Towards Federated Inference: An Online Model Ensemble Framework for Cooperative Edge AI

Zhi Zhou[§] Jiajie Xie[§] Mengke Huang[§] Tao Ouyang[§] Fangming Liu[¶] Xu Chen[§]
[§]Key Laboratory of Machine Intelligence and Advanced Computing,
School of Computer Science and Engineering, Sun Yat-Sen University, China
[¶]Huazhong University of Science and Technology, and Peng Cheng Laboratory

***Abstract*—Edge inference leverages edge computing devices for the last mile delivery of artificial intelligence (AI) services. To meet the latency requirements while overcoming the resource limitations, edge inference systems deploy lightweight — typically compressed — DNN models. However, due to data drift during deployment, these compressed edge models often fail to deliver satisfactory and stable inference accuracy. To address this issue, we propose a novel edge inference serving paradigm called Federated Inference. This approach, based on ensemble learning, groups multiple edge workers to form an ensemble, enhancing inference accuracy. A key challenge in Federated Inference is maximizing ensemble accuracy while adhering to resource budgets and Service Level Objectives (SLOs). The dynamic nature of the environment and the NP-hardness of the optimization problem add to the complexity. To address these challenges, we propose an online model ensemble framework that integrates online learning with approximate optimization, offering a theoretically rigorous and computationally efficient solution. We have implemented a prototype of our framework and, through extensive test-bed evaluations, demonstrate that it improves average inference accuracy by $12\% \sim 80\%$.**

I. Introduction

As a dominant force in AI development, the landscape of big data has radically shifted from centralized warehouses to pervasive compute nodes like geo-distributed cloud/edge data centers and mobile/IoT devices. This decentralization necessitates extracting knowledge from decentralized datasets without shipping private and large-volume raw data. To this end, federated learning (FL) has emerged as a promising solution, enabling collaborative training of AI models from decentralized data [1]. In FL, each node trains a local model on its own data, and a central server aggregates these local updates to create a global model. By exchanging only model weights instead of raw data, FL enhances both privacy preservation and communication efficiency in AI model training. Thanks to these advantages, federated learning is poised to fully unleash the potential of pervasive decentralized big data.

Currently, FL has found realistic applications ranging from recommendation systems (e.g., Google Keyboard) [2] to smart homes (e.g., Amazon Alexa) and voice assistants (e.g., Apple Siri). While recognizing these initial successes, it should be noted that FL has not yet deeply penetrated some privacy-sensitive areas, such as credit scoring, finance, and healthcare. Although FL significantly improves privacy over traditional centralized approaches, it is not completely immune to privacy concerns. Recent research has shown that it is possible to reconstruct raw training data from model updates in certain scenarios, especially if the model or update mechanism is not properly secured [3]. Additionally, the central server may not always be trusted to correctly aggregate updates and avoid intentional or unintentional data leaks [4]. Due to these privacy concerns, some privacy-sensitive organizations, such as banks and hospitals, are reluctant to share their model updates with a third-party central server [5].

Instead, for privacy-sensitive application domains such as finance and healthcare, ensemble learning [6] is commonly used to harness the power of private models from various organizations. Unlike federated learning, where clients train the same model and a central server aggregates the model updates, ensemble learning allows each client to maintain a private model. The central server aggregates the inference requests or predictions of these diverse models to generate the final predictions. By doing this, ensemble learning can effectively handle changes and uncertainties in data distribution while maintaining high performance. Additionally, ensemble learning does not require each client to share their model weights, further improving privacy protection. Therefore, ensemble learning is widely applied in these fields to enhance model reliability and privacy protection. Although distributing inference requests to multiple clients might raise privacy concerns, techniques like data washing, anonymization, encryption, and confidential computing (e.g., Trusted Execution Environment, TEE) [5] can safeguard input data privacy. For instance, removing personal information from medical records and bills ensures user privacy is protected and not disclosed.

Ensemble learning is naturally suitable for the edge environment for two primary reasons. First, the significant variations in memory capacity among edge devices make it challenging to train models of the same size across these heterogeneous devices. Although advanced FL algorithms can effectively aggregate updates from models of diverse sizes, this often impacts model accuracy and convergence [7]. In contrast, en-

This work was supported in part by the National Science Foundation of China under Grants 62172454 and 62432004, the Guangdong Basic and Applied Basic Research Foundation under Grant 2023B1515020120, the Major Key Project of PCL under Grants PCL2024A06 and PCL2022A05, and the Shenzhen Science and Technology Program under Grant RCJC20231211085918010. The corresponding author is Xu Chen (chenxu35@mail.sysu.edu.cn).

semble learning inherently accommodates heterogeneous models. Second, due to the resource constraints of edge devices, deployed models are typically compressed, leading to accuracy loss. Additionally, the dynamic nature of edge environments means that the real-time distribution of inference data can deviate from the training data (i.e., data drift), exacerbating the accuracy loss of compressed models [8]. Ensemble learning mitigates this issue by combining predictions from multiple individual edge models, thereby improving inference accuracy and counteracting the effects of data drift.

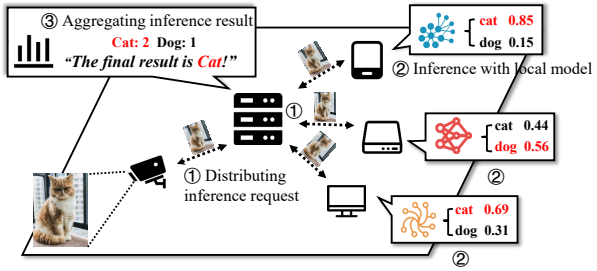


Fig. 1: An illustration of federated inference at the edge

In this paper, we address the challenge of using ensemble learning to integrate predictions from multiple edge models to handle data drift in cooperative edge AI, as shown in Fig. 1. This scenario is also known as "Federated Inference" [9] or "Federated Prediction Serving" [5]. Similar to federated learning, dynamic participant selection [10], [11] is a critical issue in federated inference. This is particularly important for latency-sensitive applications like fraud detection in finance, where strict service-level objectives (SLOs) on latency are common. Additionally, due to resource constraints at the edge, the cost of federated inference must also be controlled. Given the dynamically arriving inference requests, it is crucial to balance accuracy, latency, and cost through dynamic participant selection. However, maximizing ensemble accuracy under cost and SLO constraints presents two practical challenges. First, in an online setting with dynamic data drift and varying communication/computation latencies, the inference accuracy and latency of each participant remain unknown until inference is complete. Second, even in an offline setting where these parameters are predefined, the problem is NP-hard, as it can be reduced to the classical Knapsack problem.

To address these challenges, we propose an online model ensemble framework that combines the strengths of online bandit learning with approximate optimization techniques. Our approach starts with online bandit learning, incorporating performance metrics of model ensembles to dynamically select participants on-the-fly. With the learned accuracy and latency parameters, we apply a lightweight and lazy knapsack algorithm to tackle the underlying combinatorial optimization problem and address its NP-hardness. Specifically, we build upon the combinatorial variant of multi-armed bandits (MAB) [12] to introduce `SOEE`, which adapts to request distributions and model performances over time, optimizing decisions as the system operates. Additionally, we exploit the monotonicity and submodularity of the one-slot combinatorial optimization problem, devising a lightweight algorithm, `LGEE`, which offers both near-optimal and computationally efficient solutions. By integrating `SOEE` and `LGEE`, our online model ensemble framework maximizes inference accuracy with sublinear regret. Furthermore, we have implemented a prototype of our framework and, through extensive test-bed evaluations, demonstrate that it improves average inference accuracy by $12\% \sim 80\%$.

II. System Model and Problem Formulation for Federated Inference

A. System Overview

As shown in Fig. 1, we consider a federated inference paradigm in which a set of edge workers, denoted by $\mathcal{M} = \{1, 2, \cdots, M\}$, each possess a private DNN model locally trained for specific tasks such as disease diagnosis, credit risk analysis, and fraud detection. The edge workers are heterogeneous, with local models trained on different datasets, resulting in diverse inference accuracy, latency, and resource demands. To coordinate these heterogeneous edge workers, an edge server is designated as the master node to distribute inference requests to the edge nodes and collect the independent inference results to perform an ensemble, ultimately outputting a high-accuracy inference result.

Without loss of generality, we assume that inference requests arrive periodically. To characterize this dynamic, we consider the system operates in a time-slotted fashion within a large time span of $\mathcal{T} = \{1, 2, \cdots, T\}$. At each time slot $t \in \mathcal{T}$, the edge server first receives an inference request, which can originate from one of the edge workers or an external user device. The edge server then selects a subset of edge workers to participate in the federated inference and broadcasts the inference request to them. Each chosen participant performs local inference using its private model and uploads the inference result to the edge server. After collecting these independent inference results, the edge server aggregates them using ensemble algorithms such as majority voting [13] (e.g., the class label that receives the most votes from the edge workers is selected as the final prediction) and weighted voting [14], and outputs a high-accuracy inference result. For the privacy of the input data of the inference request, as we have discussed in Sec. I, it can be well protected by means such as data washing, anonymization, encryption, and confidential computing (e.g., Trusted Execution Environment, TEE) [5].

B. Entropy-Based Accuracy Model

For each edge worker $m \in \mathcal{M}$, due to the data drift [8], [15], the inference accuracy may vary over time. Thus, we denote the inference accuracy of its private model as $a_{m,t}$ at time slot $t \in \mathcal{T}$, we also use a decision variable $x_{m,t}$ to denote whether this edge work is selected ($x_{m,t} = 1$) to participate into the federated inference or not ($x_{m,t} = 0$) at time slot $t \in \mathcal{T}$. When conducting federated inference, one of our primary objective is to maximize the accuracy of the final prediction after ensemble. However, given the

individual accuracy of each selected model, the final prediction accuracy after ensemble cannot be readily model as a closed-form function of $a_{m,t}$ and $x_{m,t}$. To address this issue, in this work we adopt a combination of individual model accuracy and model diversity to depict the ensemble performance. The rationale of this rule is intuitive and widely-recognized [16] [17] [18]: the selected models should be as accurate as possible to avoid degrading the ensemble accuracy, while also being as diverse as possible to adapt to various data distributions and reduce covariance[1].

To evaluate the individual model accuracy, we consider the summation item $\sum_{m\in\mathcal{M}} a_{m,t}x_{m,t}$. Intuitively, this term represents the "accurate" part of the ensemble at time slot t. If the ensemble contains more accurate enough individual model, the performance of the ensemble is also expected to be good. Furthermore, to evaluate the ensemble diversity, we adopt the system-wise joint entropy $E(\mathcal{S}_t)$ to capture the joint entropy of the currently selected participant over the past time slot $[1, 2, \cdots, t-1]$, where $\mathcal{S}_t = \{m|m \in \mathcal{M}, x_{m,t} = 1\}$ denotes the set of selected participants of the federated inference at time slot t. The definition of the entropy $E(\mathcal{S}_t)$, originally proposed by Shannon [19], is in consistent with recent literatures on ensemble learning [20] [21] [22] [23]:

$$E(\mathcal{S}_t) = -\sum_{\mathcal{S}_t} p(z_t)\log p(z_t), \tag{1}$$

where $z_t \in \{0,1\}^{|\mathcal{S}_t|}$ is the result vector of the selected participants at time slot t, and $z_{m,t} = 1$ if edge worker m's inference is correct at time slot t, and $z_{m,t} = 0$ if incorrect; $p(z_t)$ is the frequency of z_t over the time slot $[1, 2, \cdots, t-1]$. In our problem, note that $E(\mathcal{S}_t)$ captures the model diversity over the past slots $[1, 2, \cdots, t-1]$, for the participants $\mathcal{S}_t$ selected at time slot t. While the expression of the joint entropy is complicated and not intuitive, latter in Sec. III. D we will show that this expression holds a nice property of *submodularity*, which facilitate the algorithm design.

Request No.	Worker 1	Worker 2	Worker 3
I	0	1	1
II	0	1	0
III	1	1	0
IV	0	1	1
V	0	1	0

TABLE I: An example of joint entropy

For example, assuming there are 3 workers, and workers have finished 5 requests. The results are shown in Table. I. Then at the beginning of time slot 6, the entropy of this ensemble is $E(\mathcal{S}_6) = -(0.2 * \log 0.2 + 0.4 * \log 0.4 + 0.4 * \log 0.4) \approx 0.458$, as result $[0, 1, 1]$ and $[0, 1, 0]$ appear twice of five and result $[1, 1, 0]$ appears once of five. Intuitively, for a given ensemble, a smaller joint entropy indicates that its historical results are less diverse; in other words, the models within the ensemble are more likely to make the same mistakes. Consequently, we consider such an ensemble to lack sufficient variety. Conversely, a larger joint entropy signifies greater diversity within the ensemble. To improve the generalization ability of the ensemble, it is essential not only to select high-performing models but also to ensure they are as diverse as possible.

By combining the above two part, at each time slot $t \in \mathcal{T}$, we measure the accuracy of the selected participant by:

$$F(\mathcal{S}_t) = \sum_{m\in\mathcal{M}} a_{m,t}x_{m,t} + \lambda E(\mathcal{S}_t), \tag{2}$$

where λ is the trade-off parameter between individual accuracy and ensemble diversity.

C. Cost and SLO Model

In practice, when selecting edge workers to participate into the federated inference, the selection is typically constrained by the cost and performance SLO. Specifically, due to the computational intensity of the model inference, edge workers will consume a large amount of computing resources as well as energy. Therefore, we set a cost budget C for each time slot. Furthermore, for each inference request, we denote each worker m's inference cost as c_m. Thus, at each time slot t, we can have the following cost constraint: $\sum_{m=1}^{M} c_m x_{m,t} \leq C, \forall t \in \mathcal{T}$.

On the other hand, consider the final prediction results of the federated inference is typically consumed by some real time business such as credit risk analysis and fraud detection, thus the process is enforced with stringent Service Level Objective (SLO) in terms of total latency. Here we use $d_{m,t}$ each edge worker m's latency which include the transmission latency of the input data of the request, and the inference latency of the local model. Note that in practice, due to the fluctuation of the network bandwidth as well as the resource competition at each edge node, $d_{m,t}$ typically fluctuates over time, and can be only measured after the completion of the inference. By denoting the SLO as D, we have the following SLO constraint: $d_{m,t}x_{m,t} \leq D, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}$.

D. Problem Formulation

In this paper, we aim to develop a cost-efficient online model ensemble framework to facilitate federated inference, towards the goal of maximizing the long-term inference accuracy under cost and SLO constraints. This problem can be formally cast as the following integer linear programming (ILP) $\mathbf{P_1}$.

$$\mathbf{P_1}: \text{maximize} \quad \sum_{t=1}^{T}\left(\sum_{m=1}^{M} a_{m,t}x_{m,t} + \lambda E(\mathcal{S}_t)\right), \tag{3}$$

$$\text{subject to} \quad \sum_{m=1}^{M} c_m x_{m,t} \leq C, \forall t \in \mathcal{T}, \tag{3a}$$

$$d_{m,t}x_{m,t} \leq D, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{3b}$$

$$x_{m,t} \in \{0,1\}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{3c}$$

$$\mathcal{S}_t = \{m|m \in \mathcal{M}, x_{m,t} = 1\}, \forall t \in \mathcal{T}. \tag{4d}$$

[1]Note that this decision rule is orthogonal to the specific fusion rule for inference results, such as majority voting [13] and weighted voting [14].

Challenge Analysis: Solving Problem P_1 is far from trivial due to the following dual challenges. First, the problem involves uncertain/stochastic information $a_{m,t}$ and $d_{m,t}$, which cannot be accurately obtained when making the ensemble decisions at the beginning of each time slot t. For instance, due to the fluctuation of network bandwidth and resource competition at each edge node, $d_{m,t}$ typically varies over time and can only be measured after the completion of the inference. This represents an *online learning* setup, requiring us to learn the parameters $d_{m,t}$ on-the-fly. Second, even with an offline setup where $d_{m,t}$ is known a priori at the beginning of each time slot t, the corresponding optimization problem remains NP-hard, as it can be reduced from the classical knapsack problem.

III. Online Optimization Framework Design and Analysis

In response to the above dual challenges, in this section we design an online optimization framework by fusing the power of MAB for online learning and sub-modular optimization for approximate optimization.

A. Framework Overview

Fig. 2 illustrates the overview of the proposed online optimization framework, it consists of three components: Transformation module, Learning module and Optimization module. The Transformation module first reformulates the SLO constraint into the objective function, facilitating the application of MAB. Then, the Learning module — an online ensemble creation algorithm based on Combinatorial MAB — enhances model ensemble performance by dynamically learning both the performance of the edge workers and the data distribution of the arriving requests, without any prior knowledge of uncertain data. Finally, the Optimization module, with the learned parameters of the Learning module, it solves the NP-hard problem via the computationally-efficient sub-modular technique.

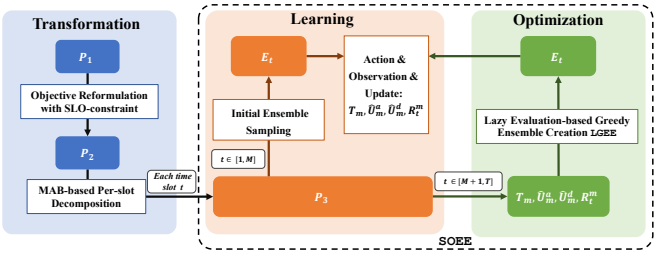


Fig. 2: The algorithm framework overview

B. Problem Transformation

While MAB have been proven to be powerful in solving sequential decision-making problem, it is not directly applicable to our problem P_1. This is because that if we simply view a subset of edge workers as an arm, the decision space will be overwhelmingly large with an size of 2^M, making it difficult to balance the exploration and exploitation of each arm, and causing MAB fail to converge. To deal with this, we adopt a variant of MAB, called Combinatorial MAB (CMAB) [24] to transform combinatorial Problem P_1.

Another challenge when applying MAB is incurred by Constraint (3b). As we can see with stochastic latency parameter $d_{m,t}$, Constraint (3b) might be violated dynamically. Since $d_{m,t}$ only appears in constraints, it is difficult for MAB to learn its distribution so as to decrease the probability of SLO-violation. To address this problem, we integrate the accuracy and latency of each edge workers into an unified term $u_{m,t} = Pr\{d_{m,t} \leq D\} * a_{m,t}$, i.e., the effective inference accuracy $u_{m,t}$ of the worker m, which is product of the actual inference accuracy $a_{m,t}$ multiplied by the probability of finishing the inference within the SLO D. By introducing $u_{m,t}$, we can learn $a_{m,t}$ and $d_{m,t}$ simultaneously. We also establish *hard barriers* to preclude any violation of this constraint during the real-time operation. Specifically, when the latency has reached the SLO constraint, SOEE prevents subsequent slower workers to join the ensemble even though it is selected by the algorithm.

By introducing the effective inference accuracy term $u_{m,t}$, we incorporate the SLO constraint into the objective function, this enable us to transform the original problem P_1 into an online learning Combinatorial MAB as follows.

$$\mathbf{P_2}: \text{maximize} \quad \sum_{t=1}^{T} \left(\sum_{m=1}^{M} u_{m,t} x_{m,t} + \lambda E(\mathcal{S}_t) \right), \tag{4}$$

$$\text{subject to} \quad \sum_{m=1}^{M} c_m x_{m,t} \leq C, \forall t \in \mathcal{T}, \tag{4a}$$

$$x_{m,t} \in \{0,1\}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{4b}$$

$$\mathcal{S}_t = \{m | m \in \mathcal{M}, x_{m,t} = 1\}, \forall t \in \mathcal{T}. \tag{4c}$$

C. Stochastic Online Edge Ensemble via CMAB

Now we are ready to present our combinatorial multi-armed bandit algorithm for online participant selection for federated inference, i.e., SOEE as shown in Algorithm 1.

Throughout the algorithm, we maintain the following parameters for each worker $m \in \mathcal{M}$: (i) A counter vector T_m recording the number of times worker m has been selected; (ii) the empirical distribution $\hat{U}^a_m$ of worker m's accuracy $a_{m,t}$, represented by its Cumulative Distribution Function (CDF) $\hat{P}^a_m$; (iii) the empirical distribution $\hat{U}^d_m$ of worker m's inference latency $d_{m,t}$, represented by its CDF $\hat{P}^d_m$; (iv) the empirical distribution $\hat{U}_m$ of worker m's transformed inference latency $d_{m,t}$, represented by its CDF $\hat{P}_m$; and (v) An inference result matrix $R \in \{0,1\}^{M \times T}$, with element R^m_t representing the inference result of worker m in time slot t (1 for correct and 0 for incorrect), which is used to calculate the joint entropy.

Line 1-8 is the initialization phase (or pure exploration phrase) of SOEE, where the algorithm randomly selects an ensemble that includes each worker m to fully explore their performance. In this phase, SOEE initializes T_m, $\hat{P}^a_m$, $\hat{P}^d_m$ for each worker m based on their inference results. The matrix R is also recorded in this phase.

Line 9-16 is the exploration and exploitation phase of SOEE. Specifically, Line 12 estimates the model accuracy

Algorithm 1: Stochastic Online Edge Ensemble (SOEE)

1 $t = 0$;
2 *// Initialization*
3 **for** $m = 1$ ***to*** M **do**
4 $t = t + 1$;
5 Select an arbitrary ensemble S_t that contains worker m;
6 Receive the inference results, update T_n, $\hat{P}_n^a$ and $\hat{P}_n^d$ of each $n \in S_t$;
7 update R;
8 **end**
9 *// Exploration & Exploitation*
10 **for** $t = M + 1$ ***to*** T **do**
11 $t = t + 1$;
12 For each $m \in \mathcal{M}$, let $\underline{U}_m$ be the distribution of accuracy whose CDF $\underline{P}_m$ is updated by (5);
13 Select the model ensemble $S_t \leftarrow \texttt{LGEE}(\underline{U}, R)$;
14 Receive the rewards, update T_n and $\hat{P}_n^a$ and $\hat{P}_n^d$ of each $n \in S_t$;
15 Update R;
16 **end**

distribution P_m^a of worker m using Eq. (5), which incorporates the empirical distribution $\hat{P}_m^a$ and a padding term $\sqrt{\frac{\beta \ln t}{T_m}}$ to balance exploration and exploitation. Specifically, when T_m is small, worker m is explored less frequently, increasing its selection probability. Once worker m has been sufficiently explored, its selection probability is primarily influenced by its inference accuracy.

$$\underline{P}_m(x) = \begin{cases} \max\left\{\hat{P}_m(x) - \sqrt{\frac{\beta \ln t}{T_m}}, 0\right\}, & 0 \le x < 1; \\ 1, & x = 1. \end{cases} \tag{5}$$

Then the problem is reduced into line 13, where SOEE calls LGEE to solve the underlying optimization problem for each time slot t. The per-slot optimization problem P_3 of time slot t is presented below. The details of LGEE is presented in Section 4.5.

$$\mathbf{P_3}: \text{maximize} \quad \left(\sum_{m=1}^{M} u_{m,t} x_{m,t} + \lambda E(\mathcal{S}_t)\right), \tag{6}$$

$$\text{subject to} \quad \sum_{m=1}^{M} c_m x_{m,t} \le C, \tag{6a}$$

$$x_{m,t} \in \{0,1\}, \forall m \in \mathcal{M}, \tag{6b}$$

$$\mathcal{S}_t = \{m | m \in \mathcal{M}, x_{m,t} = 1\}, \forall t \in \mathcal{T}. \tag{6c}$$

The solution to Problem P_3 will generate the ensemble for time slot t. Then, SOEE will use the selected edge workers to complete the inference task and obtain rewards based on their inference results. Specifically, if a worker infers correctly within the SLO constraint, its reward is 1; if the worker violates the SLO constraint or infers incorrectly, its reward is 0. This approach, where rewards are obtained in such a manner, is known as semi-feedback bandits. In Lines 14 and 15, SOEE updates the maintained parameters.

D. Approximated Algorithm for the Per-Slot Problem

For the per-slot problem P_3, while it is NP-hard (as it can be reduced from the classical knapsack problem), it poses two nice properties which enables us to design efficient approximation algorithm. Specifically, the first property is known as **Monotonicity**: Adding a new worker to an existing ensemble does not worsen the objective function of P_3 (since it neither decreases the summed accuracy nor the joint entropy). The second property is **Submodularity**, also known as *diminishing returns*, where the marginal gain of adding a new worker to an ensemble either decreases or remains constant as the ensemble size increases. In our scenario, the linear summation of accuracy is modular, while the entropy function is typically sub-modular [20].

Given the above two properties, the classical sub-modular method [25] would select the workers in a greedy manner: First calculate the ratio of the marginal reward value and cost of the workers, then add workers into the ensemble based on the ratio from largest to smallest until the cost constraint is violated, and finally return the best decision between the above ensemble and the best single worker. While this method have good theoretical and empirical performance [26], it does not scale well due to its quadratic time complexity.

In this paper, we adopt an accelerated variant of the sub-modular method. Specifically, we introduce LGEE, which leverages the benefits of lazy evaluation [27], as shown in Algorithm 2. The key idea behind lazy evaluation is that as the size of an ensemble increases, the marginal gain from adding the same worker to the ensemble decreases. Therefore, in each round, we do not compute the marginal gain for every candidate worker but instead check if the *old best* remains the *new best*. At each round, LGEE also chooses the worker with the highest marginal gain-cost ratio (i.e., $\text{argmax}_{i \in \mathcal{M}'} \frac{\delta_i}{c_i}$ in Line 10). However, instead of directly updating the marginal gain for every worker, LGEE first marks each worker as *dirty*, indicating that their marginal gain records are from the previous round (lines 5-8). Next, LGEE sorts the workers by their marginal gain-cost ratio in non-decreasing order, selects the worker with the highest ratio, and updates its marginal gain based on the current worker ensemble if the worker is *dirty* (lines 9-19). Once updated, the worker is marked as *clean*. A round terminates when we encounter the first *clean* worker (lines 11-14).

In many cases, the updated marginal gain-cost value for the selected *dirty* worker remains the highest among all workers. When this occurs, LGEE directly selects this worker without recalculating the marginal gains of all workers. In the best case, LGEE has a time complexity of $O(M)$. While in the average case, the time complexity of LGEE is $O(M \log M)$, where M is the number of edge workers.

Algorithm 2: Lazy Greedy Edge Ensemble (LGEE)

Initialize $\mathcal{S} = \emptyset$, $B = C_t$, $\mathcal{M}' = \mathcal{M}$;
foreach $m \in \mathcal{M}$ **do**
  $\delta_m = +\infty$;
end
while $|\mathcal{M}'| > 0$ **do**
  for $i \in \mathcal{M}'$ **do**
    $f_i \leftarrow$ **false**;
  end
  while ***true*** **do**
    $i^* = \text{argmax}_{i \in \mathcal{M}'} \frac{\delta_i}{c_i}$;
    if f_{i^*} **then**
      if $c(\mathcal{S} \cup \{i^*\}) \leq B$ **then**
        $\mathcal{S} = \mathcal{S} \cup \{i^*\}$;
      end
      $\mathcal{M}' = \mathcal{M}' \setminus \{i^*\}$;
      Break;
    end
    else
      $\delta_{i^*} = F(\mathcal{S} \cup \{i^*\}) - F(\mathcal{S})$;
      $f_{i^*} \leftarrow$ **true**;
    end
  end
end
$m^* \leftarrow \text{argmax}_{m \in \mathcal{M}} F(m)$;
return $\text{argmax}\{F(m^*), F(S)\}$;

E. Performance Analysis

In this subsection, we conduct performance analysis of the presented online optimization framework, including the regret of SOEE and the approximation ratio of LGEE.

Theorem 1. LGEE achieves an approximation ratio of α, where α is the root of

$$(1-\alpha)\ln(1-\alpha) + (2-\tfrac{1}{e})(1-2\alpha) = 0,$$

and satisfies $\alpha > 0.405$.

The approximation ratio of 0.405 means that the solution value of LGEE is no less than 0.405 times the optimum. Based on this ratio α, we further drive the regret bound of SOEE.

Theorem 2. *$\alpha-$approximation regret bound:* The total regret achieved by SOEE over the finite time slots of T is bounded:

$$93F^{\max}M\sqrt{T\ln T} + \left(\tfrac{\pi^2}{3}+1\right)\alpha F^{\max}M,$$

where $F^{\max}$ is the upper bound of the objective function F in Eq. (2), M is the number of edge workers.

SOEE achieves a $O(\sqrt{T\log(T)})$ regret bound, which increases sub-linearly and has a derivative approaching zero in the limit. This implies that the difference between the objective value and the optimal value of the long-term optimization problem will converge as the number of time slots increases.

Proof sketch: To demonstrate the approximation ratio of LGEE, we can show that LGEE accelerates the traditional submodular method [26] while producing identical outputs. For the regret bound, we integrate the approximation ratio of LGEE into the regret analysis of the MAB method which exactly solves the per-slot problem [12]. □

IV. Prototype Implementation

We implemented a prototype of our proposed federated inference serving system, as shown in Fig. 3. For the edge server, we used an Intel NUC11 PC to provide inference services. For edge inference workers, we employed two settings: (1) nine Firefly-RK3399 development platforms as homogeneous edge workers, and (2) three Raspberry Pi 3B Plus, three Nvidia Jetson AGX Orin, and three Firefly-RK3399 development platforms as heterogeneous edge workers. The hardware specifications are detailed in Table II. Docker technology [28] was used for fine-grained control of hardware attributes such as CPU and memory to scale system performance.

The edge server and workers were connected via 5GHz Wi-Fi. TCP socket connections between the edge server and workers were implemented using the ZeroMQ protocol [29], a widely adopted message queue for prototype testing. The listening sockets of the edge server were implemented using multi-threading techniques to monitor different workers' results in real-time. The overall system was implemented using Python3, with more than 2200 LoCs.

The implemented prototype consists of four modules: the SOEE Decision-making Module, the Communication Module, the Request Dispatching Module, and the Post-processing Module. The system workflow is as follows: First, inference requests are sent to the edge server. In the experimental scenario, inference requests are generated by the system from datasets based on some certain predetermined distributions. In real-world scenarios, inference requests are generated by user devices and transmitted to the edge server through the network. The edge server then receives these requests, calls the proposed SOEE algorithm to make the edge ensemble decision. Then, inference requests, also known as inference tasks, are distributed to the selected workers via Wi-Fi network through Zero-MQ. Edge workers keep active by running listening threads for message queues. Upon receiving the task, each edge worker falls into callback state and calls its own inference module to perform individual inference. Finally, once the inference is complete, the selected workers send messages to inform the edge server of the inference results and its corresponding inference latency. The edge server collects workers' inference results and latency, calls the Post-processing Module to update necessary parameters, and waits for the next round of inference requests to arrive.

V. Performance Evaluation

A. Experiment Setup

DNN Models: We adopt three model families in our experiments: LeNet [30], VGG [31], and ResNet [32]. For *homogeneous* edge workers, i.e., the nine Firefly-RK3399 development platforms, we divide them into three groups with different inference quality levels, and train LeNet and VGG each in three different sizes, as shown in Table III. For the

Device	CPU	Memory	Operating System
Intel NUC 11	8-core Intel Core i5-1135G7 CPU, 2.40GHz × 4, 2.42GHz × 4	32GB	Windows 11
Firefly-RK3399	6-core aarch64 CPU, 1.80GHz Cortex A72 × 2, 1.42GHz Cortex A53 × 4	4GB	Ubuntu 18.04
Raspberry Pi 3B Plus	4-core ARMv7l CPU, 1.40GHz Cortex A73 × 4	1GB	Ubuntu 22.04
Nvidia Jetson AGX Orin	12-core ARMv8l CPU, 2.20GHz Cortex A78 × 12	32GB	Ubuntu 20.04

TABLE II: Specification of the hardware test-bed

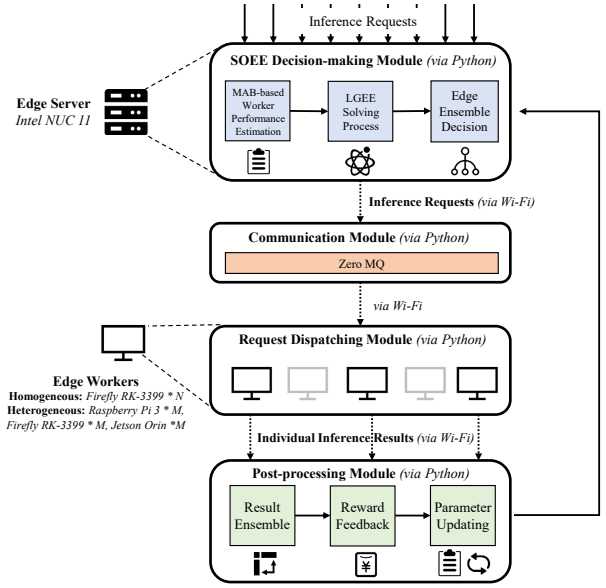


Fig. 3: System architecture of the prototype

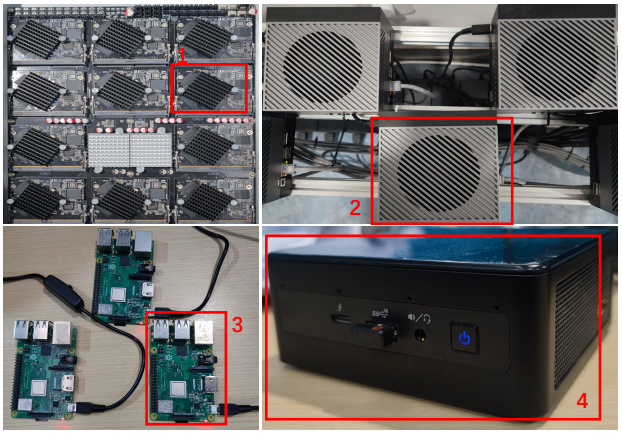


Fig. 4: Configurations of the test-bed: (1) Firefly-RK3399s; (2) Jetson Orins; (3) Raspberry Pis; (4) Intel NUC.

heterogeneous edge workers, we deploy an identical network architecture, ResNet50, as the inference serving network for different devices.

Datasets: We adopt three datasets for experiments. Explicitly, MNIST [33] and CIFAR-10 [34] are adopted to train and evaluate LeNet, while CIFAR-100 [34] is adopted to train and evaluate VGG and ResNet. In order to simulate different inference quality levels, the number and type of training classes of the models vary from group to group and from worker to worker in the same group.

Model Group	LeNet	VGG	ResNet
SMALL	2×CONV+3×FC (70K)	VGG13	ResNet50
MEDIUM	3×CONV+3×FC (336K)	VGG16	ResNet50
LARGE	3×CONV+3×FC (429K)	VGG19	ResNet50

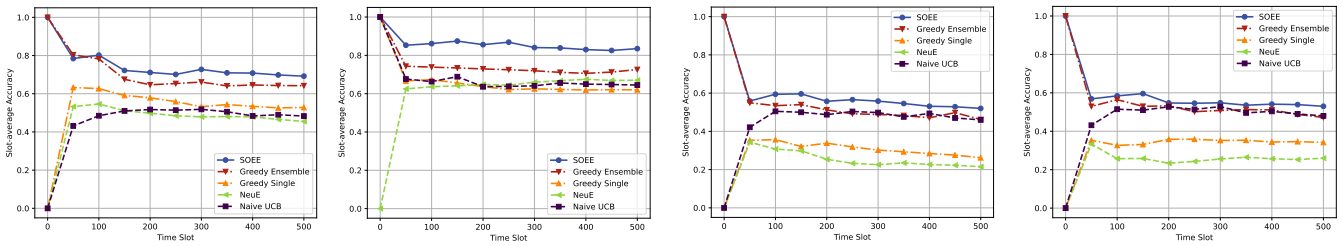
TABLE III: Model sizes of different groups

Other Parameters: We consider $T = 500$ time slots. To simulate the data drift, we change the data distribution of the inference request (i.e., input image) every 50 time slots. The cost coefficient c_m of each edge worker m can be regarded as relative with the size of the serving model. The larger the model is, the more system resources (e.g., memory, CPU) the inference request consumes, thus we set the cost of each model according to its size. Besides, we set $C = 100$ units and $D = 0.25$s in our experiment. For homogeneous workers, we conducted experiments with the setup of LeNet + CIFAR10, LeNet + MNIST, and VGG + CIFAR100 respectively; for heterogeneous workers, we conducted experiments with the setup of ResNet + CIFAR100.

Baselines: We consider four baselines for the benchmark experiments. **NeuE [35]:** NeuE adopts NeuralUCB [36], a contextual MAB algorithm which treats each possible model combination as an arm and utilizes a utility predicting neural network to approximate the reward function. **Naive UCB:** a classic MAB algorithm which considers each possible model combination as an arm, and balances the exploration and exploitation according to each arm's upper confidence bound (UCB) [12]. **Greedy Ensemble:** which selects the best ensemble via the greedy algorithm, i.e., `LGEE`. Its parameters are not estimated by Equation (5), but only depend on the historical empirical values. **Greedy Single**: This algorithm selects the best single DNN model based on the historical empirical values of the inference accuracy, rather than assembling the results of multiple DNNs.

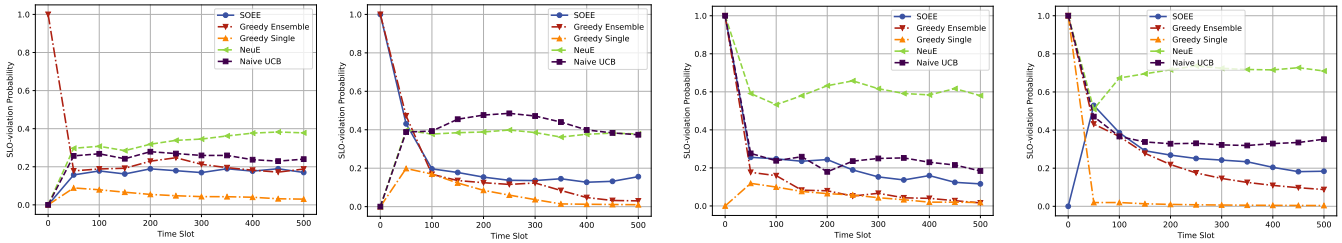
B. Experiment Results

Improvement of Inference Accuracy. Fig. 5 compares the accuracy performance of our proposed scheme and the other four baselines under various setups. We observe that `SOEE` achieves the highest average accuracy under all setups. Specifically, `SOEE` outperforms NeuE, Naive UCB, Greedy Ensemble, and Greedy Single by 80%, 24%, 12%, and 54% on average, respectively. Note that Naive UCB regards each ensemble as an arm, which leads to an exponential decision space and makes the initialization and exploration of the decision space extremely difficult. Therefore, it takes much



(a) CIFAR-10 with LeNet (b) MNIST with LeNet (c) CIFAR-100 with VGG (d) CIFAR-100 with ResNet

Fig. 5: The slot-average accuracy under various schemes and models/datasets



(a) CIFAR-10 with LeNet (b) MNIST with LeNet (c) CIFAR-100 with VGG (d) CIFAR-100 with ResNet

Fig. 6: The slot-average SLO violation probability under various schemes and models/datasets

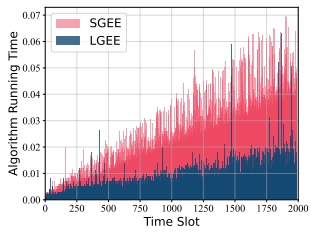


Fig. 7: Comparison of the algorithm execution time (s)

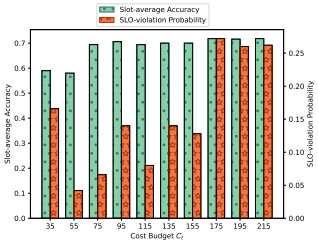


Fig. 8: The effect of the cost budget C

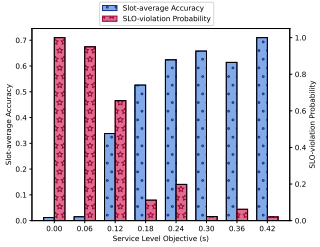


Fig. 9: The effect of the SLO D

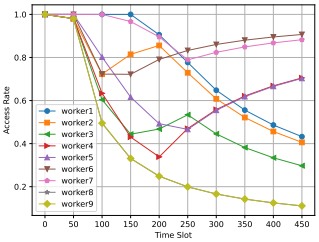


Fig. 10: The dynamics of access rate of each workers

longer for Naive UCB to converge than SOEE. As for NeuE, it cannot outperform Naive UCB in most cases. As a contextual ucb-based algorithm, it cares about a decision space of the same size as Naive UCB. Its optimization objective is not a deterministic function, but is fitted by a neural network (i.e., NeuralUCB from [36]). This design makes it being sensitive to the amount of training data, and the environment dynamics (which is just the case of our experiment). Besides, Greedy Ensemble is also 38% better than Greedy Single on average. This indicates the benefit of model ensemble, regardless of the online learning characteristic.

Reducing SLO Violations. Fig. 6 depicts the probability of SLO violation of various schemes. For the 500 time slots, the probability of SLO violation with SOEE, NeuE, and Naive UCB are 16%, 44%, and 29%, respectively. Naive UCB only pays coarse-grained attention to the ensemble, leading to the inadequate representation of the SLO-satisfaction space. Therefore, Naive UCB achieves a poorer performance than that of SOEE. As for NeuE, due to its inherent algorithm design, it is unable to learn and update the latency performance of different workers or ensembles in real time. On the other hand, Greedy Ensemble and Greedy Single, have a low-level SLO violation probability, which converges to 6% and 1%, respectively. This is rationale, since online learning based schemes makes some sacrifices in the probability of SLO violation, but obtains a higher accuracy performance. The inherent reason is that SOEE pursues a trade-off between exploitation and exploration, instead of merely selecting the local best solutions.

Computational Efficiency. We demonstrate the computational efficiency of the accelerated sub-modular method LGEE by comparing it with the traditional sub-modular method SGEE [25] in Fig. 7. The results show that, aside from incidental CPU jitter errors, the running latency of SOEE with LGEE is consistently less than 50% of that with SGEE, validating our choice of LGEE. Notably, as the number of time slots increases, the running latency also rises due to the increased computation latency for calculating joint entropy in Equation (2). To prevent an infinite increase in computation latency, we can integrate a *sliding window mechanism* into SOEE to periodically discard outdated result records, thereby reducing computation latency.

Sensitivity of Cost Budget C. Fig. 8 shows the effect of the cost budget C. As the cost budget C increases, both

the slot-average inference accuracy and the SLO violation probability rise. This is straightforward because a higher cost budget relaxes constraint (4a), motivating the edge server to select high-accuracy but cost-expensive workers (i.e., workers in the large model group). Since the SLO constraint (3b) is transformed into a soft barrier in Problem P_2, it cannot prevent SOEE from selecting these workers as effectively as the hard constraint (4a). Therefore, both accuracy and the SLO violation probability increase.

Sensitivity of SLO D. Fig. 9 shows the effect of the SLO D. As the SLO D increases, the slot-average inference accuracy increases, while the SLO violation probability decreases. With an increasing SLO, workers from different groups can gradually meet the SLO soft barrier and provide meaningful and acceptable inference results. When the SLO is equal to or higher than 0.30s, all workers can meet the SLO, leading to the highest accuracy. Regarding the SLO violation probability, it is 100% when SLO is 0.00s, since no worker can meet the SLO barrier. As the SLO increases, the violation probability dynamically decreases because the SLO becomes achievable for more workers across different groups.

Adapting to Data Drift. Finally, we demonstrate how SOEE dynamically selects edge workers to adapt to data drift. Since the data distribution of requests changes every 50 time slots (referred to as one interval), we present the access rate, i.e., the probability of being selected for each worker, in Fig. 10. Worker 1 is selected until the fourth interval because the second interval's distribution matches Worker 1's training classes, optimizing its performance and influencing selection longer. Worker 2's access rate rises at the third interval as SOEE detects a distribution match with Worker 2's training classes. This pattern repeats for Workers 3, 4, and 5 at their respective intervals. Worker 6's training classes include those of Workers 2 and 3, so its turning points are at the third and fourth intervals. Worker 7, with training classes including those of Workers 1 and 5, follows Worker 1's pattern and turns at the sixth interval. Workers 8 and 9, with high SLO violation probabilities, are not selected after the first interval's initialization.

VI. Related Work

A. Edge Inference Serving

When deploying DNN model inference at the edge, a fundamental challenge is how to satisfy low-latency and high-accuracy requirements under limited resource availability. Various efforts have been proposed to address this issue. Among these, model compression has been widely adopted. For example, Liu *et al.* [37] developed a usage-driven selection framework, AdaDeep, which automatically selects a combination of compression techniques (e.g., weight pruning and quantization) for a given DNN, aiming to maximize user-specified performance goals within resource constraints. To harness the power of cooperative edge devices, Hu *et al.* [38] proposed a distributed inference mechanism, EdgeFlow, which partitions model layers into independent execution units that can be distributed to nearby edge devices. Notably, compared to model compression, model partitioning is lossless and does not degrade inference accuracy. To address data drift, which can deteriorate runtime inference accuracy of compressed models, Cai *et al.* [8] presented an online optimization framework that adaptively picks the best configurations for both model inference and retraining co-located within an edge node. For the emerging Transformer-based Large Language Models (LLMs), Ye *et al.* [39] designed a hybrid parallelism mechanism that integrates tensor parallelism and sequence parallelism to accelerate distributed LLM inference at the edge.

B. Federated Inference with Ensemble Learning

The concept of federated inference was first introduced by Malka *et al.* [9], who proposed an edge ensemble inference protocol. Their approach allows each user to perform inference independently while benefiting from collaboration with neighboring devices to form an ensemble of DNNs. Weng *et al.* [5] proposed an incentive mechanism based on Bayesian game theory to enhance overall accuracy in federated inference services. Wang *et al.* [40] theoretically analyzed the unnecessary communication in federated inference and reshaped the inference communication process to minimize the communication costs.

The most relevant work to ours is NeuE [35], which uses neural network-based contextual multi-armed bandits to predict the performance of model ensemble. However, neural network-based techniques require pre-training, which partially conflicts with online requirements and poses challenges for maintaining a training set during real system operation. Additionally, the time complexity and empirical performance of neural network computation can be unpredictable. In contrast, our approach focuses on the joint and online optimization of inference accuracy, performance, and cost — a three-way tradeoff not explored in previous literature. Our method is "model-free", i.e., it makes no prior assumptions about the model and does not require a global model for inference services.

VII. Conclusion

In this paper, we present an online model ensemble framework for model inference serving at the edge. The objective of the presented framework is to maximize the ensemble accuracy under the resource budget and latency constraint. Achieve this goal is nontrivial since the underlying problem involves uncertain information and being NP-hard. In response, the presented framework first leverages online bandit learning to adaptively selects the best ensemble for each inference request. Then, to address the NP-hardness, it develops a lazy greedy algorithm based on submodularity to achieve near-optimal performance. We provide theoretically rigorous performance analysis of the framework. We have also implemented a prototype of our framework and, through extensive test-bed evaluations, demonstrate that it improves average inference accuracy by $12\% \sim 80\%$.

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[2] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy," in *Proc. of ACM Mobicom*, 2020.

[3] H. Yang, M. Ge, D. Xue, K. Xiang, H. Li, and R. Lu, "Gradient leakage attacks in federated learning: Research frontiers, taxonomy and future directions," *IEEE Network*, 2023.

[4] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame *et al.*, "Safelearn: Secure aggregation for private federated learning," in *Proc. of IEEE Security and Privacy Workshops (SPW)*, 2021.

[5] J. Weng, J. Weng, H. Huang, C. Cai, and C. Wang, "Fedserving: A federated prediction serving framework based on incentive mechanism," in *IEEE INFOCOM*, 2021.

[6] Z.-H. Zhou and Z.-H. Zhou, *Ensemble learning*. Springer, 2021.

[7] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. of IEEE INFOCOM*, 2021.

[8] H. Cai, Z. Zhou, and Q. Huang, "Online resource allocation for edge intelligence with colocated model retraining and inference," in *Proc. of IEEE INFOCOM*, 2024.

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

[10] Y. Zhao and X. Gong, "Quality-aware distributed computation and user selection for cost-effective federated learning," in *Proc. of IEEE INFOCOM WKSHPS*, 2021.

[11] Z. Jiang, Y. Xu, H. Xu, Z. Wang, and C. Qian, "Heterogeneity-aware federated learning with adaptive client selection and gradient compression," in *Proc. of IEEE INFOCOM*, 2023.

[12] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[13] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information fusion*, vol. 6, no. 1, pp. 63–81, 2005.

[14] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowledge and information systems*, vol. 38, pp. 259–275, 2014.

[15] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *Proc. of USENIX NSDI*, 2022.

[16] A. Chandra, H. Chen, and X. Yao, "Trade-off between diversity and accuracy in ensemble generation," *Multi-objective machine learning*, pp. 429–464, 2006.

[17] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I 23*. Springer, 2012, pp. 330–345.

[18] Y. Wu, L. Liu, Z. Xie, K.-H. Chow, and W. Wei, "Boosting ensemble accuracy by revisiting ensemble diversity metrics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 469–16 477.

[19] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[20] C. Sha, K. Wang, X. Wang, and A. Zhou, "Ensemble pruning: A submodular function maximization perspective," in *International Conference on Database Systems for Advanced Applications*. Springer, 2014, pp. 1–15.

[21] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, pp. 181–207, 2003.

[22] H. Dutta, "Measuring diversity in regression ensembles." in *IICAI*, vol. 9. Citeseer, 2009, p. 17p.

[23] H. R. Kadkhodaei, A. M. E. Moghadam, and M. Dehghan, "Hboost: A heterogeneous ensemble classifier based on the boosting method and entropy measurement," *Expert Systems with Applications*, vol. 157, p. 113482, 2020.

[24] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *International conference on machine learning*. PMLR, 2013, pp. 151–159.

[25] L. A. Wolsey, "Maximising real-valued submodular functions: Primal and dual heuristics for location problems," *Mathematics of Operations Research*, vol. 7, no. 3, pp. 410–425, 1982.

[26] J. Tang, X. Tang, A. Lim, K. Han, C. Li, and J. Yuan, "Revisiting modified greedy algorithm for monotone submodular maximization with a knapsack constraint," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 1, pp. 1–22, 2021.

[27] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*. Springer, 2005, pp. 234–243.

[28] "Docker: Accelerated container application development." [Online]. Available: https://www.docker.com

[29] "Zeromq: An open-source universal messaging library." [Online]. Available: https://zeromq.org

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016.

[33] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[34] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[35] Y. Bai, L. Chen, and J. Xu, "Neue: Automated neural network ensembles for edge intelligence," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[36] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with ucb-based exploration," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 492–11 502.

[37] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *Proc. of Mobisys*, 2018.

[38] C. Hu and B. Li, "Distributed inference with deep learning models across heterogeneous edge devices," in *Proc. of IEEE INFOCOM*, 2022.

[39] S. Ye, J. Du, L. Zeng, W. Ou, X. Chu, Y. Lu, and X. Chen, "Galaxy: A resource-efficient collaborative edge ai system for in-situ transformer inference," in *Proc. of IEEE INFOCOM*, 2024.

[40] J. Wang, L. Zhang, Y. Cheng, S. Li, H. Zhang, D. Huang, and X. Lan, "Tvfl: Tunable vertical federated learning towards communication-efficient model serving," in *Proc. of IEEE INFOCOM*, 2023.