Automated Ensemble for Deep Learning Inference on Edge Computing Platforms

Yang Bai, Lixing Chen, Mohamed Abdel-Mottaleb, *Fellow, IEEE,* Jie Xu, *Senior Member, IEEE,*

***Abstract*—Advances in deep learning (DL) have triggered an explosion of mobile intelligence, posing a soaring demand for computing resources that cannot be satisfied by mobile devices. In this paper, we employ *Edge Computing* to deliver better DL inference services to end-users. The key is to leverage deep neural network (DNN) ensemble techniques that provide state-of-the-art performance for many machine learning applications in terms of inference accuracy and robustness. Compared to end-devices, the edge computing platform is endowed with more powerful computing resources, making it feasible to implement DNN ensembles for DL inferences. However, due to the constrained computing capacity of edge servers and the possible service response deadline, an edge server can only use a limited number of DNNs to construct DNN ensembles. This poses a unique problem, namely DNN ensemble selection, for identifying the best-fit DNN ensembles. We propose a novel algorithm called Automated DNN Ensemble (AES) algorithm to solve this problem. Because DNNs exhibit performance variations over different distributions of input data, AES adaptively determines a DNN ensemble according to the features of admitted inference tasks. AES is an online learning algorithm that learns DNNs' in-use performance over time. An ensemble selection rule is further designed as a subroutine of AES to recruit members into the DNN ensemble based on the accuracy and diversity of DNNs. In particular, we theoretically prove that AES can achieve asymptotic optimality. We carry out experiments on real-world datasets. The results show that using the DNN ensemble technique on edge computing platforms dramatically improves the DL inference quality, and AES outperforms other benchmark schemes.**

***Index Terms*—Edge computing, multi-armed bandit, deep neural network ensemble**

I. Introduction

There is a growing trend to bring deep learning (DL) intelligence to mobile devices. Smartphones and hand-held devices are tied to DL in many of their functionalities. This trend is continuously driving the advance of DL techniques for mobile devices. New-generation mobile hardware, e.g., Apple neural engine [1], is designed to accelerate neural network processing. Lightweight DL libraries, e.g., Tensorflow Lite [2] and Core ML [3] are released for on-device DL inference. Novel algorithms, e.g., Deep Neural Network (DNN) compression [4], [5], help compress large-scale DNNs into compact models that fit the size of on-chip RAM. While these techniques enable mobile devices to run the DL inference, they are unlikely to be a universal solution for delivering DL services due heterogeneous computing capacities of mobile devices. A recent study by Facebook [6] shows that over 50% mobile devices are using processors at least six years old, limiting what is possible of DL services. Besides, frequently running DL services also drains the battery fast [7]. Therefore, external boosters become necessary to realize the full potential of DL services for mobile devices. The recently emerged *Edge Computing* [8] is envisioned to be a promising alternative for delivering high-quality DL service [9], [10] to end-users. Being physically close to users and leveraging fast network technologies such as 5G, edge computing promises several benefits compared to traditional *Cloud Computing*, including lower latency, higher energy efficiency, and reduced bandwidth consumption [11]. With the assistance of computation offloading techniques [12], the edge computing platform becomes an optimal site for providing DL inference service to mobile devices — a DL service provider can deploy its DNNs on edge servers and users can send their tasks (i.e., input data to DNNs) to edge servers to complete DL inferences.

This paper aims to take full advantage of edge computing platforms and enhance the service quality of DL inference by employing the *DNN ensemble technique*. The DNN ensemble technique is well recognized in DL communities for its ability of reducing prediction variances and improving inference accuracy [13], [14]. It has been providing state-of-the-art performances for many learning problems. For example, the winning teams of ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) in the latest four consecutive years all incorporate the DNN ensemble technique in their method [15]. Recent years also see a revitalization of the DNN ensemble technique due to its robustness to adversarial attacks [16], [17]. All these advantages make the DNN ensemble technique a natural choice for improving DL inference quality. Using DNN ensembles is originally unfavorable to resource-constrained mobile devices because running multiple DNNs can be costly, e.g., large computation complexity, rapid battery drain, high resource occupation rate. The deployment of edge computing platforms makes the DNN ensemble technique a feasible paradigm for mobile devices. Powerful computing resources at edge servers allow the DL service provider to trade computing resources for better DL inference quality using DNN ensemble techniques. To be specific, a DL service provider will form a committee of DNNs (hereinafter referred to as **EdgeCmte**) at an edge server. Each DNN in EdgeCmte will be used to process users' inference tasks, and then the outputs of individual DNNs are combined to generate final inference results which will be returned to users. This general framework seems straightforward, however, there are several challenges to

This work is supported in part by NSF under grants 2006630, 2033681, 2029858 and 2044991.

Y. Bai, M. Abdel-Mottaleb, J. Xu are with the Department of Electrical and Computer Engineering, University of Miami, FL, 33146. E-mail: y.bai9@umiami.edu, {mottaleb, jiexu}@miami.edu.

L. Chen is with the Institute of Cyber Science and Technology, Shanghai Jiao Tong University, China, 200240. E-mail: lxchen@sjtu.edu.cn

be addressed before the DNN ensemble technique can deliver what it is capable of.

1) The first challenge is the limited computing resources at edge servers. Although the computing capacity of edge servers is much larger than mobile devices, it is still limited [18] compared to a cloud-scale datacenter. Besides, DL service providers may operate under budget constraints that only allow them to rent a limited amount of computing resources for deploying their services. Therefore, an edge server may not be able to run all available DNNs simultaneously for ensemble prediction. Running DNNs sequentially tends to incur a large inference delay that is unfriendly to latency-critical services. This requires the DL service provider to judiciously decide *how many* and *which* DNNs to include in EdgeCmte, namely a *DNN ensemble selection* problem. While the DNN ensemble technique is often used in the DL community, the DNN ensemble selection is still an under-investigated problem in the current literature.

2) The second challenge is the heterogeneity of DNNs. Available DNNs collected by the service provider may come from different sources. This is a natural assumption because the data collected by a single institution is often limited. While data sharing is desirable, it can lead to severe privacy issues, especially for clinical, financial, and social data. By contrast, the trained DNN is a generalization of structured knowledge that contains less privacy-sensitive information. Therefore, sharing DNNs is more welcomed than directly sharing the source data. For example, Facebook has disclosed the open-source release of its Deep Learning Recommendation Model [19] but veils the source data due to privacy concerns. As a result, DNNs collected by a DL service provider can be trained/validated on *different* data sources with *different* DNN architectures by *different* institutions, and hence achieving *different* performances. These four "*different*"s characterize the heterogeneity of DNNs and make it necessary to perform the DNN ensemble selection.

3) The third challenge is the unknown in-use performance of DNNs. While the DNN performance can be evaluated on standard test data, one cannot guarantee that the input data from users in practice has the same distribution as the standard test data. Therefore, the actual performance delivered by DNNs is revealed only during implementation and needs to be learned over time.

4) The fourth challenge is the variability of user inference tasks. The features of inference tasks vary due to many external factors. For example, consider images as DNN input, the device camera determines the resolution and noise of captured images; the time and location affect the image brightness. Different DNNs have different sensitivities to these feature variations, and hence their inference performance also varies across inference tasks. In other words, there is no "master key" for all inference tasks, and the discrimination of the "right key" for certain inference tasks is crucial. This requires the DL service provider to adaptively change its EdgeCmte according to the features of admitted inference tasks.

We perform DNN ensemble selection to deal with above challenges. A novel online learning method is proposed to automate the construction of EdgeCmtes in a way that optimizes the inference accuracy for user tasks. The key contributions of this paper are summarized as follows:

- We design a framework for the edge computing platform to implement DNN ensemble techniques and deliver DL inference services. A unique problem called DNN ensemble selection is formulated to optimize the DL inference quality. The goal of DNN ensemble selection is to adaptively form an EdgeCmte that is best-fit for the currently received inference tasks, meanwhile satisfies constraints posed by the limited computing resources and service response deadline.
- We propose an online algorithm, Automated Ensemble Selection (AES), to provide a solution to the DNN ensemble selection problem. AES leverages the framework of multi-armed bandit. It learns the impact of task features on DNNs' performance and judiciously balances exploration (i.e., learning DNN performances) and exploitation (i.e., maximizing inference quality) to achieve asymptotic optimality. A novel ensemble selection rule is designed as a subroutine of AES to identify best-fit EdgeCmtes based on the learned knowledge. It jointly considers the accuracy and diversity of DNNs in EdgeCmte, and dramatically improves the accuracy and robustness of DL inference services.
- We run experiments on two real-world datasets, Caltech101 [20] and MASATI-v2 [21], to evaluate the proposed algorithm. The experimental results show that AES improves inference accuracy by 24.0% compared to the best single DNN. AES also provides higher learning efficiency compared to benchmark learning schemes.

The rest of this paper is organized as follows. Section II reviews related works. Section III introduces the system model and defines the DNN ensemble selection problem. Section IV designs the AES algorithm. Section V shows experimental results, followed by conclusions in Section VI.

II. Related Work

A. Deep Learning on Edge System

Recent efforts on bringing DL techniques to edge computing platforms have introduced a new research area [11], [22]. The existing works in this area can be categorized into three types. The first type of works is to enable DL training over a network of edge servers, e.g., authors in [23] use distributed DL to build a video surveillance system on edge systems. The second type of works exploits DL for designing better control policies for edge computing platforms, e.g. the work [24] designs a DL-based offloading policy. The third type of work provides DL inference service on the edge platform, and our work belongs to this category. Several recent works have studied to improve the efficiency of DL inference using edge computing. For example, authors in [25], [26] use the DNN partitioning technique to design a collaborative DL inference framework leveraging both end-devices and edge computing platforms; the work [27] provides machine learning inference service on the edge computing platform and improves the inference quality by considering the uncertainty of inference workload;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3102945, IEEE Internet of Things Journal

recently, Microsoft's Data Box edge [28] releases a new functionality that allows users to deploy machine learning model on its edge devices and deliver machine learning inference service. In stark contrast, our work investigated the possibility of using the DNN ensemble technique, which provides state-of-the-art performance to many DL applications [15], on edge computing platforms. This is a new topic that has not been studied by existing works.

B. DNN Ensemble Prediction and Ensemble Selection

The ensemble prediction/forecasting is a longstanding machine learning strategy and the earliest work can be traced back to 1979 [29]. The key is to combine several base models strategically to produce a better predictive model. This strategy reduces the variance of predictions and therefore gives more robust and accurate prediction results [30]. The ensemble prediction technique has also contributed to the development of DL in recent years. For example, authors in [14] create a DNN ensemble by averaging the output of multiple individual DNNs, which far outperforms existing benchmarks in terms of inference accuracy. Because averaging outputs of individual DNNs is probably sub-optimal, more advanced ensemble/fusion rules are further investigated. For example, the work [31] uses weighted averaging to combine outputs of individual DNNs and proposes a learning scheme to determine the weight of DNNs. Authors in [32] use meta-learning and train a neural network to combine the outputs of individual DNNs. Besides higher inference accuracy, recent works [17] further show that using the DNN ensemble technique effectively defends adversarial attacks and therefore provides more robust and secure DL inference services. Note that our work is not simply applying the DNN ensemble technique on edge computing platforms, but involves a unique DNN ensemble selection problem due to the limited computing capacity of edge servers and the response deadline of DL inference services.

While predicting with DNN ensembles is widely used, DNN ensemble selection is still an under-investigated topic in the existing literature. DNN ensemble selection is somehow related to the model selection problem which aims to identify one single DNN model that provides the best inference performance. Several works have been done on model selection for on-device DL inferences. For example, the work [33] shows that different DNNs have different accuracy and delay performances, and a model selector is trained to choose the best-fit DNN. The authors in [34] design a big/little DNN framework where a little DNN is used whenever possible and a big DNN is used only when the confidence of the little DNN is less than a predefined threshold. However, selecting DNN ensembles is much more complicated than selecting one DNN model because it is difficult to analytically capture the inter-dependency among multiple models in the DNN ensemble. There is still no consensus in the community on how to build an optimal DNN ensemble, and the performance of a DNN ensemble depends heavily on the test data [35]. This paper designs a novel ensemble selection rule. It considers the accuracy of individual DNNs and the diversity of DNN members in the ensemble, which are empirically recognized as two key factors for constructing high-quality DNN ensembles [36], [37]. Because the DNN accuracy is unknown for user inference tasks, our work also uses online learning methods to learn the inference accuracy of DNNs.

III. System Models

A. Edge Computing Platforms

Our DL inference framework is compatible with most edge computing platforms. Let us consider a typical multi-access edge computing (MEC) system [8], [38]. A set of edge servers are deployed in a service area, and each edge server is co-located with a wireless access point. Users can offload their DL inference tasks to reachable edge servers via wireless connections. The edge system operator manages computing resources on edge servers with virtualization techniques, e.g., virtual machines (VMs) and containers. To provide DL inference service, a DL service provider rents computing resources (e.g., VMs) from edge servers using certain rental mechanisms [39]. With allocated computing resources, the DL service provider can configure its DNNs and application programming interfaces on the edge server for processing user inference tasks. In addition, the edge computing platform is able to accommodate multiple DL service providers. Different DL service providers will be allocated with isolated computing resources for deploying their own DL services. The resource scheduling for multiple DL service providers [40] is a problem orthogonal to the theme of this paper. Our goal is to enhance DL inference performance with DNN ensemble techniques. This problem is independent for a DL service provider once the computing resources on an edge server have been allocated. Without loss of generality, we present our method for one DL service provider on one edge server. Fig. 1 illustrates the implementation scenario.

B. DNN Ensemble Selection and Implementation Constraints

The DL service provider collects a set of DNNs, indexed by $\mathcal{M} = \{1, 2, ..., M\}$ at the edge server. We assume that all available DNNs can be stored on the edge server since the storage is less likely to be a constraint due to its low price nowadays. The operational timeline for the DL service provider is discretized into time slots $t = 1, 2, \ldots, T$ (e.g., a few seconds per slot). We let $\mathcal{X}^t = \{x_1^t, x_2^t, \ldots, x_{K^t}^t\}$ be the set of inference tasks received from users in time slot

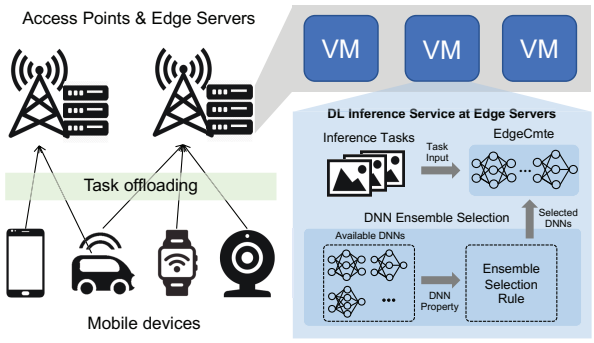

Fig. 1: Illustration of the implementation scenario.

t, where K^t is the total number of received tasks. Note that we consider our problem for one DL service provider, and therefore the received inference tasks request the same type of DL service, and the data size of tasks is a predetermined constant. As aforementioned, the user inference tasks exhibit different features due to the impact of external environment, which affects the performance of DNNs in different ways (we validate this claim via a simple trial in the experiment setup Section V-A). Therefore, instead of using a constant EdgeCmte, we adaptively change EdgeCmte based on the features of received tasks $\mathcal{X}^t$. We let $\mathcal{S}^t \subseteq \mathcal{M}$ denote the EdgeCmte selected by the DL service provider in time slot t. DNNs in the selected EdgeCmte will be loaded into the edge server RAM to process inference tasks.

While EdgeCmte improves inference accuracy and robustness, it also increases the space and time complexity. The resource usage and delay for running an EdgeCmte depend on the implementation scheme. For example, if DNNs in EdgeCmte are run in parallel, then a large amount of computing resources are required to load all DNNs in RAM at the same time. If DNNs are run sequentially (e.g., one by one), then less computing resource is required because only one DNN is loaded in RAM at a time. But using sequential execution will need more time to complete the inference of all DNNs. Besides these two basic schemes, other implementation schemes can also be applied and most of them are compatible with our framework. Given a certain implementation scheme, we let $c(\mathcal{S})$ and be the computing resource usage and $d^{\mathrm{F}}(\mathcal{S})$ be the inference delay for running EdgeCmte $\mathcal{S}$. It is intuitive to see that $c(\cdot)$ and $d^{\mathrm{F}}(\cdot)$ are non-decreasing functions of the ensemble size $|\mathcal{S}|$ (i.e., the number of DNNs in the EdgeCmte). Due to the limited computing resources at edge servers and the potential requirement on the service response time, EdgeCmte may not able to include all available DNNs.

1) Computing Resource Constraint: Let $\bar{c}$ be the amount of computing resources available at the edge server. Therefore, for a feasible EdgeCmte $\mathcal{S}$, its resource usage should not exceed the computing resource constraint $\bar{c}$, i.e., $c(\mathcal{S}) \leq \bar{c}$.

2) Response Deadline Constraint: We let $d^t(\mathcal{S}^t) = d^{\mathrm{T},t} + d^{\mathrm{F},t}(\mathcal{S}^t)$ denote the service delay given the selected EdgeCmte $\mathcal{S}^t$. The service delay consists of transmission delay $d^{\mathrm{T},t}$ and inference delay $d^{\mathrm{F},t}(\mathcal{S}^t)$. As discussed above, the inference delay is determined by the selected EdgeCmte $\mathcal{S}^t$. The transmission delay is incurred due to task offloading and result return over the wireless network. Because the decision for DNN ensemble selection is made after the offloading process is completed. At that time, tasks have been received at the edge server, and therefore the task offloading delay can be directly obtained by observing timestamps of the data packets that record when they are sent and received. However, we cannot get delays for result return using timestamps because the result return has not happened when we perform DNN ensemble selection. Therefore, we use the expected downlink rate to calculate its expected value. Because the data size of inference results is usually small, the transmission delay is dominated by the task offloading delay, and using the expected delay for result return will not cause large errors. In this case, the transmission delay $d^{\mathrm{T},t}$ is actually a known value when solving the DNN ensemble selection problem. Let $\bar{d}$ denote the service response deadline, the constraint for the inference delay is $d^{\mathrm{F}}(\mathcal{S}^t) \leq \bar{d} - d^{\mathrm{T},t}$.

C. Utility Definition and Problem Formulation

Running inference with EdgeCmtes, we need a fusion rule to combine individual outputs of DNNs to generate a final inference result. The fusion rules are often different for different service applications (e.g., image classification, object detection, and regression) because their output formats are different. In Appendix A, we give several examples of fusion processes for different application services. Without loss of generality, we will take classification-related applications as an example to define the utility function and present our algorithm. However, our method can be easily extended to other application services with very slight modifications (see Appendix A for more discussions). Let $\hat{y}^t_{m,k}$ be the output of DNN m for task x^t_k. If an EdgeCmte contains only DNN m, then the final inference result $\hat{y}^t_k$ for task x^t_k is $\hat{y}^t_k \leftarrow \hat{y}^t_{m,k}$. However, if an EdgeCmte contains more than one DNN, we will need a fusion rule to combine outputs of multiple DNNs. There exist various fusion rules for classification problems, e.g., majority voting and confidence averaging [31], and we do not want to confine our method to a specific fusion rule. Therefore, we generalize the fusion process into a general function $\mathcal{F}$. It takes the outputs of DNNs in the selected EdgeCmte, i.e., $\{\hat{y}^t_{m,k}\}_{m\in\mathcal{S}^t}$, as input and gives the final inference result $\hat{y}^t_k$ for task x^t_k, i.e., $\hat{y}^t_k \leftarrow \mathcal{F}(\{\hat{y}^t_{m,k}\}_{m\in\mathcal{S}^t})$.

The DL service provider derives rewards by serving users' inference requests. We define the reward of processing a task as the correctness of its inference result. Mathematically, for tasks x^t_k, its reward is defined by

$$u(x^t_k, \mathcal{S}^t; \mathcal{F}) = \mathbf{1}\left\{\mathcal{F}(\{\hat{y}^t_{m,k}\}_{m\in\mathcal{S}^t}) = y^t_k\right\}, \tag{1}$$

where y^t_k is the ground truth of task x^t_k and $\mathbf{1}\{\cdot\}$ is an indicator function. Given the set of admitted tasks $\mathcal{X}^t$, the total reward in time slot t is

$$U(\mathcal{X}^t, \mathcal{S}^t; \mathcal{F}) = \sum\nolimits_{x^t_k \in \mathcal{X}^t} u(x^t_k, \mathcal{S}^t; \mathcal{F}). \tag{2}$$

Our goal to maximize the cumulative reward in a total of T time slots by choosing a sequence of EdgeCmtes $\{\mathcal{S}^t\}_{t=1}^T$. This gives our DNN ensemble selection problem:

$$\mathscr{P}1: \max_{\{\mathcal{S}^t\}_{t=1}^T} \sum\nolimits_{t=1}^T U(\mathcal{X}^t, \mathcal{S}^t; \mathcal{F}) \tag{3a}$$

$$\text{s.t.} \quad c(\mathcal{S}^t) \leq \bar{c}, \forall t \tag{3b}$$

$$d^{\mathrm{F}}(\mathcal{S}^t) \leq \bar{d} - d^{\mathrm{T},t}, \forall t \tag{3c}$$

$$\mathcal{S}^t \subseteq \mathcal{M}, \forall t \tag{3d}$$

Recall that (3b) is the constraint posed by the computing capacity of edge servers and (3c) is the constraint on the response deadline. The key difficulty in solving $\mathscr{P}1$ is the obscured process of ensemble prediction. The performance of EdgeCmte is jointly determined by its constituting DNNs, and can be affected by various factors, e.g., the accuracy of individual DNNs, diversity among DNNs, and orthogonality\complementary of DNNs' training data [41].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3102945, IEEE Internet of Things Journal

Some of these factors are difficult to be characterized analytically, and some of them, e.g., DNNs' training data, are probably unknown in our problem. What's more thorny is that the impact of these factors is not entirely understood. As a result, it is extremely difficult to provide an analytical solution to $\mathscr{P}1$. Fortunately, the existing literature [17], [35], [42] has provided valuable empirical experiences on constructing good DNN ensembles, which inspire us to design effective heuristic solutions for DNN ensemble selection.

D. Heuristic Rules for DNN Ensemble Selection

Previous works [17], [35], [42] show that the accuracy and diversity of base models are two vital factors for constructing high-quality ensemble predictors. The accuracy of individual models determines the worst-case performance of the constructed ensemble predictor, and therefore including models with the highest accuracy in ensemble predictors often produces good performances [35]. Besides, it is widely accepted that an ensemble predictor generalizes better with diverse members. Therefore, we propose a heuristic rule for solving problem $\mathscr{P}1$. The core idea of our heuristic rule is to construct EdgeCmtes with DNNs that have the highest accuracy and exhibit large diversity for received tasks.

We first decouple the $\mathscr{P}1$ into subproblems, one for each time slot t as follows:

$$\max_{\mathcal{S}^t} \quad \sum\nolimits_{x_k^t \in \mathcal{X}^t} u(x_k^t, \mathcal{S}^t; \mathcal{F}) \quad \text{s.t.} \quad (3b), (3c), (3d). \tag{4}$$

Next, we give the heuristic rule for the above per-slot problem. Let $a_m(x_k^t)$ denote the accuracy of DNN m for task x_k^t, our heuristic rule can be written as:

$$\max_{\mathcal{S}^t} \quad \left(\frac{1}{K^t}\sum_{k=1}^{K^t}\sum_{m\in\mathcal{S}^t} a_m^t(x_k^t)\right) + \kappa \cdot \Phi(\mathcal{S}^t, \mathcal{X}^t), \tag{5a}$$

$$\text{s.t.} \quad (3b), (3c), (3d). \tag{5b}$$

where the first term in (5a) considers the accuracy of individual DNNs in EdgeCmte, and the second term is the diversity of DNNs in EdgeCmte given the received tasks $\mathcal{X}^t$. The objective is to maximize a weighted sum of two terms with their importances adjusted by weight κ. Note that the diversity measure $\Phi(\cdot,\cdot)$ in (5a) is for now given in a general form. We will give a detailed definition of $\Phi(\cdot,\cdot)$ later.

The heuristic rule provides us a way to identify best-fit EdgeCmtes, however, it cannot be directly applied in practice because the DNN accuracy, i.e., $a_m^t(x_k^t)$, for user tasks is unknown. Although it is possible to measure DNN accuracy on standard test data, one cannot guarantee that the distribution of user tasks is the same as that of standard data. The actual DNN accuracy is revealed during implementation and needs to be learned over time. In the next section, we cast the DNN ensemble selection into an online learning problem and use Multi-armed Bandit (MAB) techniques to provide a solution.

IV. Automated EdgeCmte Selection via Contextual Combinatorial Multi-armed Bandit

We use the framework of Contextual and Combinatorial Multi-armed Bandit (CC-MAB) [43] to design our Automated DNN Ensemble Selection (AES) algorithm. AES is "contextual" because it leverages the side-information (i.e., context) associated with inference tasks to learn the DNN accuracy, and it is "combinatorial" because it selects multiple DNNs to form an EdgeCmte. To show our algorithm, we first introduce context-parameterized accuracy.

Our method uses context of inference tasks to facilitate the learning of DNNs' accuracy. We consider simple contexts that can be directly observed without processing tasks (e.g., with images as inputs, the context can be image resolution, contrast, and noise). Using these simple context will not incur much extra computation burdens to edge servers. We let $\omega_k^t \in \Omega$ denote the context of task x_k^t, where Ω is the context space. We slightly abuse the notation of DNN accuracy $\{a_m(x_k^t)\}_{m\in\mathcal{M}}$ by defining the context-parameterized accuracy $\{a_m(\omega_k^t)\}_{m\in\mathcal{M}}$, i.e., the accuracy of DNN m for task x_k^t depends on context ω_k^t. We let $\mu_m(\omega_k^t) = \mathbb{E}[a_m(\omega_k^t)]$ be the expected accuracy of DNN m for inference tasks with the context ω_k^t.

A. Oracle Solution

Before presenting our method, we first show the oracle solution for DNN ensemble selection by assuming existence of an oracle that knows the expected accuracy $\mu_m(\omega_k^t)$ of an arbitrary DNN m for an arbitrary task x_k^t with an arbitrary context ω_k^t. Given the oracle, we replace $a_m(x_k^t)$ in (5) with the expected context-parameterized accuracy $\mu_m(\omega_k^t)$.

$$\max_{\mathcal{S}^t} \quad \left(\frac{1}{K^t}\sum_{k=1}^{K^t}\sum_{m\in\mathcal{S}^t} \mu_m(\omega_k^t)\right) + \kappa \cdot \Phi(\mathcal{S}^t, \mathcal{X}^t), \tag{6a}$$

$$\text{s.t.} \quad (3b), (3c), (3d). \tag{6b}$$

Based on the context-parameterized accuracy, we define the diversity function $\Phi(\cdot,\cdot)$. The diversity among DNNs is measured by how DNNs serve differently for tasks with different context. Mathematically, the diversity is defined by:

$$\Phi(\mathcal{S}^t, \mathcal{X}^t) = \frac{1}{K^t} \sum_{\{m,n\}\in\binom{\mathcal{S}^t}{2}} \left\| \amalg\left(\boldsymbol{\mu}_m(\boldsymbol{\omega}^t)\right) - \amalg\left(\boldsymbol{\mu}_n(\boldsymbol{\omega}^t)\right)\right\|^2,$$

where $\boldsymbol{\mu}_m(\boldsymbol{\omega}^t) = \left\{\mu_m(\omega_k^t)\right\}_{k=1}^{K^t}$ collects the expected context-parameterized accuracy of DNN m for tasks in $\mathcal{X}^t$, $\binom{\mathcal{S}^t}{2}$ denotes all 2-element combinations of DNNs in $\mathcal{S}^t$, and $\amalg(\cdot)$ is a normalization process to eliminate the impact of DNN accuracy which has already been considered in the first term of (6a). The ensemble diversity is a distance-based measure that calculates the Euclidean distance between any two DNNs' normalized context-parameterized accuracy for received tasks. This value reflects how DNNs work differently for tasks with different contexts. Therefore, the Euclidean distance is very suitable for measuring the diversity of DNNs in our problem.

The ensemble selection rule in (6) is a combinatorial optimization problem. When the number of available DNNs $|\mathcal{M}|$ is small, we can use brutal force to find the optimal solution. When $|\mathcal{M}|$ is large, greedy algorithms can be used to derive a 1/2-approximate solution in polynomial time. We let $\mathcal{S}^{*,t}$ denote the EdgeCmte selected by the oracle in time slot t.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3102945, IEEE Internet of Things Journal

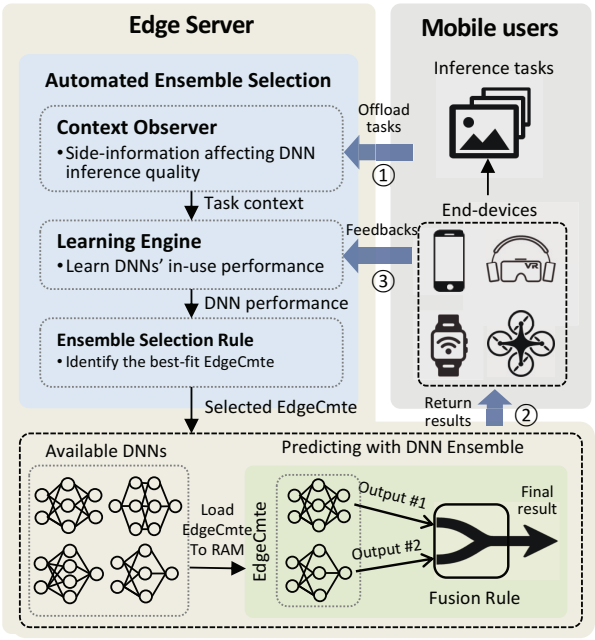

Fig. 2: Block diagram of AES.

B. Online Learning with CC-MAB

In the previous subsection, we have already presented our DNN ensemble selection rule with oracle information on the context-parameterized accuracy $\mu_m(\omega_k^t)$. However, such information is unknown in practice, and therefore a learning algorithm is needed. Next, we design our online learning algorithm, Automated EdgeCmte Selection (AES). AES uses the framework of contextual combinatorial MAB (CC-MAB). In each time slot t, AES operates as follows: (i) the edge server observes the context ω_k^t of each admitted task $x_k^t \in \mathcal{X}^t$. (ii) An EdgeCmte $\mathcal{S}^t$ is determined based on the observed context and DNN accuracy learned in previous time slots. (iii) Inference tasks are forwarded into each DNN in EdgeCmte, and the outputs of individual DNNs are fused to final results that are returned to end-users. (iv) At the end of the time slot, the ground-truths of inference tasks are observed, and then AES updates the estimated accuracy for DNNs in the selected EdgeCmte. Fig. 2 gives a block diagram of AES.

Intuitively, we want to select the best-fit EdgeCmte in each time slot using the DNN ensemble selection rule in (5). However, an underlying pre-condition for the selected EdgeCmte to deliver its expected performance is that the accuracy estimation for DNNs is accurate enough. To precisely estimate the context-parameterized accuracy for DNNs, the learner needs to collect an adequate amount of DNNs' inference correctness for tasks with various contexts. Note that the inference result of a DNN can be collected only when it is selected in EdgeCmte for processing admitted tasks. Therefore, instead of selecting the best-fit EdgeCmte, the learner sometimes needs to select a non-optimal DNN in order to collect its inference results for better accuracy estimation. Considering this, our AES algorithm has two phases, *exploration* and *exploitation*, with each phase targeting its own purpose. In exploration, the learner selects a DNN in order to collect its inference results and learn its accuracy for a particular context. In exploitation,

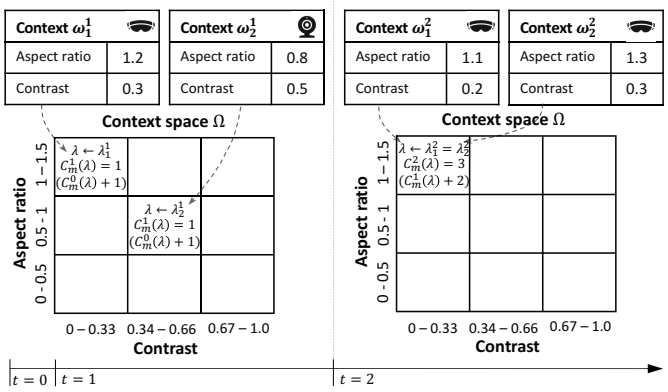

Fig. 3: Illustration of context partition and counter update.

the learner aims to select the best-fit EdgeCmte using the ensemble selection rule in (5) based on the learned accuracy. An important designing goal of AES is to balance exploration and exploitation.

The pseudocode of AES is presented in Algorithm 1. Without loss of generality, we assume the context of inference tasks is from a bounded context space that can be written as $\Omega \triangleq [0,1]^D$, where D is the dimension of context space. Due to the continuous context space, it is impossible to collect inference results of DNNs for each possible context $\omega \in \Omega$. To make the problem tractable, we consider the natural phenomena that DNNs will have similar performance for inference tasks with similar context, which is formalized by the Hölder condition:

Definition 1 (Hölder Condition)**.** There exists $L > 0$, $\alpha > 0$ such that for any DNN $m \in \mathcal{M}$ and any context $\omega, \omega' \in \Omega$, it holds that $|\mu_m(\omega) - \mu_m(\omega')| \leq L\|\omega - \omega'\|^\alpha$, where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^D$.

With Hölder condition, we can learn DNN accuracy for a group of inference tasks with similar context, which significantly improves the learning efficiency. AES starts by creating a partition Λ on the context space Ω, which splits Ω into h^D hypercubes $\lambda \in \Lambda$ with identical size of $\frac{1}{h} \times \cdots \times \frac{1}{h}$. The hypercubes can be understood as groups of similar context. The parameter h is an important algorithm parameter to be designed for determining the number of hypercubes in the partition. For each admitted task x_k^t, AES determines a hypercube λ_k^t that context ω_k^t belongs to, i.e., $\omega_k^t \in \lambda_k^t$ holds. For each DNN $m \in \mathcal{M}$ and hypercube $\lambda \in \Lambda$, AES keeps two counters, counter $C_m^t(\lambda)$ and counter $E_m^t(\lambda)$. $C_m^t(\lambda)$ records the number of tasks processed by DNN m up to time slot t whose context belong to hypercube λ; and $E_m^t(\lambda)$ records the number of tasks correctly classified by DNN m up to time slot t whose context belongs hypercube λ. Fig. 3 illustrates the context partition and the update process of counters.

In time slot t, the estimated accuracy of DNN m for tasks with context in hypercube λ is calculated by:

$$\hat{a}_m^t(\lambda) = E_m^t(\lambda)/C_m^t(\lambda) \tag{7}$$

For example, consider a tasks x_k^t with context $\omega_k^t \in \lambda_k^t$, the estimated accuracy of DNN m for x_k^t is $\hat{a}_m(\lambda_k^t)$.

Algorithm 1 Automated EdgeCmte Selection

Input: time horizon T, algorithm parameters h, $V(t)$;
Initialization create partition Λ; set $C_m(\lambda) = 0, E_m(\lambda) = 0, \forall \lambda \in \Lambda_T, m \in \mathcal{M}$;
for $t = 1, \ldots, T$ **do**
  Observe context ω_k^t of each task $x_k^t \in \mathcal{X}^t$ and find context hypercube λ_k^t such that $\omega_k^t \in \lambda_k^t$ holds;
  Identify $\mathcal{M}^{\mathrm{ue},t}$ as defined in (9);
  if $\mathcal{M}^{\mathrm{ue},t} \neq \emptyset$ **then** ▷ *Exploration*
    if $c(\mathcal{M}^{\mathrm{ue},t}) \leq \bar{c}$ and $d^{\mathrm{F}}(\mathcal{M}^{\mathrm{ue},t}) \leq \bar{d} - d^{\mathrm{T},t}$ **then**
      Get $\mathcal{S}'^{,t}$ by solving the problem in (11);
      $\mathcal{S}^t \leftarrow \mathcal{S}'^{,t} \cup \mathcal{M}^{\mathrm{ue},t}$;
    else
      Get $\mathcal{S}^t$ using the rule in (12);
  else ▷ *Exploitation*
    Get $\mathcal{S}^t$ by solving the problem in (10);
  Observe the ground truth of the inference tasks;
  for each $m \in \mathcal{S}^t$ and $\lambda \in \{\lambda_k^t\}_{k=1}^{K^t}$ **do** ▷ *Updates*
    Update counter $C_m^t(\lambda)$ and $E_m^t(\lambda)$;
    Update estimation: $\hat{a}_m(\lambda) = E_m^t(\lambda)/C_m^t(\lambda)$;

In each time slot, AES is either in exploration or exploitation. To determine the correct phase, AES first checks whether the context hypercubes have been sufficiently explored to give a precise accuracy estimation. For DNN m, its under-explored hypercubes are identified based on the counter $C_m(\lambda)$:

$$\Lambda_m^{\mathrm{ue},t} = \left\{\lambda \in \Lambda \;\middle|\; C_m^t(\lambda) < V(t)\right\} \tag{8}$$

where $V(t)$ is a control function to determine whether the hypercube has been sufficiently explored, and it needs to be appropriately designed to balance exploration and exploitation. Based on the under-explored hypercubes of DNNs $\Lambda_m^{\mathrm{ue},t}, \forall m$, we define under-explored DNNs in time slot t as:

$$\mathcal{M}^{\mathrm{ue},t} = \left\{m \in \mathcal{M} \middle|\; \exists\, x_k^t \in \mathcal{X}^t, \lambda_k^t \in \Lambda_m^{\mathrm{ue},t}\right\} \tag{9}$$

This definition indicates that the estimated accuracy of an under-explored DNN is not precise enough for at least one of the admitted tasks. The phase of AES in time slot t is determined based on the under-explored DNNs $\mathcal{M}^{\mathrm{ue},t}$.

Exploitation: If the set of under-explored DNNs is empty, $\mathcal{M}^{\mathrm{ue},t} = \emptyset$, it means that all DNNs are sufficiently explored to give precise accuracy estimation. Then, AES uses the estimated accuracy in (5):

$$\max_{\mathcal{S}^t} \quad \left(\frac{1}{K^t}\sum_{k=1}^{K^t}\sum_{m\in\mathcal{S}^t}\hat{a}_m^t(\lambda_k^t)\right) + \kappa\cdot\hat{\Phi}(\mathcal{S}^t,\mathcal{X}^t) \tag{10a}$$

$$\text{s.t.} \quad (3b), (3c), (3d). \tag{10b}$$

Note that the estimated accuracy $\hat{a}_m(\lambda_k^t)$ is used in (10) instead of the oracle information $\mu_m(\omega_k^t)$, and $\hat{\Phi}(\cdot,\cdot)$ is the diversity measure based on the estimated accuracy.

$$\hat{\Phi}(\mathcal{S}^t,\mathcal{X}^t) = \frac{1}{K^t}\sum_{\{m,n\}\in\binom{\mathcal{S}^t}{2}}\left\|\amalg\left(\hat{\boldsymbol{a}}_m(\boldsymbol{\lambda}^t)\right) - \amalg\left(\hat{\boldsymbol{a}}_n(\boldsymbol{\lambda}^t)\right)\right\|^2,$$

where $\hat{\boldsymbol{a}}_m(\boldsymbol{\lambda}^t) := \{\hat{a}_m(\lambda_k^t)\}_{k=1}^{K^t}$ collects the estimated accuracy for all received tasks.

Exploration: When the set of under-explored DNNs is non-empty, i.e., $\mathcal{M}^{\mathrm{ue},t} \neq \emptyset$, AES enters exploration and tries to select DNNs in $\mathcal{M}^{\mathrm{ue},t}$ for acquiring more precise accuracy estimation. If $c(\mathcal{M}^{\mathrm{ue},t}) \leq \bar{c}$ and $d(\mathcal{M}^{\mathrm{ue},t}) \leq \bar{d}$, then all the under-explored DNNs can be included in EdgeCmte. It is possible that the computing resource of the edge server is not fully used, and therefore we can include more DNNs in addition to the under-explored DNNs:

$$\max_{\mathcal{S}'^{,t}} \quad \frac{1}{K^t|\mathcal{S}'^{,t}|}\sum_{m\in\mathcal{S}'^{,t}}\sum_{k=1}^{K^t}\hat{a}_m^t(\lambda_k^t) + \kappa\cdot\hat{\Phi}(\mathcal{S}'^{,t},\mathcal{X}^t) \tag{11a}$$

$$\text{s.t.} \quad c\left(\mathcal{S}'^{,t}\cup\mathcal{M}^{\mathrm{ue},t}\right) \leq \bar{c}, \tag{11b}$$

$$d^{\mathrm{F}}\left(\mathcal{S}'^{,t}\cup\mathcal{M}^{\mathrm{ue},t}\right) \leq \bar{d} - d^{\mathrm{T},t}, \tag{11c}$$

$$\mathcal{S}'^{,t} \subseteq \mathcal{M}\backslash\mathcal{M}^{\mathrm{ue},t}. \tag{11d}$$

In this case, the selected EdgeCmte is $\mathcal{S}^t = \mathcal{S}'^{,t} \cup \mathcal{M}^{\mathrm{ue},t}$. If $c(\mathcal{M}^{\mathrm{ue},t}) > \bar{c}$ or $d^{\mathrm{F}}(\mathcal{M}^{\mathrm{ue},t}) > \bar{d} - d^{\mathrm{T},t}$, we will randomly remove DNNs from $\mathcal{M}^{\mathrm{ue},t}$ until the reduced set satisfies the constraints of computing capacity and response deadline. The EdgeCmte $\mathcal{S}^t$, in this case, is determined by

$$\text{Initial:} \quad \mathcal{S}^t \leftarrow \mathcal{M}^{\mathrm{ue},t} \tag{12a}$$

$$\text{Repeat:} \quad \mathcal{S}^t \leftarrow \mathcal{S}^t \backslash \left\{s \overset{R}{\leftarrow} \mathcal{S}^t\right\} \tag{12b}$$

$$\text{Until:} \quad c(\mathcal{S}^t) \leq \bar{c} \text{ and } d^{\mathrm{F}}(\mathcal{S}^t) \leq \bar{d} - d^{\mathrm{T},t} \tag{12c}$$

C. Algorithm Parameter Design and Performance Analysis

Now, we design two important parameters h and $V(t)$ in AES. The parameter design affects the algorithm performance which is measured by the reward loss compared to oracle (termed as *regret*). Let $\{\mathcal{S}^{*,t}\}_{t=1}^T$ and $\{\mathcal{S}^t\}_{t=1}^T$ be the EdgeCmtes selected by oracle and AES, respectively. The regret is defined as:

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K^t} u\left(x_k^t, \mathcal{S}^{*,t}; \mathcal{F}\right) - u\left(x_k^t, \mathcal{S}^t; \mathcal{F}\right)\right].$$

The goal of AES is to achieve a sublinear regret $R(T) = O(T^\gamma)$ with $\gamma < 1$, which means that AES is asymptotically optimal since $\lim_{T\to\infty} R(T)/T \to 0$. Because the impact of DNN diversity on the inference correctness is difficult to characterize analytically, we simplify the diversity term to a constant, i.e., for any EdgeCmte $\mathcal{S}$ with $|\mathcal{S}| \geq 2$, we assume $\Phi(\mathcal{S},\mathcal{X}) = \Phi^{\mathrm{c}}$. Based on this, the theorem below gives the design of h and $V(t)$, and the regret upper bound of AES.

Theorem 1 (Bound for Regret $R(T)$)**.** Let $V(t) = t^{\frac{2\alpha}{3\alpha+D}}\log(t)$ and $h = \lceil T^{\frac{1}{3\alpha+D}}\rceil$. If AES is run with these parameters, Hölder condition holds true, and $\Phi(\mathcal{S},\mathcal{X}) = \Phi^{\mathrm{c}}$, $\forall\mathcal{S}$ with $|\mathcal{S}| \geq 2$, then the leading order of regret $\mathbb{E}[R(T)]$ is $O\left(T^{\frac{2\alpha+D}{3\alpha+D}}\log(T)\right)$.

Proof. See Appendix C in the supplemental file. □

Theorem 1 indicates that the regret of AES is sublinear in the time horizon T. Moreover, the theorem provides a bound on the performance loss for any finite time slot. This can be used to characterize the convergence speed of AES. Although our analytical results rely on the simplification of ensemble

diversity, we show in the experiment that AES can still provide sublinear regret in practice when $\Phi(\mathcal{S}, \mathcal{X}) = \Phi^c$ does not hold.

Complexity analysis: Next, we analyze the space and time complexity of the proposed algorithm. The space complexity of AES is determined by the number of hypercubes in the partitioned context space. According to the algorithm design in Theorem 1, parameter h for context space partition is set to $\lceil T^{\frac{1}{3\alpha+D}} \rceil$, which means that the number of hypercubes in the context partition is $h^D = \lceil T^{\frac{1}{3\alpha+D}} \rceil^D$. Each context hypercube maintains an accuracy estimation. Because there are M candidate DNNs, the total number of accuracy estimations maintained by AES is $M\lceil T^{\frac{1}{3\alpha+D}} \rceil^D$. However, the actual memory usage can be smaller because an accuracy estimation is created for a context hypercube only after contexts of some received tasks fall in that context hypercube.

The time complexity of AES mainly lies in updating accuracy estimation for context hypercubes. The proposed algorithm uses recursive averaging for accuracy estimation, and we let $O(1)$ denote the time complexity of one recursive averaging operation. Note that the accuracy estimation of a DNN is updated only when it is selected in EdgeCmte, and the update is performed for the context hypercubes to which the contexts of received tasks belong (see Fig. 3 for illustration). Therefore, at most $K_{\max}M$ accuracy estimations will be updated, where $K_{\max}$ is the maximum number of inference tasks received in one time slot. This indicates that the time complexity is $O(K_{\max}M)$. The actual time complexity can be smaller in practice because not all DNNs are selected in the EdgeCmte and contexts of certain inference tasks may fall into the same context hypercube.

V. Experiments

A. Experiment Setup

1) Edge Computing Testbed: A DELL workstation is used as an edge server. It is equipped with a 64-bit 8 Intel i7-4770 CPU cores running at 3.40GHz, and one NVIDIA Geforce GTX 1080 Ti GPU. The experiment is run on Ubuntu 16.04 LTS system with Pytorch v1.1.0, CUDA v9.0, cuDNN v7.0. In each time slot, tasks are sent to the edge server in a batch-wise manner. The average size of task batches is 10.

2) Data Sources and DNN Models: Two datasets are used in our experiment: Caltech101 [20] and MASATI-v2 [21]. The Caltech dataset has 101 categories. A category can contain 40 to 800 images, and the size of images is roughly 300 × 200 pixels. The MASATI dataset collects images of maritime scenes under different weather and illumination conditions. It contains 7,389 satellite images which are categorized into seven classes: land, coast, sea, ship, multi, coast-ship, and detail. We conduct experiments separately on each dataset. To run our algorithm, we first need to generate a set of available DNNs and store them on the edge server. DNNs differ from each other in two aspects: 1) the training data, and 2) the DNN architecture. We three DNN architectures, Googlenet [49], Resnet50 [14], and Mobilenet-v2 [50], respectively. These DNNs have been pre-trained by the ImageNet repository [51] and will be further retrained with Caltech101 and MASATI-v2 datasets. Each dataset (Caltech101 or MASATI-v2) will be pre-processed in different ways to generate different data sources. To be specific, the pre-processing modifies the noise and resolution of images in the dataset. The original dataset is denoted as ORI. We create two noisy data sources by adding white Gaussian noise with variance 0.1 (referred to as NOS-1) and 0.2 (referred to as NOS-2), respectively. We also create two data sources with different image resolutions by down-sampling original images to the resolution of 40×80 (referred to as RES-1) and 80×150 (referred to as RES-2), respectively. Each data source is separated into two parts, training data and test data. The training data contains 60% of the images, and the rest 40% are used as test data to evaluate our method. We leave a considerable amount of data for testing because our method involves an online learning process. We retrain Mobilenet-v2, Googlenet, and Resnet50 using the training data of each data source. This imitates the scenario that collected DNNs are trained by different data sources. Given this experimental setting, AES uses the noise and resolution of images as context information. This setting is very realistic because the noise and resolution of images depends heavily on user devices, e.g., smartphones, surveillance cameras, and etc. Table I lists system and algorithm parameters used in the experiment.

TABLE I: Experimental setup.

Parameters	Settings
Datasets	Caltech101, MASATI-v2
DNN architecture	Googlenet, Resnet50, Mobilenetv2
# of available DNNs	$\lvert\mathcal{M}\rvert = 9$
Average size of task batch	10
α of Hölder condition	0.3
Context dimension D	2
Context information	Image noise, Image resolution
Time horizon T	900 (MASATI), 1100 (Caltech)
GPU memory capacity	200MB
Response deadline	300ms
Average wireless transmission rate	20Mbps

3) Property of Individual DNNs: We next show some properties of individual DNNs. Fig. 4 illustrates a simple trial where three DNNs with the architecture of Googlenet, Resnet50, and Mobilenet-v2, are trained by the training data of Caltech/MASATI ORI, Caltech/MASATI NOS-1, and Caltech/MASATI NOS-2, respectively. Each trained DNN is evaluated on the test data of Caltech/MASATI ORI, Caltech/MASATI NOS-1, and Caltech/MASATI NOS-2. Table II and Table III shows the accuracy of individual DNNs generated on the Caltech and MASATI dataset, respectively. There is a clear trend that a DNN achieves higher accuracy on the test data that comes from the same data source as its training data. We can also see that DNNs tend to perform better on the test dataset that is similar to its training data, e.g., DNN trained on ORI has higher accuracy on NOS-1 (low noise) than that on NOS-2 (high noise). We train available DNNs on ORI, RES-1, and RES-2 datasets in the same way, and give their accuracy performance in Appendix B in the supplementary file. Similar trends can also be observed.

B. Results and Evaluations of AES

We compare AES with four benchmarks:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3102945, IEEE Internet of Things Journal

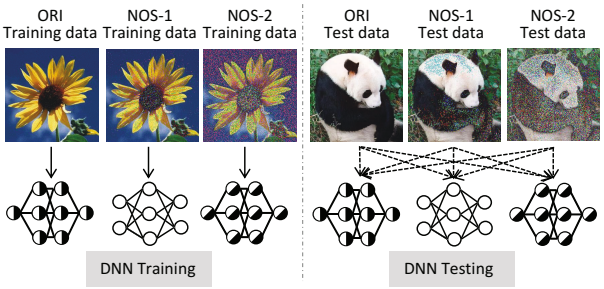

Fig. 4: Illustration of experimental setup.

TABLE II: Accuracy of individual DNNs (Caltech dateset).

Train on \ Test on		Caltech ORI	Caltech NOS-1	Caltech NOS-2
Caltech ORI	Googlenet	**0.828**	0.2755	0.048
	Resnet50	**0.930**	0.303	0.052
	Mobilenetv2	**0.898**	0.263	0.035
Caltech NOS-1	Goolenet	0.470	**0.655**	0.308
	Resnet50	0.222	**0.806**	0.386
	Mobilenetv2	0.294	**0.709**	0.294
Caltech NOS-2	Goolenet	0.056	0.338	**0.434**
	Resnet50	0.014	0.418	**0.584**
	Mobilenetv2	0.017	0.361	**0.433**

TABLE III: Accuracy of individual DNNs (MASATI dataset).

Train on \ Test on		MASATI ORI	MASATI NOS-1	MASATI NOS-2
MASATI ORI	Googlenet	**0.8244**	0.5618	0.3221
	Resnet50	**0.9029**	0.5036	0.1916
	Mobilenetv2	**0.8084**	0.4196	0.1499
MASATI NOS-1	Goolenet	0.6564	**0.7442**	0.6264
	Resnet50	0.3609	**0.8388**	0.6990
	Mobilenetv2	0.3550	**0.7349**	0.4373
MASATI NOS-2	Goolenet	0.3512	0.5103	**0.6843**
	Resnet50	0.1524	0.2410	**0.7108**
	Mobilenetv2	0.1292	0.2786	**0.5741**

1) Oracle: Oracle knows the context-parameterized accuracy of DNNs a priori. In each time slot, it selects an EdgeCmte using the DNN ensemble selection rule in (5).

2) UCB1: UCB1 is a classic MAB algorithm that selects one action in each time slot. We create super-actions, i.e., all feasible DNN ensemble decisions, and use UCB1 to learn the reward of each super-arm.

3) Best-single: It selects one single DNN that delivers the highest accuracy for admitted tasks in each time slot. This method is used as a baseline to validate the efficacy of DNN ensemble technique.

4) Random: It randomly selects B DNNs to construct an EdgeCmte in each time slot, where B is the maximum number of DNNs that can be included in an EdgeCmte given the computing capacity and response deadline.

The α value in Hölder Condition is set to 0.3. The computing capacity is 200MB. The response deadline is set to 300ms. We use accuracy-weighted confidence as the fusion rule to combine outputs of individual DNNs. Detailed descriptions for accuracy-weighted confidence can be found in Appendix A-A. We first run the experiment with one-dimension context space, where AES only observes the noise of images as context. (Caltech or MASATI) ORI, NOS-1, and NOS-2 datasets are used to generate 9 DNNs as described in the experiment setup

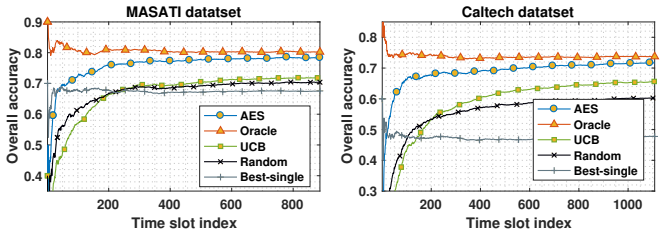

Fig. 5: Comparison of accuracy evolution.

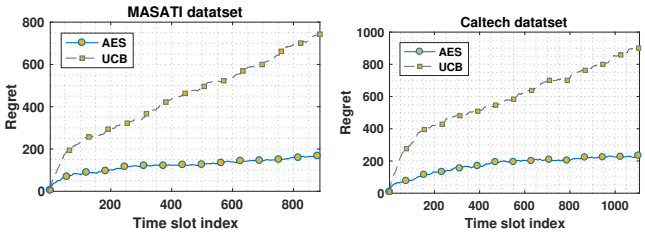

Fig. 6: Comparison of algorithm regret.

(Section V-A). AES and other benchmarks run on the test data of ORI, NOS-1, and NOS-2. Later in this section, we will show the results when running with two-dimension context space.

1) Comparison of Inference Accuracy: Fig. 5 compares the accuracy achieved by AES and the other four benchmarks on the Caltech and MASATI dataset. As expected, Oracle achieves the highest accuracy for both datasets. Among the others, AES outperforms other benchmarks, and achieves close-to-oracle accuracy. We see on the Caltech dataset that the accuracy of Best-single converges to 48.0%, by contrast, AES is able to achieve 72.0% accuracy which is a 24.0% accuracy increase compared to Best-single. Also, it can be observed on both datasets that using Random can provide higher accuracy than Best-single, indicating that using the DNN ensemble technique can effectively improve the DL inference accuracy.

2) Regret Analysis: Fig. 6 compares the regret of two online learning algorithms, AES and UCB. It can be clearly observed that the regret of UCB increases linearly over time, which means that UCB cannot identify the optimal EdgeCmte as the learning proceeds. By contrast, AES gives a sublinear regret, which means that the performance of AES is asymptotically optimal compared to Oracle. It should be noted that the assumption $\Phi(\mathcal{S}, \mathcal{X}) = \Phi^c$ is not guaranteed when running the experiment on these two real-world datasets, but AES can still provide a sublinear regret.

3) Benefit of Ensemble Diversity: In Fig. 7, we evaluate the benefit of considering ensemble diversity during DNN ensemble selection. For the benchmark *AES w/o Diversity*, we set weight κ in (6) to 0 such that ensemble diversity Φ will not affect DNN ensemble selection. In this case, *AES w/o Diversity* selects DNNs that have the highest accuracy for received tasks. We can see from Fig. 7 that AES improves the accuracy by 78.5% − 73.8% = 4.7% for MASATI and 72.8% − 68.9% = 3.9% for Caltech compared to *AES w/o Diversity*. This indicates that promoting diversity of DNNs in EdgeCmte improves DL inference accuracy.

4) Impact of Computing Capacity: Fig. 8 evaluates the impact of computing capacity. The resource constraint in this experiment is GPU RAM (VRAM). We vary VRAM from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3102945, IEEE Internet of Things Journal

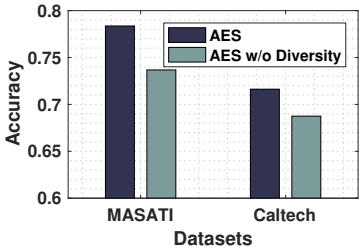

Fig. 7: Impact of ensemble diversity.

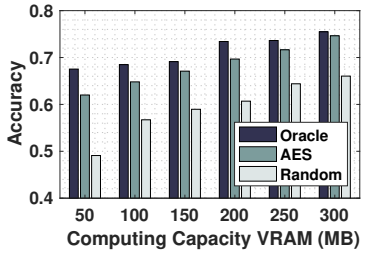

Fig. 8: Impact of computing capacity.

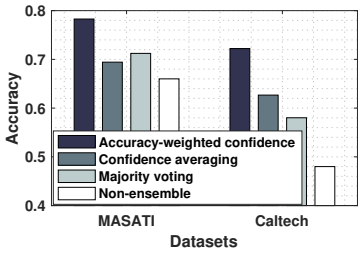

Fig. 9: Performance of fusion rules.

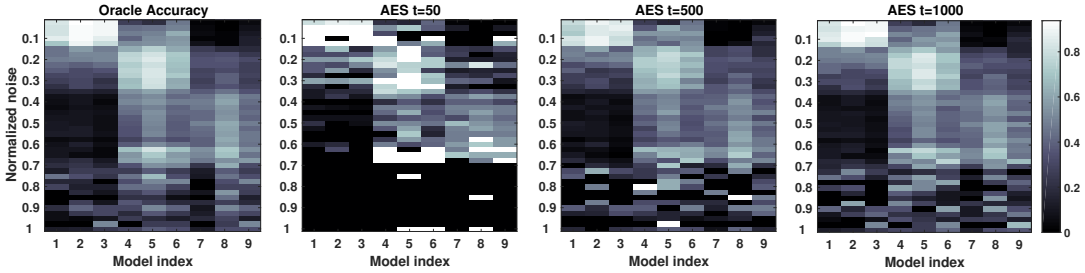

Fig. 10: Estimation of context-parameterized accuracy.

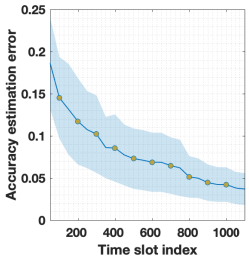

Fig. 11: Estimation errors.

50MB to 300MB and Fig. 8 shows the accuracy achieved by Oracle, AES, and Random on the Caltech dataset. In general, we see that all three methods achieve higher accuracy with more computing resources available at the edge server. This is because the DL service provider can recruit more DNNs in the EdgeCmte, which tends to produce more robust inference results. It is also worth noticing that the gap between AES and Random decreases with the increase in computing resources. For example, when the computing capacity is 50MB, the accuracy improvement provided by AES over Random is 14%; when the computing capacity becomes 300MB, the accuracy improvement provided by AES over Random decreases to 6%. This means that learning is more necessary when the computing resource is limited.

5) Performance of Fusion Rules: Besides the accuracy-weighted confidence, we also applied our method with other commonly used fusion rules, e.g., confidence averaging and majority voting. Fig. 9 compares accuracy achieved by AES when applied with accuracy-weighted confidence, confidence averaging [31], and majority voting. We see that accuracy-weighted confidence achieves the highest accuracy among the applied fusion rules. While the performance of confidence averaging and majority voting is worse than that of accuracy-weighted confidence, it still provides noticeable performance improvement compared with the non-ensemble scheme. We also see that the performance of fusion rules depends on the applied datasets, e.g., majority voting is better than confidence averaging on MASATI, but is worse on Caltech.

6) Estimation of Context-parameterized Accuracy: Fig. 10 shows the estimation of context-parameterized accuracy for all 9 available DNNs. The y-axis is the partition created on the context normalized noise, the x-axis is model index, and each color cube denotes the accuracy value. We depict the oracle accuracy and the accuracy value estimated by AES at $t = 50$, $t = 500$, $t = 1000$. We can see that AES can effectively learn the accuracy of DNNs for tasks with different contexts. The estimated accuracy is almost the same with oracle accuracy at $t = 1000$. Fig. 11 shows the average accuracy estimation error for context hypercubes when running AES. We see that the estimation error diminishes over time.

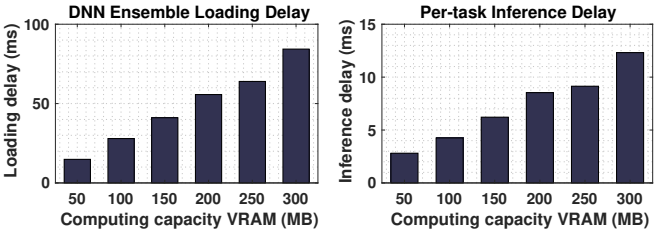

Fig. 12: Service delay of AES.

7) Ensemble Loading Delay, Inference Delay, and Task Offloading delay: Fig. 12 shows the DNN ensemble loading delay and per-task inference delay. The DNN ensemble loading delay is the time consumed for loading DNNs to GPU RAM for processing the received tasks. The figure on the left side shows that the average DNN ensemble loading delay varies from 15ms to 82ms depending on the computing capacity. An EdgeCmte often recruits more DNNs when the computing capacity is large and as a result, it takes more time to load the EdgeCmte. The task inference delay is the delay between feeding tasks to an EdgeCmte and generating of inference results. The figure on the right side shows that the per-task inference delay can be as low as 3ms and the highest per-task inference delay is 12.2ms.

8) Higher Context Space Dimension: We next increase the dimension of context space. To be specific, AES observes the noise and resolution of images as two pieces of context information. In this experiment, we generate two DNNs (with the architecture of Googlenet and Resnet50) on each of (Caltech or MASATI) ORI, NOS-1, NOS-2, RES-1, and RES-2 datasets. Therefore, there is a total of 10 available DNNs. The α value in Hölder Condition is set to 0.3. In Fig. 13, we

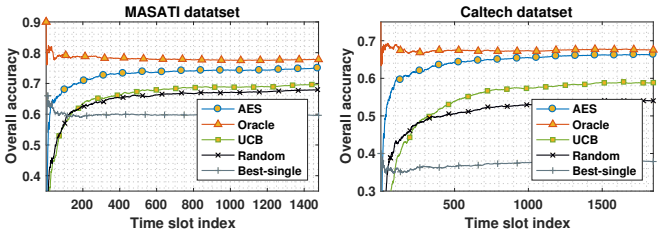

Fig. 13: Comparison of accuracy with 2-dimension context.

give a comparison of accuracy achieved by AES and other benchmarks. The general trend in Fig. 13 is similar to that in Fig. 5, where AES achieves close-to-oracle performances. It is worth noticing is that Best-single performs much worse in this case. This is because the feature of inference tasks may vary across more contexts, and it becomes difficult for one DNN to work well for inference tasks with diverse features.

VI. Conclusions

In this paper, we investigated the possibility of using the DNN ensemble technique on edge computing platforms to enhance the performance of DL inference services. A unique problem, DNN ensemble selection, was defined and studied. The goal of DNN ensemble selection is to identify a subset of DNNs for constructing a DNN ensemble that works the best for user inference tasks subject to the limited computing capacity of edge servers and the possible deadline of response time. An online learning algorithm called Automated Ensemble Selection (AES) was proposed to solve the DNN ensemble selection problem. It uses the context information associated with inference tasks to learn the DNN accuracy and selects the best-fit DNN ensemble based on the learned knowledge. The proposed algorithm is practical and easy to implement. It also guarantees asymptotic optimality.

References

[1] D. Sima, "AppleâĂŹs mobile processors," 2018.

[2] T. Lite, "Android to launch tensorflow lite for mobile machine learning," 2017.

[3] Core ml: Integrate machine learning models into your app. [Online]. Available: https://developer.apple.com/documentation/coreml.

[4] X. Xie and K.-H. Kim, "Source compression with bounded dnn perception loss for iot edge computer vision," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[5] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–199.

[6] C.-J. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia *et al.*, "Machine learning at facebook: Understanding inference at the edge," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 331–344.

[7] 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 *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2018, pp. 389–400.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv:1701.01090*, 2017.

[9] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

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

[11] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[12] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

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

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] (2017) Large scale visual recognition challenge (ilsvrc). [Online]. Available: http://www.image-net.org/challenges/LSVRC/

[16] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie *et al.*, "Adversarial attacks and defences competition," in *The NIPS'17 Competition: Building Intelligent Systems*. Springer, 2018, pp. 195–231.

[17] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 4970–4979.

[18] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[19] M. Naumov, D. Mudigere, and et al, "Deep learning recommendation model for personalization and recommendation systems," *CoRR*, vol. abs/1906.00091, 2019. [Online]. Available: https://arxiv.org/abs/1906.00091

[20] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[21] A. P. Antonio-Javier Gallego and P. Gil, "Automatic ship classification from optical aerial images with convolutional neural networks," *Remote Sensing*, vol. 10, no. 4, 2018.

[22] Y. Han, X. Wang, V. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *arXiv preprint arXiv:1907.08349*, 2019.

[23] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.

[24] X. Li, Y. Qin, H. Zhou, Y. Cheng, Z. Zhang, and Z. Ai, "Intelligent rapid adaptive offloading algorithm for computational services in dynamic internet of things system," *Sensors*, vol. 19, no. 15, p. 3423, 2019.

[25] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[26] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint multi-user dnn partitioning and computational resource allocation for collaborative edge intelligence," *IEEE Internet of Things Journal*, 2020.

[27] Y. Jin, L. Jiao, Z. Qian, S. Zhang, N. Chen, S. Lu, and X. Wang, "Provisioning edge inference as a service via online learning," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2020, pp. 1–9.

[28] (2019) What is azure data box edge? [Online]. Available: https://docs.microsoft.com/zh-cn/azure/databox-online/data-box-edge-overview

[29] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[30] R. Polikar, "Ensemble learning," in *Ensemble machine learning*. Springer, 2012, pp. 1–34.

[31] Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 2015, pp. 435–442.

[32] Y. Xiao, J. Wu, Z. Lin, and X. Zhao, "A deep learning-based multi-model ensemble method for cancer prediction," *Computer methods and programs in biomedicine*, vol. 153, pp. 1–9, 2018.

[33] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive selection of models on embedded systems," *arXiv preprint arXiv:1805.04252*, 2018.

[34] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo, "Big/little deep neural network for ultra low power inference," in

Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis. IEEE Press, 2015, pp. 124–132.

[35] S. Rothe and D. Söffker, "Comparison of different information fusion methods using ensemble selection considering benchmark data," in *2016 19th International Conference on Information Fusion (FUSION)*. IEEE, 2016, pp. 73–78.

[36] L. Liu, W. Wei, K.-H. Chow, M. Loper, E. Gursoy, S. Truex, and Y. Wu, "Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2019, pp. 274–282.

[37] S. Gu and Y. Jin, "Generating diverse and accurate classifier ensembles using multi-objective optimization," in *2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*. IEEE, 2014, pp. 9–15.

[38] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[39] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2364–2376, 2019.

[40] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic service placement in geographically distributed clouds," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 762–772, 2013.

[41] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[42] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 206–219, 2011.

[43] L. Chen, J. Xu, and Z. Lu, "Contextual combinatorial multi-armed bandits with volatile arms and submodular reward," in *Advances in Neural Information Processing Systems*, 2018, pp. 3247–3256.

[44] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167–176.

[45] J. Howe, *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House, 2008.

[46] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 748–756.

[47] A. Mehrtash, W. M. Wells, C. M. Tempany, P. Abolmaesumi, and T. Kapur, "Confidence calibration and predictive uncertainty estimation for deep medical image segmentation," *IEEE Transactions on Medical Imaging*, 2020.

[48] C. Tekin, J. Yoon, and M. Van Der Schaar, "Adaptive ensemble learning with confidence bounds," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 888–903, 2017.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[51] L. Fei-Fei, J. Deng, and K. Li, "Imagenet: Constructing a large-scale image database," *Journal of vision*, vol. 9, no. 8, pp. 1037–1037, 2009.

[52] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

Yang Bai received her Ph.D. degree in the College of Engineering, University of Miami, USA, in 2021. She received the B.S. degree from Northeastern University, Shenyang, China, and the M.S. degree from the College of Engineering, University of Miami. Her primary research interests include machine learning, computer vision, and game theory.

Lixing Chen is an Assistant Professor in Institute of Cyber Science and Technology at Shanghai Jiao Tong University, China. He received the Ph.D. degree in Electrical and Computer Engineering from the University of Miami in 2020, and the BS and ME Degrees from the College of Information and Control Engineering, China University of Petroleum, Qingdao, China, in 2013 and 2016, respectively. His primary research interests include mobile edge computing and machine learning for networks.

Mohamed Abdel-Mottaleb (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Maryland, College Park, in 1993. He joined the University of Miami in 2001. He is currently a Professor and the Chairman of the Department of Electrical and Computer Engineering. His research focuses on biometrics, visual tracking, human activity recognition, and medical image processing. From 1993 to 2000, he was with Philips Research, Briarcliff Manor, NY, USA, where he was a Principal Member of the Research Staff and a Project Leader. At Philips Research, he led several projects in image processing and content-based multimedia retrieval. He represented Philips in the standardization activity of ISO for MPEG-7, where some of his work was included in the standard. He holds 22 U.S. patents and more than 30 international patents. He has published more than 150 journal and conference papers in the areas of image processing, computer vision, and contentbased retrieval. He serves as the Editor-in-Chief for the IEEE BIOMETRICS COMPENDIUM.

Jie Xu (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively, and the Ph.D. degree in electrical engineering from UCLA in 2015. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Miami. His research interests include mobile edge computing/intelligence, machine learning for networks, and network security. He received the NSF CAREER Award in 2021.