Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Cloud-edge-end integrated Artificial intelligence based on ensemble learning

Zhen Gao [a], Daning Su [a], Shuang Liu [a], Yuqi Zhang [a], Chenyang Wang [b],*, Cheng Zhang [c], Xiaofei Wang [d], Tarik Taleb [e]

[a] School of Electrical and Information Engineering, Tianjin University, Tianjin, 300072, China
[b] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, 518000, Guangdong, China
[c] Faculty of Digital Economics and Managements, Tianjin University of Finance and Economics, Tianjin, 300222, China
[d] College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China
[e] Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, 44780, Germany

ARTICLE INFO

Keywords:
Cloud-edge-end architecture
Edge computing
Ensemble learning
DNN

ABSTRACT

Deep neural networks (DNNs) have been extensively used in the domains of artificial intelligence (AI) applications. Their inherent complexity primarily drives the deployment of DNN models in cloud environments. However, the geographical distance between the cloud and the end-users fails to meet the low-latency requirements of time-sensitive applications. Edge computing has emerged as a viable way to address this issue, nevertheless, the inherent constraints of limited resources on edge servers pose challenges in supporting intricate models. Solutions relying on network compression or model segmentation often fall short in meeting both performance and reliability needs. For the few ensemble-based solutions, the diversity between base models is not fully explored, and the low-latency advantage of edge computing is not fully utilized. In this paper, we propose a cloud–edge-end integrated approach for building an efficient and reliable DNN inference platform based on ensemble learning. In this design, heterogeneous models are trained on the cloud according to the resource constraints of edge servers, and the inference process is performed independently on each edge server, whose outputs are combined at the end-user side to get the final result. Furthermore, a diversity-based deployment scheme is proposed to build a user-centric network for edge AI. The generation of base models is explored, and the effectiveness of the proposed approach is demonstrated through two case studies.

1. Introduction

Artificial intelligence (AI) has seen rapid and widespread development in recent years, impacting domains such as computer vision [1], natural language processing [2], speech recognition [3] and robot control [4]. For image classification tasks, Convolutional Neural Networks (CNNs) like VGGNets [5], GoogLeNets [6], and ResNets [7] have shown outstanding performance. Particularly, the computational complexity of DNNs/CNNs has progressively increased [8] with the increasing demands of the users. Hence, the deployment of large-scale deep learning (DL) models often involves their utilization in cloud environments. In this design, end devices primarily transmit input data to the cloud and then await the DL inference outcomes. However, relying solely on cloud-based inference poses challenges in delivering DL services ubiquitously, particularly for time-sensitive applications like autonomous driving [9].

Edge computing is a promising solution to provide computing services to users with low latency [10–12]. However, edge servers are usually resource-limited and may be inadequate to run the complete DL model and provide quality-guaranteed results. Various approaches have been explored to mitigate this, primarily focusing on two categories: model compression and model segmentation [13]. The compression approaches are utilized to decrease the computation complexity during inference so that the simplified model fits the resource capacity of the edge nodes. By applying different compression strategies, a set of DL models can be created, each offering a balance between performance and resource use [14]. Nevertheless, high compression rates often come at the cost of reduced performance. To guarantee task execution performance, model segmentation techniques are widely studied, where the original DL model is partitioned horizontally or vertically into multiple sub-models and offloaded to the edge servers [15,16]. These sub-models are partitioned based on resource requirements and data exchange demands. For instance, [17] proposes Early Exit of Inference (EEoI) to shorten the inference time on edge servers for a portion of the input. However, the high bandwidth should be guaranteed between edge servers for adjacent model partitions, leading to additional latency.

* Corresponding author.
E-mail address: chenyangwang@ieee.org (C. Wang).

https://doi.org/10.1016/j.comcom.2025.108103
Received 1 May 2024; Received in revised form 2 December 2024; Accepted 5 February 2025

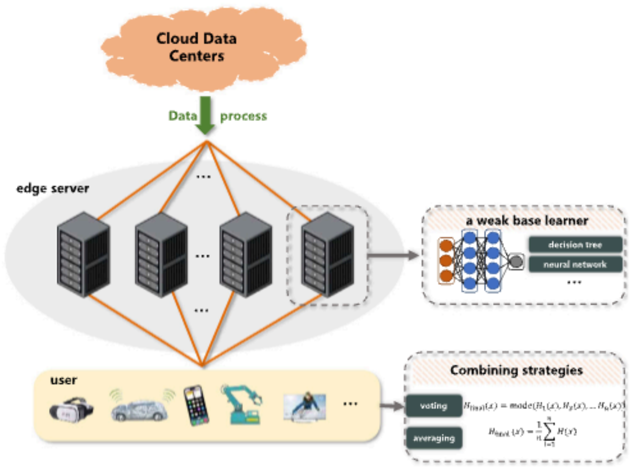

Fig. 1. The cloud–edge-end ensemble AI learning architecture.

In this paper, we present a cloud–edge-end integrated AI system that addresses the aforementioned issues, as shown in Fig. 1. Initially, base models (*a.k.a.,* weak learners) are trained in a simplified manner in the cloud. Second, simplified base models on edge servers provide inference results for input from the end-user side. Third, we can obtain the final result by aggregating the inference results on the end-user side. Unlike traditional methods, we incorporate complexity constraints in the generation of base models and emphasize both their similarity and diversity. The proposed approach strikes a balance between model complexity and performance, with minimal data exchange overhead. Additionally, the failure of any edge server will have minimal impact on service performance, as other servers will compensate for the loss. The contributions of this paper are listed in the following:

- Considering the geographical distance between end users and the cloud, as well as the limited resources of edge servers that cannot handle complex models, we propose an integrated intelligence approach, the cloud–edge-end system, based on ensemble learning. This approach enables rapid deployment to meet users' low-latency requirements.
- To maximize model diversity in ubiquitous AI services, while meeting the low-latency and high-accuracy demands of tasks, we propose an edge server selection algorithm that achieves a balance between delay and accuracy. By leveraging an ensemble of multiple models and redundant deployment, the scheme tolerates multiple model failures, significantly improving system reliability.
- Case studies, using ensemble ResNets of varying layers and VGGNets with different pruning rates, demonstrate that the ensemble of diverse low-complexity models achieves accuracy comparable to high-complexity models while maintaining strong performance even with a single model failure. The results demonstrate the superiority of the cloud–edge-end integrated system in ensuring accuracy and enhancing reliability.

The rest of the paper is organized as follows: Section 2 introduces the related works about edge AI and ensemble learning. In Section 3, we propose the cloud–edge-end integrated system paradigm and the diversity-based deployment scheme. The problem formulation is presented in Section 4. Then, we use the Markov decision process to model the process of users selecting edge servers and discuss the generation principle of base models in Section 5. Subsequently, two case studies are presented in Section 6 to exemplify the efficacy of the proposed methodology. Finally, Section 7 concludes the whole article.

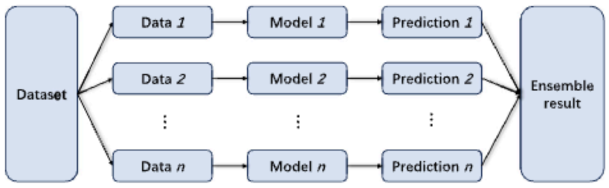

Fig. 2. Illustration of ensemble learning process.

2. Related work

2.1. Basics of ensemble learning

Unlike traditional learning techniques that construct a single model from training data, ensemble methods generate multiple models and combine their predictions. As seen in Fig. 2, an ensemble is created by many models, referred to as base models, which are derived from training data using a base learning method such as a decision tree, neural network, or any other kind of learning algorithm. Ensemble approaches may use either a single base learning algorithm or numerous learning algorithms to generate ensembles that are either homogeneous or heterogeneous. The approach used to merge the outcomes is to conduct a vote or calculate the average of the category scores in classification issues. An alternative approach is to use an extra learner who is specifically educated to integrate the base learners [18]. An ensemble's capacity to generalize is often more powerful than that of its base learners. Ensemble techniques are appealing primarily because they can amalgamate weak learners, which exhibit just a minor improvement above random guessing, to construct strong learners that can generate precise predictions.

2.2. Ensembles to improve the accuracy of CNNs

Ensembles are often used to enhance the target detection or classification accuracy of CNNs across many applications. To acquire a variety of base models, one may use various training data or employ alternative architectures [19,20]. Sirinukunwattana et al. [19] used images with different pixel offsets to train the same CNN structure which can generate diverse base CNNs to enhance the classification accuracy. Similarly, different cross-sections of a lung nodule trained a CNN structure to generate multiple CNNs in [20]. Other research uses the same dataset to train CNNs with varying architectures. In [21], AlexNet, GoogLeNet and ResNet are trained based on the same dataset to improve the food classification accuracy. Duan et al. [22] used an ensemble of AgeNet, RaceNet and GenderNet which are trained by the same image net to make age estimation. In addition, Ding et al. [23] and Pons et al. [24] used distinct datasets and CNN architectures concurrently to enhance the accuracy of face recognition and facial emotion categorization, respectively.

Two key characteristics are evident in previous studies that focus on improving classification accuracy through ensemble methods. (1) the factor of complexity is usually neglected in the base model; (2) base models are expected to complement each other, and the failure of base models will be discarded. This paper uses the same method as [25] to assess ensemble diversity, which calculates the dissimilarity between any two learners and then subsequently computes the average dissimilarity over all pairs for a comprehensive evaluation. Typically, greater diversity is associated with improved ensemble performance.

2.3. Application of ensemble learning in edge AI

Some research has considered applying ensemble learning to edge intelligence. Chang et al. [26] promoted the EEoI approach, in which the inference results from different side branch classifiers are combined to further improve the task performance. However, the failure of a lower branch classifier will fail the whole system, and the diversity of each branch classifier, which shares weights, is not considered. Wang et al. [27] considered the face recognition application in the cloud–edge cooperative scenario, where each edge server independently trains a model with the same structure based on the diverse locally collected samples. However, the failure of a lower branch classifier compromises the entire system, and the diversity between branch classifiers, which share weights, is not taken into account.

To achieve the trade-off between diversity and the performance of base models, Chen et al. [28] proposed to exchange the part of data to increase each data size, which can further improve the ensemble performance. However, aside from data diversity, ensemble performance depends on the performance of individual base learners, which is influenced by the size of the training set.

3. System model

3.1. Basic idea and system architecture

As introduced above, the focus of this paper is to explore the use of ensembles to enable high-performance AI services based on resource-limited edge servers and improve the fault tolerance capability of the system. First, only a single edge server around a user may fail in most cases. Hence, the distribution and deployment of varied models surrounding a user ensures that the loss of any one edge server will have little impact on task execution. Furthermore, the collection of weak classifiers might attain comparable or even superior performance compared to a strong classifier due to its variety. Using a variety of low-complexity base models makes it possible to achieve accuracy comparable to that of more complex models. Ensemble-based edge intelligence is a viable option for balancing the limitations of edge resources with the need for high-performance capabilities.

In this paper, we consider that there is sufficient data collected in the cloud for model training, and the system needs to provide a low-latency AI service to end users. For such an application scenario, we proposed a cloud–edge-end integrated AI system and a user-centric approach for the service. Fig. 3 illustrates the fundamental operations of an AI service based on the proposed method, which are outlined as follows:

1. Cloud-based training is used for heterogeneous models.
2. The models are deployed on edge servers that are specifically selected to meet the complexity of the model.
3. The AI service receives input from the end user, such as a picture to be classified.
4. Every edge server operates independent inference upon the input and sends its results to the end user.
5. The end user combines all the results to obtain the final result.

This design fully accounts for the complexity requirements at each stage of the ensemble-based AI service, while considering the unique characteristics of cloud computing, edge computing, and end users. First, it is important to recognize that model training is highly demanding in terms of time and resources. Given the extensive computational resources available in the cloud, it is advisable to carry out model training on the cloud platform. While model training is more complex, the inference process is relatively simpler. However, due to the need for quick response times, the inference process is executed on edge servers that have limited resources and are near the consumers. Furthermore, due to the straightforward nature of the result combination, the computational capacity on the end-user's side is sufficient to obtain the final outcome without any further delay.

In principle, the models on edge servers could be homogeneous or heterogeneous. We additionally discuss this issue in Section 2.2 and shown by the case studies in Section 6, heterogeneous models are preferred for higher diversity. Besides, the dataset is used in any

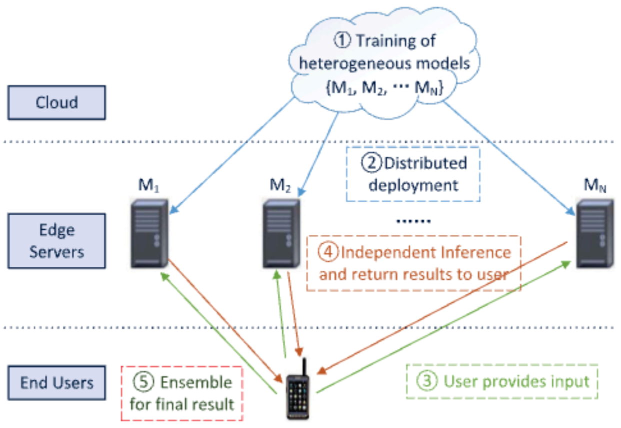
① Training of heterogeneous models {M_1, M_2, ⋯ M_N}
Cloud
②Distributed deployment
M_1
M_2
M_N
Edge Servers
④Independent Inference and return results to user
End Users
⑤ Ensemble for final result
③ User provides input

Fig. 3. Cloud-edge-end integrated scheme for AI service.

way to train the models in the cloud, including the ones in [28,29]. The experimental study in Sections 6.2.3 and 6.3.3 further shows that using all data for the training of each base model achieves the best performance.

3.2. Diversity-based user-centric deployment

Compared with the cloud-centric approach in [26,29], our design is user-centric so that the ensemble procedure introduces no delay. To fully explore the model diversity for ubiquitous AI service for all users, a diversity-based deployment of heterogeneous models is proposed.

The basic idea is borrowed from the spectrum reuse scheme for mobile cellular networks [30]. As shown in Fig. 4, the models (indexed by the number) are distributed throughout the serving region (the hexagons), so that diverse models always surround a user (the red point). For example, if the edge servers are so weak that the ensemble 7 models are needed to provide a service meeting the performance requirement, the models can be deployed as shown in Fig. 4(a). As we can see, an end-user in any position can access the 7 models on the adjacent 7 edge servers. If the edge servers are stronger, the ensemble of 4 or 3 stronger models would provide good enough service. Then the base models could be deployed according to the patterns in Figs. 4(b) and 4(c), respectively.

The deployment topology presented in Fig. 4 serves as an illustrative example of user-centric deployment, but it is not a strict requirement for the effectiveness of the proposed methods. The methods are designed to be adaptable to various deployment topologies, depending on the specific network and user requirements. In real-world scenarios, achieving the exact deployment depicted may pose practical challenges, such as infrastructure limitations and varying user densities. However, the core principle of the approach is to optimize resource allocation and performance in a user-centric manner which can be applied across different topologies to achieve similar benefits.

Considering that the base stations (BSs) in mobile networks are also deployed in this way for spectrum reuse, and edge servers are very likely to be deployed at the BS in the future [31], the proposed deployment of diverse models matches the trend of edge computing and mobile wireless networks very well. It can be observed from Fig. 4 that there are multiple copies of the same model deployed around the user but at different distances. When a model on the closest edge server fails, the user can access the same model from its neighbor edge server. Due to this property, the failure of multiple models can also be tolerated, which tremendously improves the system's reliability. In other words, the proposed edge intelligence scheme is double protected in two ways. One is the ensemble of multiple models, and the other is the redundant deployments of each model. Since dense small cells would be applied in 5G or beyond [32], edge servers would be located closer to the users.

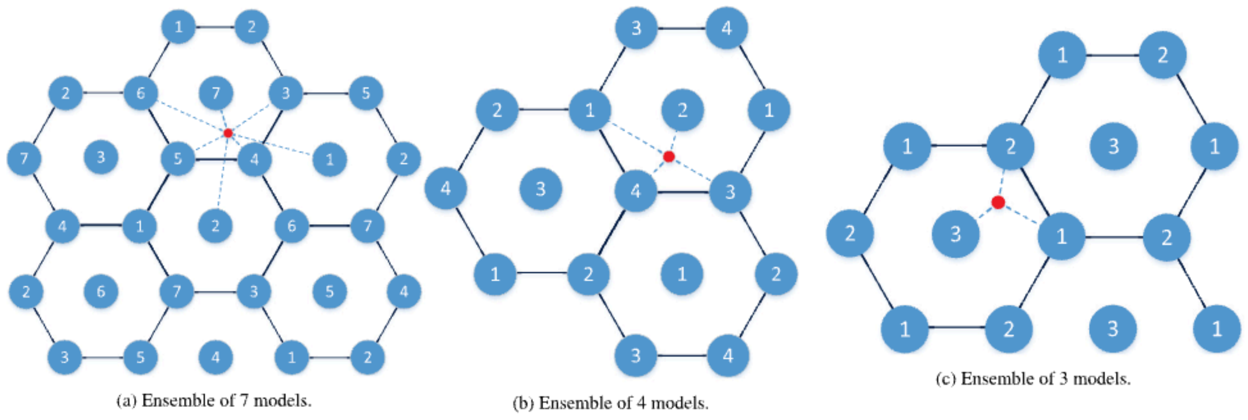
(a) Ensemble of 7 models. (b) Ensemble of 4 models. (c) Ensemble of 3 models.

Fig. 4. Optimized distributed deployment of diverse models.

Then redundant deployment could improve service quality and system reliability at the same time.

The users who publish tasks are deemed as $U = \{1, 2, \dots, N_U\}$. The set of edge servers that deploy the same type of model is considered as $S_m(m = 1, 2, \dots, M)$, where the superscript m indicates the classes of diverse models selected by users. Then all servers performing tasks can be represented as $S = \{S_1, S_2, \dots, S_M\}$. For any user n, it can select N_s^n edge servers, expressed as $S_n = \left\{S_1^n, S_2^n, \dots, S_{N_s^n}^n\right\} \in S$. For the uplink between edge server S_i^n $(i = 1, 2, \dots, N_s^n)$ and user n, we suppose P_{ni}^{UL} is the transmitting power, Γ_{ni}^{UL} is the received noise power and B_{ni}^{UL} is the channel bandwidth. Therefore, the uplink rate r_{ni}^{UL} between user n and edge server S_i^n can be obtained as:

$$r_{ni}^{UL} = B_{ni}^{UL} log(1 + \frac{P_{ni}^{UL}}{\Gamma_{ni}^{UL}}) \tag{1}$$

Similarly, for the downlink between edge server $S_i^n (i = 1, 2, \dots, N_s^n)$ and user n, we suppose P_{ni}^{DL} is the transmitting power, Γ_{ni}^{DL} is received noise power and B_{ni}^{DL} is the channel bandwidth. Therefore, the downlink rate r_{ni}^{DL} is

$$r_{ni}^{DL} = B_{ni}^{DL} log(1 + \frac{P_{ni}^{DL}}{\Gamma_{ni}^{DL}}) \tag{2}$$

We consider that users send tasks with the same total bit length b_{UL} and edge servers return data with the same length b_{DL}. Since the model deployed on each edge server matches its computing power, the task processing time is assumed to be t_p. Therefore, the uplink time between user n and the edge server S_i^n is

$$t_{ni}^{UL} = \frac{b_{UL}}{r_{ni}^{UL}} \tag{3}$$

The downlink time between user n and the edge server S_i^n is

$$t_{ni}^{DL} = \frac{b_{DL}}{r_{ni}^{DL}} \tag{4}$$

According to Eqs. (3) and (4), the total communication time between user n and edge server S_i^n is

$$t_i^n = t_{ni}^{UL} + t_p + t_{ni}^{DL} \tag{5}$$

Considering all edge servers S_n are selected by user n, the communication time of user n to the edge server is

$$t_n = \max\left\{t_i^n\right\} \tag{6}$$

4. Problem formulation

The total time to complete the task t_n increases as the distance between the user n and the edge server S_i^n increases. For users, they want to get task results with the minimum power consumption and high performance in the shortest time. For edge servers, intelligent tasks are expected to be completed with minimal power consumption. To this end, this article aims to minimize the total performance time t, which can be expressed as a min–max joint optimization problem. That is, each user should select a group of edge servers that can complete the task so that all users can complete the task in the shortest time. In this way, we have the following expression:

$$\begin{aligned} \min_{S_n} \max_{s} \quad & \sum_{n=1}^{U}\sum_{i=1}^{N} s(t_{ni}^{UL} + t_p + t_{ni}^{DL}) \\ s.t. \quad & n \in U, i \in S_i^n \\ & s = \{0, 1\} \end{aligned} \tag{7}$$

In addition to power, bandwidth and communication duration, user n also needs to consider the diversity and accuracy of edge servers for performance when selecting S_n. The accuracy of different models and their diversity to each other can be determined after training in the cloud. The model accuracy of the same structure is similar, but the model accuracy of different structures is different. The diversities between the models of the same structure are close and relatively small, while the diversities between the models of different structures are relatively large.

We express the model accuracy of different structures as $A_m(m = 1, 2..., M)$, the diversity among the same structural models is expressed as $d_m(m = 1, 2...M)$, the diversity between class p and q models is expressed as $d_{pq}(p, q = 1, 2..., M$ and $p \neq q)$. Note that A_m, d_m and d_{pq} are all within $(0, 1)$. The improvement of task performance by ensemble learning is related to the current accuracy and the number of base models, where better single model performance and a higher number of diverse base models generally contribute to better ensemble accuracy. Based on this, the ensemble accuracy of models with the same structure is

$$A_m^{EL} = A_m + \alpha \sum_{i=2}^{N_m} d_m^{i-1} \tag{8}$$

where $\alpha \sum_{i=2}^{N_m} d_m^{i-1}$ accounts for the incremental improvement in accuracy derived from the diversity of additional models. Here, α is a scaling factor that ensures the ensemble accuracy remains bounded. As N_m increases, the diversity parameter d_m^{i-1} gradually diminishes, reflecting diminishing returns in ensemble accuracy as more models are included.

To measure the diversity between base models, we calculate the pairwise dissimilarity of their outputs on the same input dataset. For classification tasks, it is defined as the proportion of input samples where two models produce different predictions. Alternatively, for probabilistic outputs, the dissimilarity can be calculated as the average absolute difference in predicted probabilities. The overall diversity of the ensemble is then obtained by averaging the pairwise dissimilarities across all model pairs. This approach ensures that higher diversity corresponds to greater heterogeneity among the models, which is beneficial for ensemble performance, as noted in [25]. Similarly, the ensemble accuracy of models with different structures can be modeled as Eq. (9).

$$\begin{aligned} A^{EL} &= \max(A_{m_p}) + \alpha(d_{m_1m_2} + (\frac{d_{m_1m_3}d_{m_2m_3}}{2})^2 \\ &+ (\frac{d_{m_1m_4}d_{m_2m_4}d_{m_3m_4}}{3})^3 + \cdots \\ &+ (\frac{1}{N_E}\sum_{i=1}^{N_E-1} d_{m_im_{N_E}})^{N_E}) \end{aligned} \tag{9}$$

where N_E is the number of models of different ensemble structures, $m_p \in [1, 2, \ldots, M]$, where $p = \{1, 2, \ldots, N_E\}$ represents the serial number of different model structures. In fact, when $m_q = m$ and $d_{m_im_j} = d_m$, Eq. (8) is a special case of Eq. (9). Therefore, the ensemble accuracy of all models selected by a user (including different architectures and the same architectures) can be obtained directly based on Eq. (9). From the perspective of accuracy, each user selects a group of edge servers, and the optimal scheme selected by all users maximizes the average value of all users' task accuracy, which can be expressed as follows:

$$\max_{S_n} A_{avg} = \max_{S_n} \frac{1}{N_u}\sum_{n=1}^{N_u} A_n^{EL} \tag{10}$$

5. Ensemble deep reinforcement learning design

To address the above problem, we first define some crucial equations for the Markov decision process to model the process of users selecting edge servers. Then the edge servers selection algorithm is obtained. Finally, the generation of the base models ensemble for edge intelligence is introduced.

5.1. Markov decision process

To better model the edge server selection process, we first define some related equations, such as the total upstream and downstream power, the total upstream and downstream bandwidth, and the total delay of the system, shown as Eqs. (11) to (15) respectively. To improve the overall performance of a system, it is necessary to reduce the power and bandwidth loss, reduce the delay and increase the accuracy.

$$P_{UL} = \sum_{n=1}^{N_u}\sum_{i=1}^{N_n^s} P_{ni}^{UL} \tag{11}$$

$$P_{DL} = \sum_{n=1}^{N_u}\sum_{i=1}^{N_n^s} P_{ni}^{DL} \tag{12}$$

$$B_{UL} = \sum_{n=1}^{N_u}\sum_{i=1}^{N_n^s} B_{ni}^{UL} \tag{13}$$

$$B_{DL} = \sum_{n=1}^{N_u}\sum_{i=1}^{N_n^s} B_{ni}^{DL} \tag{14}$$

$$T_{all} = \max_{n\in U}\{t_n\} \tag{15}$$

Based on the analysis, while it is possible to reduce delay by moving closer to specific edge servers, this typically results in accessing a smaller subset of the available models, thus limiting diversity. The inherent trade-off between minimizing delay and maximizing diversity remains because achieving large diversity generally requires accessing models distributed across more distant servers, which can introduce additional delay. Therefore, in most scenarios, small delay and large diversity cannot be fully achieved simultaneously. To analyze how to improve system performance, we use the deep reinforcement learning method.

It is worth noting that when deploying ensemble models on edge computing devices, there are notable differences in latency compared to centralized or cloud-based systems. While edge devices may have limited computational resources, leading to longer processing times for complex ensemble models, they benefit from reduced transmission latency due to their proximity to the user. In scenarios where network latency is the primary bottleneck (*e.g.,* high-bandwidth or real-time applications), edge computing significantly outperforms centralized systems by lowering the overall delay. However, for computationally intensive tasks, there can be an increase in local processing time. Therefore, the choice of where to deploy ensemble learning, on the edge or in the cloud should depend on the specific use case, whether prioritizing fast processing or minimizing network latency.

Therefore, to analyze how to improve system performance, we use the deep reinforcement learning method. The definition of environment, state, action and reward is very important in deep reinforcement learning.

System environment: We consider that there are $S_m = \{1, 2, \ldots, m, \ldots, M\}$ models with different structures. The Model distribution $[M_1, M_2, \ldots, M_{N_s}]$ represents the deployment of models of different structures on edge servers. In addition, the environment contains the transmission power and channel bandwidth required for each user to communicate with each edge server.

System state: If the edge server S has been selected by a certain user n, represented as $s \in \{0, 1\}$, $s = 1$ means selected, 0 is otherwise.

System action: If the user n selects edge server S, represented as $a \in \{0, 1\}$, $a = 1$ means selected, 0 is otherwise.

System reward: The reward function is designed to balance various performance metrics, including time consumption, power, bandwidth, and accuracy.

$$R_{tmp} = \frac{\mu T_{all} + \beta(P_{UL} + P_{DL}) + \lambda(B_{UL} + B_{DL}) + \theta}{A_{avg}} \tag{16}$$

where, μ, β, λ, and θ are coefficients. These coefficients may vary depending on the scenario. The A_{avg} is served to penalize lower accuracy, ensuring that the immediate reward R_{tmp} decreases as accuracy drops. To ensure the objective of minimizing the average time consumption of the system and achieving the maximum average task accuracy, we hereby define the system reward by using the negative exponential function to normalize the R_{tmp}, *i.e.,* $R = e^{-R_{tmp}}$ which ensures it is bounded and emphasizes differences between high and low rewards.

5.2. Base models ensemble generated for edge intelligence

5.2.1. Basic principle

Compared to ensembles focused solely on improving accuracy, ensemble-based edge AI introduces two key differences in base model design. First, accuracy-focused ensembles pay little attention to model complexity, whereas edge intelligence requires simpler base models to fit within the limited resources of edge servers. Second, the accuracy-oriented design tends to have base models as diverse as possible. In this case, if one of the base models is not available (*e.g.,* edge connection breaks or server crashes), the accuracy of the ensemble will decrease significantly. To enhance the reliability of the ensemble-based edge AI system, intentional similarity among base models is essential. This ensures that even if one model fails, the ensemble can maintain high accuracy. In general, to meet the requirements for performance, resource constraints on edge and system reliability at the same time, the ensemble system prefers to have more and simpler base models, so that the failure of a single base network has negligible influence on the task performance. This principle differs from the approach used in accuracy-focused ensembles.

Table 1
Parameter setting (Part I).

Parameter	Value	Parameter	Value	Parameter	Value
A_1	0.100	A_2	0.100	A_3	0.100
A_1	0.100	A_2	0.100	A_3	0.100
d_1	0.100	d_2	0.101	d_3	0.099
d_{12}	0.110	d_{23}	0.111	d_{13}	0.109
μ	0.500	β	0.500	λ	0.500
θ	0.100	α	0.100		

Table 2
Parameter setting (Part II).

3 ESs	B_{UL}, B_{DL}:500 Hz–1000 Hz; P_{UL}, P_{DL}:1 W–1.3 W
3 other ESs	B_{UL}, B_{DL}:1000 Hz–1500 Hz; P_{UL}, P_{DL}:1.3 W–2 W
6 other ESs	B_{UL}, B_{DL}:1500 Hz–2000 Hz; P_{UL}, P_{DL}:2 W–3 W
The other ESs	B_{UL}, B_{DL}:2000 Hz–3000 Hz; P_{UL}, P_{DL}:3 W–5 W

5.2.2. Low-cost heterogeneous models generated

Based on the principles discussed earlier and the observation that diverse structures efficiently generate variety, we propose two methods for constructing base models for edge intelligence. One approach is to use models that have the same unit structure but vary in the number of layers. The second method uses networks with different compression rates while maintaining the same complex architecture. These techniques naturally include both diversity and similarity in the base networks.

Within the realm of popular deep neural networks (DNNs), several models include identical unit structures but vary in terms of their layers. ResNets consist of many networks that share the same residual module but have varying numbers of layers. For CIFAR-10, these layers are {20, 32, 44, 56, or 110}, while for ImageNet, they are {18, 34, 50, 101, or 152}. GoogLeNet is built by stacking many Inception modules together. To address the issue of gradient vanishing that occurs throughout the training process, softmax layers are included in the output of some Inception modules. Conversely, network compression is a more straightforward approach to creating models with varying levels of complexity. In general, networks that are deeper or less compacted are anticipated to provide superior task performance.

The model segmentation strategy involves selecting the model with the highest performance and dividing it into many parts for dispersed deployment on the edge. Alternatively, we choose several simpler models that have lesser performance and rely on ensemble methods to get high performance. This technique allows for the adaptive selection of a base model with varying complexity based on the resource limitations of the edge server. By utilizing edge servers with comparable resources, these two approaches can be combined to create heterogeneous structures with similar levels of complexity. As an example, ResNet 20, 32, and 44 may be compressed with pruning rates of 20%, 30%, and 40%, respectively.

While the primary focus of this study is on validating the cloud–edge-end integrated system architecture and model deployment strategies, the experimental results presented in Figs. 5 and 6 are consistent with the goals of the ensemble DRL design. The choice to focus on these figures, rather than DRL-specific evaluations, was made to establish the foundational performance metrics related to system delay and reliability. The improvements demonstrated in these areas through optimized deployment and resource allocation strategies suggest that the DRL method, which aims to dynamically optimize these aspects, would likely be effective in further improving the overall performance of the system. Although a direct empirical validation of the RL method is not included in this study, the alignment between the results and the objectives of the RL method provides a strong foundation for its proposed application.

Table 3
Experimental results.

No. of users	1	2	3	4	5	6
Delay	1.050	1.058	1.058	1.111	1.106	1.111
Power	35.935	64.762	96.502	118.515	145.752	171.799
Bandwidth ($\times 10^5$)	0.224	0.431	0.651	0.812	0.987	1.145
Accuracy (%)	92.210	92.260	92.207	92.230	92.208	92.223

6. Case studies for ensemble-based edge intelligence

6.1. Experimental setup

Tables 1 and 2 show the settings of experimental parameters, and we employ 3 models in the experiment. In addition, we set the maximum number of edge servers selected per user to 5, and the number of users is less than 7.

Table 3 highlights the trade-offs between delay, power consumption, bandwidth usage, and accuracy across different system configurations. Based on the results, as the number of users increases (*i.e.,* from 1 to 6), the delay rises from 1.050 s to 1.111 s, power consumption increases from 35.935 W to 171.799 W, and bandwidth increases from 0.224×10^5 to 1.145×10^5. The accuracy remains relatively stable, ranging from 92.207% to 92.260%. The results show that reducing delay often increases power and bandwidth usage due to the need for faster processing and communication. Conversely, minimizing power and bandwidth typically leads to higher delays. Accuracy, while generally improved with higher resource usage, can be maintained at an optimal level through careful balancing of these factors. This analysis underscores the optimization problem's goal of minimizing delay, power, and bandwidth while maximizing accuracy, demonstrating the importance of finding configurations that best navigate these trade-offs.

Furthermore, we present two case studies that demonstrate the efficacy of ensemble-based edge intelligence, as shown by experimental data. The first ensemble consists of ResNets with varying layers, whereas the second ensemble consists of pruned VGGNets with varied pruning rates. Each case study begins by introducing the implementation of basic models, followed by the execution of three kinds of analysis.

The first objective is to demonstrate that a collection of numerous weak base networks may achieve comparable or superior performance compared to a complex network with similar total computational complexity. The second objective is to demonstrate the reliability of the ensemble system. The outcomes of these two components may demonstrate the superiority of the proposed scheme compared to the model segmentation system. Additionally, the impact of using different numbers of data on the overall performance of the ensemble is also shown. The findings are derived exclusively from the PyTorch framework using the CIFAR-10 dataset. The dataset comprises 60,000 images distributed over 10 categories. Specifically, 50,000 images are allocated for training purposes, while the remaining 10,000 images are designated for testing. Furthermore, the ensemble strategy used in the subsequent tests is averaging the category scores, since the base models exhibit similar performance. The results were derived from multiple independent runs to ensure statistical significance and reliability.

Initially, we conducted the ensemble learning by using ResNet-20 models. The optimization framework is designed to minimize delay across the cloud–edge-end architecture, balancing this objective with the need for efficient resource allocation.

Fig. 5 validated the system's efficiency in utilizing low-complexity base models at the edge and demonstrates that the ensemble accuracy surpasses 93% when there are 3 ResNet-20 models. However, even with 10 ResNet-20 models, the ensemble accuracy remains below 94%. The performance exhibits a consistent upward trend as the number of models in the ensemble grows. Specifically, the accuracy dramatically rises from 1 to 2 and stabilizes from 3 onwards. It is demonstrated

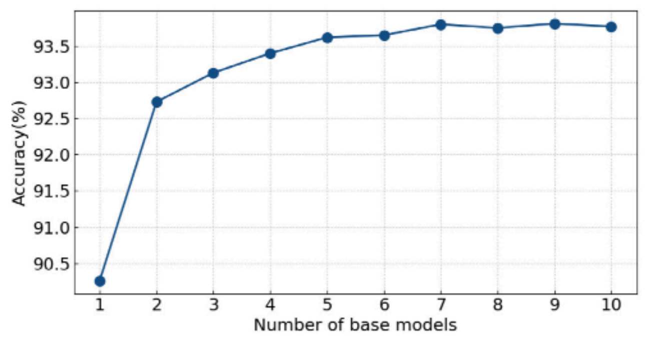

Fig. 5. Performance of the ensemble of multiple ResNet-20.

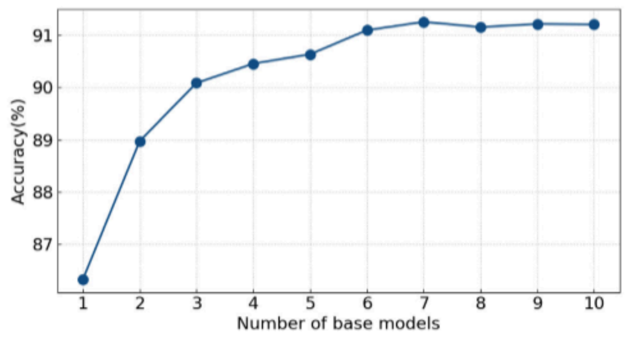

Fig. 6. Performance of the ensemble of multiple VGG16 with 0.2 pruning rate.

Table 4
Performance and complexity of base ResNets.

Base ResNets	20	32	44	56	110
Accuracy (%)	90.27	90.88	91.17	91.8	91.96
Parameters (M)	0.27	0.46	0.66	0.85	1.7
Complexity ($\times 10^7$)	4.09	6.95	9.80	12.66	25.5

that the proposed architecture achieves high accuracy with minimal resource consumption, highlighting its ability to balance performance and efficiency in edge scenarios. The results validate the system's design principle of distributing lightweight models across edge servers to reduce latency and computational burden. Hence, in the following scenarios, we use a combination of three different ResNet models for our investigation.

Fig. 6 highlights the performance of pruned VGG16 models, which were tested to evaluate the system's adaptability to resource-constrained edge environments, demonstrating that when the number of VGG16 models is 3 and a pruning rate of 0.2 is used, the ensemble process exhibits a consistent accuracy trend. The curve representing the accuracy has gone beyond the first rapid development stage and is approaching a plateau. The results demonstrated that even with significant pruning, the ensemble accuracy remained above 90%, validating the architecture's efficiency in maintaining high accuracy while reducing computational complexity. This experiment underscores the system's ability to effectively utilize limited edge resources, making it suitable for low-latency AI applications in dynamic edge environments. Thus, we selected three pruned VGG16 models for the subsequent case study.

6.2. Ensemble of shallow ResNets

6.2.1. Implementation of ResNets with different layers

This case study utilizes an open-source PyTorch code for ResNet [33]. Each ResNet in this part is trained using all the images in the training set. As stated in Ref. [7], the weight decay and momentum values are assigned as 0.0001 and 0.9, respectively. The initial learning rate is set at 0.1 and is then reduced by a factor of 10 at the 32,000 and 48,000 iteration marks.

Table 4 presents the values for the volume of parameters, computation complexity, and classification accuracy on CIFAR-10 for ResNets with varying numbers of layers, given the specified conditions. The complexity is quantified by the number of floating point multiplications over all convolution layers and fully connected layers. According to Table 4, the ResNet-20 model has an accuracy of 90.27%, which increases when additional layers are added to the ResNet models. The ResNet-110 model reaches an accuracy of 91.96%. Furthermore, the magnitude of parameters and computational complexity have a roughly linear relationship with the number of layers.

Table 5
Performance and complexity of ensemble of multiple ResNets. # 1 is the ensemble of ResNet-20, ResNet-32 and ResNet-56. # 2 is the ensemble of ResNet-20, ResNet-44 and ResNet-56. # 3 is the ensemble of ResNet-32, ResNet-44 and ResNet-56. # 4 is the ensemble of three ResNet-56s.

Index	Ensemble scheme	Complexity	Accuracy	Diversity
# 1	20 + 32 + 56	23.7	93.22%	6.35%
# 2	20 + 44 + 56	26.5	**93.5%**	6.23%
# 3	32 + 44 + 56	29.4	**93.5%**	**6.45%**
# 4	56 * 3	**37.98**	93.49%	6.18%

6.2.2. Performance of model diversity for ResNets

If the model segmentation approach is applied for edge intelligence with ResNets, ResNet-110 will be selected to be segmented. In our case, different ensembles can be constructed based on the ResNets with fewer layers. Table 5 displays the complexity and accuracy of four ensemble strategies. The first three schemes use ResNets with varying layers as foundational models, while the fourth strategy serves as a benchmark system, using three ResNet-56 models with independently randomized weights.

From Table 5, it is evident that all the ensemble frameworks outperform ResNet-110 in terms of accuracy. The performance enhancement achieved by the ensemble approach is confirmed using two distinct methods. The total complexity of the first three schemes is similar to that of ResNet-110, indicating that the heterogeneity of the models effectively enhances task performance without increasing complexity.

Furthermore, the accuracy of the first three strategies is comparable to or greater than the combined accuracy of three ResNet-56 models. This demonstrates that the variation across dissimilar models may contribute to sustaining classification accuracy while reducing overall complexity.

To provide proof for the second conclusion, diversity analysis is performed on each of the ensemble schemes according to Eq. (1), and the results are listed in the last column of Table 5. As we can see, the diversity for the first three schemes is higher than that for the last scheme.

6.2.3. Effect of data differences for ResNets

To examine the impact of data discrepancies on heterogeneous base models, we perform additional experiments to assess the level of variety they bring. Specifically, the data exchange strategy suggested in [33] is used to modify the overlap ratio within the training data set for each base model. The classification accuracy for each ensemble scheme with varying sizes of the training data set is shown in Fig. 7. It illustrates the classification accuracy of different ensemble schemes under varying training dataset sizes. The results demonstrate the impact of resource allocation strategies on system performance, particularly in terms of maintaining classification accuracy across different dataset conditions. This improvement is shown even in the case of the final scheme, which combines three ResNet-56 models.

The performance of an ensemble system depends on two factors: the diversity among base models and the performance of each model.

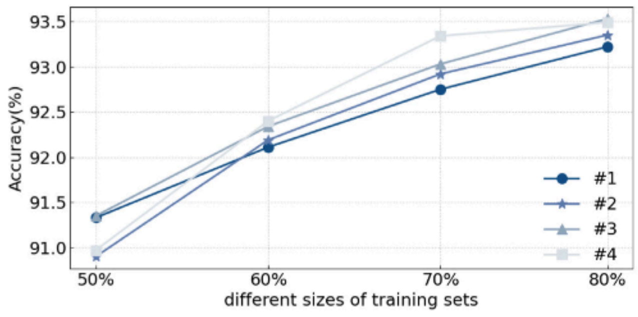

Fig. 7. Accuracy of ensemble schemes with different sizes of training sets.

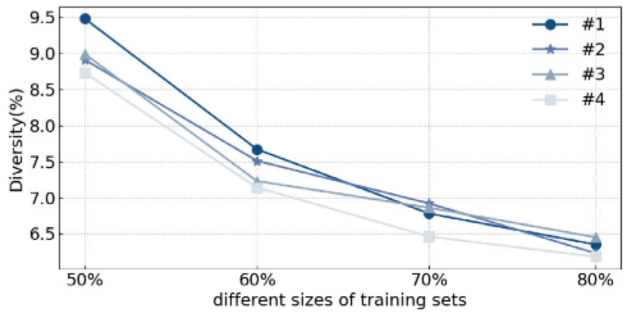

Fig. 8. Diversity for ensemble schemes with different sizes of training sets.

Table 6
Accuracy of VGG16 with different pruning rates.

Pruning rate	0	0.5	0.6	0.7	0.8
Base network	VGG-16	VGG-50	VGG-40	VGG-30	VGG-20
Acc. (%)	87.68	87.64	87.12	87.03	86.27
Weights (M)	38	19	15.2	11.4	7.6

Table 7
Performance of ensembles of pruned VGG16.

Index	#1	#2	#3	#4
Ensemble	$20+30+40$	$20+30+50$	$40*3$	$50*3$
Accuracy	90.41%	**90.59%**	90.37%	90.5%
Diversity	**9.96%**	9.79%	9.26%	9.27%

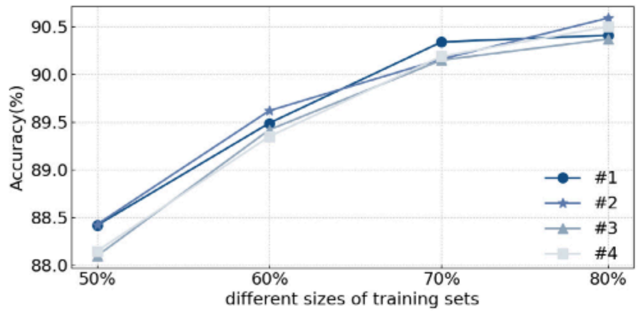

Fig. 9. Accuracy of ensemble schemes with different sizes of training sets.

Greater diversity ensures that models contribute complementary predictions, enhancing the ensemble's overall accuracy. At the same time, high-performing base models ensure that the ensemble's baseline performance remains strong. Balancing these two factors is critical to optimizing the ensemble's effectiveness. Fig. 8 illustrates the trade-off between diversity and accuracy in ensemble systems as the size of training sets increases. Smaller training sets lead to greater diversity among base models due to higher variations in the data, but this also results in lower accuracy for each model. However, this leads to a fall in the accuracy of each base model. In this scenario, the decline in the accuracy of the base models has a more significant impact on the overall performance of the ensemble compared to the advantage gained by increased variety.

6.3. *Ensemble of pruned VGGNets*

6.3.1. *Implementation of VGGNets with different pruning rates*

This case study utilizes the VGG16 architecture, which consists of 13 convolution layers and 3 fully connected layers. The construction of the model is done using an open-source PyTorch code specifically designed for the CIFAR-10 dataset [34]. In this assessment, the weight decay and momentum values are set to 0.0001 and 0.9, respectively. The initial learning rate is set to 0.1 and is reduced by a factor of 10 at iterations 30, 60, and 90. The training process concludes after 100 iterations.

In this part, we use the widely used magnitude-based pruning technique introduced in [35] to create a pruned network. The pruning procedure comprises four sequential steps: (1) Perform training on the CIFAR-10 dataset using the original VGG16 model. (2) Arrange all the weights based on their magnitude; (3) Eliminate the specified fraction of the weights; (4) Adjust the weights of the remaining connections to optimize their performance. The performance of the compressed networks with various pruning rates is shown in Table 6, according to this technique. The last row displays the volume of leftover weights for various pruning rates, which is directly related to the computational complexity. As anticipated, the network that undergoes a larger pruning rate exhibits a decrease in accuracy. However, the decline is very insignificant, even when 70% of the weights are pruned.

6.3.2. *Performance of model diversity for VGGNets*

This section evaluates four different ensemble strategies. The first two approaches use pruned VGG16 models with varying rates of pruning, while the latter two serve as reference models that utilize base models with identical pruning rates (with independently randomized weights for training).

The accuracy of the four different ensemble schemes is shown in Table 7. Similar to the first case study, the ensemble of heterogeneous networks (# 1 or # 2) achieves better performance than that of homogeneous networks (# 3 or # 4) with much lower overall complexity. This could be explained by the diversity measures shown in the last row of Table 7. As we can see, the diversity measures for scheme # 1 or # 2 are higher than that for scheme # 3 or # 4 which proves that the diversity in the ensemble system compensates for the performance loss of individual base models, so that the classification accuracy can be maintained with less overall complexity.

6.3.3. *Effect of data differences for VGGNets*

The accuracy and diversity of each ensemble method are evaluated using varying sizes of the training set. The corresponding outcomes are shown in Figs. 9 and 10, respectively. It is evident that as the size of the training data set rises, the accuracy of the ensemble system improves, but the diversity between the base networks diminishes. This research demonstrates once again that the enhancement in accuracy of the base models has a greater impact on the overall performance of the ensemble than the added variation resulting from changes in the data.

6.4. *Fault tolerance capability*

With the proposed ensemble based architecture, when one of the edge servers fails, other servers can still provide the AI service cooperatively. In this case, the reliability is measured by the accuracy of the ensemble of remaining base models. Specifically, for each ensemble scheme in Tables 5 and 7, each based model is excluded from the ensemble respectively, and the average accuracy of the ensemble of the remaining base models is measured. Taking the ensemble scheme #1 (ResNet-20, ResNet-32, and ResNet-56) as an example, we average the

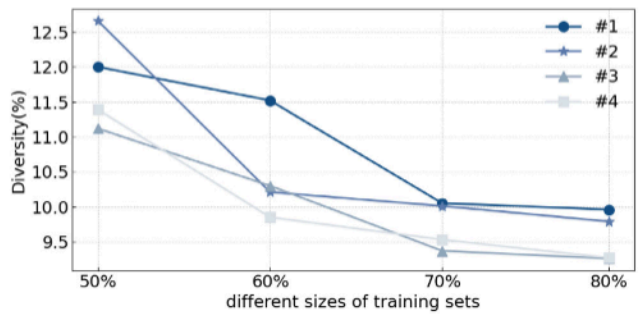

Fig. 10. Diversity of ensemble schemes with different sizes of training sets.

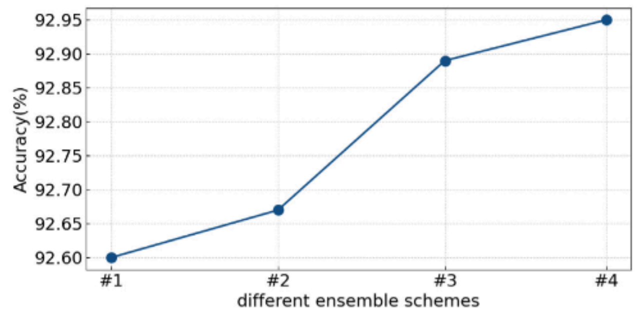

Fig. 11. Reliability for an ensemble of different ResNets.

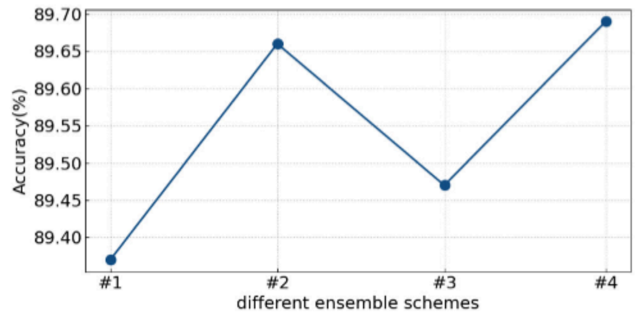

Fig. 12. Reliability for an ensemble of pruned VGGNets.

ensemble accuracy of (ResNet-20, ResNet-32), (ResNet-20, ResNet-56) and (ResNet-32, ResNet-56) as the reliability metric for this scheme.

The results for ensemble of diverse ResNets are shown in Fig. 11. As we can see, the performance with one base model excluded is still higher than that of ResNet-110 (91.96% in Table 4). This means that the proposed ensemble scheme can still provide high quality service when one edge server fails. In addition, we can see that the scheme with higher complexity (the order of complexity is #4>#3 >#2>#1 as shown in Table 5) also provides higher accuracy. This is because the diversity between stronger base models is smaller than that between weaker base models (as shown in Fig. 8), so excluding one base model from the ensemble of strong ones introduce smaller performance degradation.

The results for ensemble of pruned VGGS are shown in Fig. 12. We can see that the ensemble accuracy with one edge server failure is also higher than the accuracy of the original VGG-16 (87.68% in Table 6). This confirms that the proposed ensemble based distributed architecture can effectively improve the reliability of the AI service.

7. Conclusions and future work

In this paper, a cloud–edge-end integrated platform is proposed for efficient and reliable edge AI based on the ensemble learning approach, where the three procedures in ensemble learning are assigned to the cloud, edge, and the user, respectively, according to their characteristics: the strong capability of the cloud is used for training heterogeneous base models; the edge run base models independently to provide instant responses to user input; the user obtains the final result by combining the edge responses. A diversity-based deployment scheme is proposed for a user-centric edge AI network and the principle for the generation of base models is discussed.

The results of the experiment from two case studies yield several significant findings: (1) A collection of weak base models can outperform a powerful model with the same overall complexity. (2) An ensemble of diverse models can achieve comparable performance to that of uniform models but with significantly reduced complexity. (3) The failure of an individual base model has minimal impact on overall performance. (4) In enhancing ensemble performance, model heterogeneity is more effective than variations in the data. The first evidence demonstrates that the suggested strategy effectively resolves the conflict between complexity and performance in the compression-based solution for edge AI. The second demonstrates that the inclusion of diverse heterogeneous models is crucial for enhancing performance. The last two demonstrate the benefits of the suggested scheme compared to the model segmentation solution and the ensemble solution, respectively, in terms of reliability and system performance based on data differences. The cloud–edge-end integrated scheme is highly adaptable to various edge AI applications in varied settings, and its effectiveness is enhanced by the progress of edge computing and mobile wireless networks.

While this study demonstrates the effectiveness of the proposed method using CIFAR-10, further research is needed to explore its generalizability to other datasets, such as ImageNet, and different model combinations. The method's flexibility suggests that it could be adapted to handle more complex data distributions and a variety of model architectures. Future work will focus on applying the method to these diverse scenarios to validate its broader applicability and significance in different contexts.

CRediT authorship contribution statement

Zhen Gao: Writing – review & editing, Data curation, Conceptualization. **Daning Su:** Writing – review & editing. **Shuang Liu:** Writing – original draft, Methodology, Formal analysis, Data curation. **Yuqi Zhang:** Writing – original draft, Methodology, Investigation, Formal analysis. **Chenyang Wang:** Writing – original draft, Methodology, Formal analysis, Conceptualization. **Cheng Zhang:** Writing – review & editing, Validation, Methodology. **Xiaofei Wang:** Writing – review & editing, Investigation, Conceptualization. **Tarik Taleb:** Writing – review & editing, Methodology, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Part of this work was presented at the IEEE/CIC International Conference on Communications in China, 11–13 August 2022, Foshan City, China, which is ted as Ref. [25]. This work was supported by the National Natural Science Foundation of China (grant number 62171313).

Data availability

No data was used for the research described in the article.

References

[1] A. Parvaiz, M.A. Khalid, et al., Vision Transformers in medical computer vision—A contemplative retrospection, Eng. Appl. Artif. Intell. 122 (2023) 106126.

[2] R. Collobert, J. Weston, et al., Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (ARTICLE) (2011) 2493–2537.

[3] J. Jeon, S. Lee, H. Choe, Beyond ChatGPT: A conceptual framework and systematic review of speech-recognition chatbots for language learning, Comput. Educ. (2023) 104898.

[4] W. He, Y. Chen, Z. Yin, Adaptive neural network control of an uncertain robot with full-state constraints, IEEE Trans. Cybern. 46 (3) (2015) 620–629.

[5] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[6] C. Szegedy, et al., Going deeper with convolutions, in: Proceedings of the IEEE CVPR, 2015, pp. 1–9.

[7] K. He, X. Zhang, et al., Deep residual learning for image recognition, in: Proceedings of the IEEE CVPR, 2016, pp. 770–778.

[8] M. Ravinder, G. Saluja, S. Allabun, M.S. Alqahtani, M. Abbas, M. Othman, B.O. Soufiene, Enhanced brain tumor classification using graph convolutional neural network architecture, Sci. Rep. 13 (1) (2023) 14938.

[9] H. Khelifi, S. Luo, et al., Bringing deep learning at the edge of information-centric internet of things, IEEE Commun. Lett. 23 (1) (2018) 52–55.

[10] X. Wang, C. Wang, X. Li, V.C. Leung, T. Taleb, Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching, IEEE Internet Things J. 7 (10) (2020) 9441–9455.

[11] Y. Chen, Y. Sun, C. Wang, T. Taleb, Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning, IEEE Internet Things J. 9 (18) (2022) 16742–16757.

[12] C. Wang, H. Yu, X. Li, F. Ma, X. Wang, T. Taleb, V.C. Leung, Dependency-aware microservice deployment for edge computing: a deep reinforcement learning approach with network representation, IEEE Transactions on Mobile Computing (2024).

[13] X. Wang, Y. Han, V.C. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (2) (2020) 869–904.

[14] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, J. Du, On-demand deep model compression for mobile devices: A usage-driven model selection framework, in: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, 2018, pp. 389–400.

[15] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, L. Tang, Neurosurgeon: Collaborative intelligence between the cloud and mobile edge, ACM SIGARCH Comput. Archit. News 45 (1) (2017) 615–629.

[16] Z. Zhao, et al., ECRT: an edge computing system for real-time image-based object tracking, in: Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, 2018, pp. 394–395.

[17] S. Teerapittayanon, B. McDanel, H.-T. Kung, Branchynet: Fast inference via early exiting from deep neural networks, in: 2016 23rd International Conference on Pattern Recognition, ICPR, IEEE, 2016, pp. 2464–2469.

[18] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2019.

[19] K. Sirinukunwattana, et al., Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images, IEEE Trans. Med. Imaging 35 (5) (2016) 1196–1206.

[20] P. Sahu, D. Yu, M. Dasari, F. Hou, H. Qin, A lightweight multi-section CNN for lung nodule classification and malignancy estimation, IEEE J. Biomed. Heal. Inform. 23 (3) (2018) 960–968.

[21] P. Pandey, A. Deepthi, B. Mandal, N.B. Puhan, FoodNet: Recognizing foods using ensemble of deep networks, IEEE Signal Process. Lett. 24 (12) (2017) 1758–1762.

[22] M. Duan, K. Li, K. Li, An ensemble CNN2ELM for age estimation, IEEE Trans. Inf. Forensics Secur. 13 (3) (2017) 758–772.

[23] C. Ding, D. Tao, Robust face recognition via multimodal deep face representation, IEEE Trans. Multimed. 17 (11) (2015) 2049–2058.

[24] G. Pons, D. Masip, Supervised committee of convolutional neural networks in automated facial expression analysis, IEEE Trans. Affect. Comput. 9 (3) (2017) 343–350.

[25] Z. Gao, Y. Zhang, W. Sun, Artificial intelligence service by satellite networks based on ensemble learning with cloud-edge-end integration, in: 2022 IEEE/CIC International Conference on Communications in China, ICCC Workshops, IEEE, 2022, pp. 158–163.

[26] Y. Chang, X. Huang, Z. Shao, Y. Yang, An efficient distributed deep learning framework for fog-based IoT systems, in: 2019 IEEE Global Communications Conference, GLOBECOM, IEEE, 2019, pp. 1–6.

[27] Y. Wang, T. Nakachi, Secure face recognition in edge and cloud networks: From the ensemble learning perspective, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2020, pp. 2393–2397.

[28] W. Chen, Y. Yu, et al., Time-efficient ensemble learning with sample exchange for edge computing, ACM Trans. Internet Technol. (TOIT) 21 (3) (2021) 1–17.

[29] Y. Qin, D. Wu, Z. Xu, J. Tian, Y. Zhang, Adaptive in-network collaborative caching for enhanced ensemble deep learning at edge, 2020, arXiv preprint arXiv:2010.12899.

[30] T.S. Rappaport, et al., Wireless Communications: Principles and Practice, vol. 2, prentice hall PTR, New Jersey, 1996.

[31] Y.C. Hu, M. Patel, et al., Mobile edge computing—A key technology towards 5G, ETSI White Pap. 11 (11) (2015) 1–16.

[32] H.R. Chi, A. Radwan, An overview of on-demand deployment optimization of small cells, IEEE Netw. (2020).

[33] Y. Idelbayev, Proper ResNet Implementation for CIFAR10/CIFAR100 in PyTorch that matches description of the original paper, available on Github at https://github.com/akamaster/pytorch_resnet_cifar10.

[34] L. Wang, G. Ding, Neural network pruning PyTorch implementation, available on Github at https://github.com/wanglouis49/pytorch-weights_pruning.

[35] S. Han, J. Pool, et al., Learning both weights and connections for efficient neural networks, 2015, arXiv preprint arXiv:1506.02626.