GLaM: Efficient scaling of language models with mixture-of-experts," in *Proc. Int. Conf. Mach. Learn.*, vol. 162, 2022, pp. 5547–5569.

[17] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The Sparsely-Gated Mixture-of-Experts layer," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–19.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.

[20] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, nos. 1–2, pp. 239–263, May 2002.

[21] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Dec. 2001, pp. 511–518.

[22] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR05)*, vol. 2, Jun. 2005, pp. 236–243.

[23] X. Wang, D. Kondratyuk, E. Christiansen, K. M. Kitani, Y. Alon, and E. Eban, "Wisdom of committees: An overlooked approach to faster and more accurate models," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–22.

[24] S. Hassan, A. Khosravi, and J. Jaafar, "Bayesian model averaging of load demand forecasts from neural network models," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3192–3197.

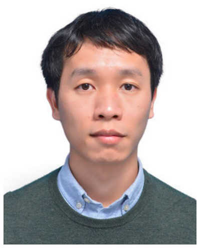
[25] L. J. Gosink, C. C. Overall, S. M. Reehl, P. D. Whitney, D. L. Mobley, and N. A. Baker, "Bayesian model averaging for ensemble-based estimates of solvation-free energies," *J. Phys. Chem. B*, vol. 121, no. 15, pp. 3458–3472, Apr. 2017.

[26] A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using Bayesian model averaging to calibrate forecast ensembles," *Monthly Weather Rev.*, vol. 133, no. 5, pp. 1155–1174, May 2005.
[27] S. Rothe and D. Söffker, "Comparison of different information fusion methods using ensemble selection considering benchmark data," in *Proc. 19th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2016, pp. 73–78.
[28] T. Iqball and M. A. Wani, "Weighted ensemble model for image classification," *Int. J. Inf. Technol.*, vol. 15, no. 2, pp. 557–564, Feb. 2023.
[29] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, Jul. 1996.
[30] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, p. 25, Jan. 2007.
[31] C. Ju, M. Combs, S. D. Lendle, J. M. Franklin, R. Wyss, S. Schneeweiss, and M. J. van der Laan, "Propensity score prediction for electronic healthcare databases using super learner and high-dimensional propensity score methods," *J. Appl. Statist.*, vol. 46, no. 12, pp. 2216–2236, Sep. 2019.
[32] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–30.
[33] A. Lyzhov, Y. Molchanova, A. Ashukha, D. Molchanov, and D. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," in *Proc. Conf. Uncertainty Artif. Intell.*, vol. 124, 2020, pp. 1308–1317.
[34] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "Better aggregation in test-time augmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1194–1203.
[35] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 4163–4174.
[36] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3008–3017.
[37] C. Luo, X. He, J. Zhan, L. Wang, W. Gao, and J. Dai, "Comparison and benchmarking of AI models and frameworks on mobile devices," 2020, *arXiv:2005.05085*.
[38] M. Ghamari, H. Arora, R. S. Sherratt, and W. Harwin, "Comparison of low-power wireless communication technologies for wearable health-monitoring applications," in *Proc. Int. Conf. Comput., Commun., Control Technol. (I4CT)*, Apr. 2015, pp. 1–6.

SHUNGO KUMAZAWA (Graduate Student Member, IEEE) received the B.E. degree in electronics from Hokkaido University, Sapporo, Japan, in 2020, and the M.E. degree in information and communication engineering from the Tokyo Institute of Technology, Yokohama, Japan, where he is currently pursuing the Ph.D. degree in information and communication engineering. His research interests include machine learning and computer architecture. He received the Research Fellowship for Young Scientists from JSPS, in 2022.

JAEHOON YU (Member, IEEE) received the B.E. degree in electrical and electronic engineering and the M.S. degree in informatics (communications and computer engineering) from Kyoto University, Kyoto, Japan, in 2005 and 2007, respectively, and the Ph.D. degree in informatics (information systems engineering) from Osaka University, Osaka, Japan, in 2013. From 2013 to 2019, he was an Assistant Professor with Osaka University. From 2019 to 2023, he was an Associate Professor with the Tokyo Institute of Technology, Japan. He is currently a VP of Technology with Samsung Electronics. His research interests include computer vision, machine learning, and system-level design. He is a member of IEICE and IPSJ.

KAZUSHI KAWAMURA (Member, IEEE) received the B.Eng., M.Eng., and Dr.Eng. degrees in computer science from Waseda University, in 2012, 2013, and 2016, respectively. From 2018 to 2019, he was an Assistant Professor with the Department of Communications and Computer Engineering, Waseda University. He is currently a specially appointed Assistant Professor with the Institute of Innovative Research, Tokyo Institute of Technology. His research interests include parallel algorithms and architectures for annealing computation and machine learning. He is a member of IEICE and IPSJ.



THIEM VAN CHU (Member, IEEE) received the Ph.D. degree from the Tokyo Institute of Technology, in 2018. Upon graduation, he joined the School of Information Science, Japan Advanced Institute of Science and Technology as an Assistant Professor. In 2020, he moved to the Tokyo Institute of Technology. His research interests include the intersection of computer architecture, reconfigurable computing, and machine learning.

MASATO MOTOMURA (Fellow, IEEE) received the B.S., M.S., and Doctor of Engineering degrees from Kyoto University, Kyoto, Japan, in 1985, 1987, and 1996, respectively. In 1987, he joined the NEC Central Research Laboratories, Kawasaki, Japan, working on various hardware architectures, including approximate text search engines, multi-threaded on-chip parallel processors, computing-in-memory chips, and reconfigurable systems. From 2001 to 2008, he was with NEC Electronics, Kawasaki, where he led the research and business development of the dynamically reconfigurable processor (DRP) he invented. He was also a Visiting Researcher with the MIT Laboratory for Computer Science, Cambridge, USA, from 1991 to 1992, and the Group Manager of Architecture-Circuits Interdisciplinary Research with the NEC Central Laboratory, from 2008 to 2011, respectively. In 2011, he changed his position from industry to academia and became a Professor with Hokkaido University, Sapporo, Japan, to cultivate solid-state circuit research activities with younger generations. Later, he became a Professor with the Tokyo Institute of Technology (TokyoTech), Yokohama, Japan, in 2019, where he established and has been leading the Artificially Intelligent Computing (ArtIC) Research Unit. Since 2011, he has been actively working on reconfigurable and parallel architectures for deep neural networks, machine learning, annealing machines, and general intelligent/domain-specific computing. His group has published "AI chip" papers almost every year at ISSCC and Symposium on VLSI, since 2017. He is a member of IEICE, IPSJ, JSAI, and EAJ. He is a 2022 IEEE Fellow (SSCS) for contributions to memory-logic integration of reconfigurable chip architecture. He received the IEEE JSSC Annual Best Paper Award, in 1992, the IPSJ Annual Best Paper Award, in 1999, and the IEICE Achievement Award, in 2011, respectively. He was also received the Ichimura Academic Award and Yamasaki Award, in 2022, for his leadership in developing and productizing DRP technology (a series of DRP-based microcontroller products are now produced by Renesas Electronics) and the accumulation of AI-chip achievements in recent years.