IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 December 2022, accepted 19 December 2022, date of publication 23 December 2022, date of current version 5 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3232073

RESEARCH ARTICLE

AI-Empowered Fast Task Execution Decision for Delay-Sensitive IoT Applications in Edge Computing Networks

BESTE ATAN[1], MEHMET BASARAN[2,3], NURULLAH CALIK[4], SEMIHA TEDIK BASARAN[1], GULDE AKKUZU[5], AND LUTFIYE DURAK-ATA[3]

[1]Department of Electronics and Communication Engineering, Istanbul Technical University, 34469 Istanbul, Turkey
[2]Kartal Research and Development Center, Siemens Sanayi Ticaret A.S., 34870 Istanbul, Turkey
[3]Information and Communications Research Group, Informatics Institute, Istanbul Technical University, 34469 Istanbul, Turkey
[4]Department of Biomedical Engineering, Istanbul Medeniyet University, 34700 Istanbul, Turkey
[5]Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany

Corresponding author: Beste Atan (besakkuzu@itu.edu.tr)

This work was supported by Umlaut Communications GmbH, Aachen, Germany.

ABSTRACT As the number of smart connected devices increases day by day, a massive amount of tasks are generated by various types of Internet of Things (IoT) devices. Intelligent edge computing is a promising enabler in next-generation wireless networks to execute these tasks on proximate edge servers instead of smart devices. Additionally, regarding the execution of tasks in edge servers, smart devices could provide a low-latency environment to the end users. Within this paper, an artificial intelligence (AI)-empowered fast task execution method in heterogeneous IoT applications is proposed to reduce decision latency by taking into account different system parameters such as the execution deadline of the task, battery level of devices, channel conditions between mobile devices and edge servers, and edge server capacity. In edge computing scenarios, the number of task requests, resource constraints of edge servers, mobility of connected devices, and energy consumption are the main performance considerations. In this paper, the AI-empowered fast task decision method is proposed to solve the multi-device edge computing task execution problem by formulating it as a multi-class classification problem. The extensive simulation results demonstrate that the proposed framework is extremely fast and precise in decision-making for offloading computation tasks compared to the conventional Lyapunov optimization-based algorithm results by ensuring the guaranteed quality of experience.

INDEX TERMS AI, classification, computation offloading, intelligent networks, Internet of Things, Lyapunov optimization, machine learning, multi-access edge computing.

I. INTRODUCTION

By 2030, it is reported that our lives and industry will be reshaped and billions of smart devices are expected to be connected to the network [1], [2]. The majority of these devices could provide various types of applications on the network edge that will be a part of smart cities, smart transportation, or smart gaming [3]. Especially, virtual reality (VR), augmented reality (AR)-based applications, live video streams from unmanned aerial vehicles (UAVs), and smart autonomous connected vehicles produce a great amount of data and require rapid response to their requests [4]. As IoT devices are built with limited computation, storage, and energy availability [5], they are not appropriate for processing huge data on their own efficiently. On the other hand, current conventional cloud computing environments which are centralized far away from the user environment are not adequate to provide uninterrupted services demanded by various IoT applications [6], [7], [8].

One of the key solutions to address the aforementioned challenges is transferring task computing processing to edge servers that are closely located to the users, called

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

multi-access edge computing (MEC) [9], [10], [11]. MEC provides additional computation capabilities and storage while meeting strict performance requirements such as low latency, energy consumption, or higher bandwidth according to the task specifications [12]. Even if edge computing is a promising solution, introducing a new node to the network brings complexity to the task offloading process between IoT devices and edge servers where traditional optimization algorithms struggle to solve such complex decision problems [6], [13]. As a result, networks must be managed intelligently in order to meet end-user expectations and efficiently utilize resources. Fortunately, the growth of learning techniques in artificial intelligence (AI) technology assists to develop new approaches to solve complicated challenges [14]. Additionally, academies and industries are highly attracted to the adaptation of machine learning (ML) and deep learning (DL) techniques in next-generation wireless networks [15]. The integration of AI algorithms into the edge network environment helps to accelerate decision-making approaches and fulfill the latency-sensitive requirements of different IoT applications.

In recent years, some studies related to ML and DL algorithms in cellular and IoT networks are investigated in literature [2], [7], [15], [16]. Studies show that the implementation of ML and DL plays a significant role in the computation offloading to the edge environment and resource management in IoT networks, and latency is one of the critical parameters for the effective and efficient usage of various IoT applications. Especially for delay-sensitive applications, learning-based task decision mechanisms have a high potential to accelerate task offloading decisions since ML techniques promptly respond to the utilization of available resources.

In this article, we propose an AI-empowered fast task execution decision framework to accelerate edge computation offloading while taking into account channel conditions, time and energy consumption of energy-harvested IoT devices, the capacity of edge servers, and the mobility of users. Apart from our previous work [17], the system model in this study is designed to be feasible to serve different types of smart connected devices with latency-sensitive applications such as smartphones, connected cars, AR or VR-based online gaming by consoles, live video streaming by UAVs or any smart home applications, and to be fast to response time-sensitive task requests while considering limited capacity of multiple edge servers. To differentiate UAV-based task requests which are mostly addressed for very time-sensitive services, different channel models and very limited packet execution deadlines are also considered. In the system model, task execution modes are chosen as device execution, edge server execution, and task drop. The proposed intelligent task classification framework uses different ML and DL models to classify task execution modes which results in the acceleration of the task execution process. Additionally, using maximum channel gain with the output of the learning model accurately achieves the edge server selection process. Comprehensive performance results show that the proposed AI-enabled framework is significantly fast and precise in the decision-making process of edge computation offloading while guaranteeing the quality of user experience compared to traditional Lyapunov optimization methods [18].

The rest of the paper is organized as follows. Related Works in Section II presents a summary of recent research on edge computing offloading methods. The system model is introduced in Section III. Afterward, we demonstrate the intelligent task classification framework in Section IV. Next, we present the performance evaluation and results in Section V, and the final conclusions in Section VI.

II. RELATED WORKS

The European Telecommunications Standards Institute (ETSI) has developed MEC [19] to integrate cloud computing capabilities into wireless networks, which is expected to be critical in addressing next-generation network requirements. Edge servers, which are positioned at the edge of the network, can provide processing power, storage, connection, and other services to edge devices. However, since edge servers have limited resources for computation and storage, computation offloading and resource allocation have become popular topics. To alleviate this problem, different schemes are proposed [20], [21], [22]. Wang et al. [20] formulate computation offloading decision as a convex optimization problem while considering resource allocation and content caching in edge networks. Authors in [21] jointly consider the time and rate of task offloading assignment to solve the optimization problems and propose a heuristic approach as well. In [22], tasks are merged to minimize latency and guarantee the reliability of computation offloading among multiple-user and multiple-server environments. The aforementioned three researchers propose traditional optimization approaches in which their target is to optimize bandwidth and computer resources, and they obtained significant results. However, they do not consider any AI-based learning methods to leverage their approach.

In recent years, researchers have focused on learning-based edge computing offloading solutions instead of traditional optimization methods. In [23] an online learning base strategy is proposed while taking into account the completion latency of the tasks and the multi-hop edge environment for cooperative offloading. According to [24], the authors propose a new ML approach in edge computing to optimize multiple objective functions which are memory, processing, and communication requirements of the edge device. Even the authors present significant results in edge computing while considering computation and latency constraints; device diversity, user mobility, and energy are still key constraints in edge computing scenarios. On the other hand, technology combining IoT with AI is a promising opportunity to obtain reliable data in a complex environment for analysis and intelligent decision-making [25]. According to the method proposed in [26], an industrial IoT environment is considered where task offloading assignment is achieved by ML-based channel selection and context awareness of

resource allocation while [27] deep reinforcement learning (DRL) approach is proposed to decide the allocation of resources to multi-service IoT systems in a flexible way. On the other hand, [28] applies a machine learning algorithm for offloading task from UAVs to edge servers while minimizing the total processing time and energy consumption. Similarly, [29] considers a vehicular edge computing network to apply a DRL-based data offloading policy and minimize the delay of average offloading. It can be inferred from the simulation results of the above articles that learning-based algorithms perform better than traditional optimization algorithms in multi-device edge networks to optimize resource allocation by considering latency and energy constraints. Furthermore, in [30], to obtain low energy consumption, energy harvesting devices are considered to address resource allocation problems in edge environments.

In this paper, AI empowerment is mainly considered to optimize the decision-offloading strategy. Unlike previous research, this paper proposes a comprehensive framework to serve an energy-harvested multi-device environment while taking into account channel conditions, the capacity of edge servers, delay, energy consumption, and user mobility. Simulation results indicate how rapidly and accurately the proposed intelligent framework can achieve computation offloading decisions to edge servers while maintaining the quality of experience.

III. SYSTEM MODEL

This section describes the system model, which is comprised of various types of smart connected devices such as smartphones, online gaming consoles, UAVs, connected cars, or any connected devices of smart homes with energy harvesting (EH) capabilities, as well as multiple edge servers located near base stations (BS) or access points as illustrated in Figure 1. To represent the set of smart connected devices and edge servers, we assume K smart connected devices as $\mathcal{K} \triangleq \{1, 2, \ldots, K\}$ and M edge servers $\mathcal{M} \triangleq \{1, 2, \ldots, M\}$. As shown in Figure 1, all smart connected devices are randomly positioned in a specified region, including edge computing servers deployed near BSs and device locations that vary between time slots. In each time slot, connected users are allowed to move across the region. The distance between smart connected devices and edge servers in the n-th time slot is represented by $d_{i,j}^{n}$ where $n \in \mathcal{N}, i \in \mathcal{K}, j \in \mathcal{M}$.

In this model, edge computing servers are utilized to execute task requests originating from various smart devices, where all of these devices generate delay-sensitive task requests in IoT applications. After receiving a task request from users, the connected devices decide whether to execute computations locally or offload them to edge servers based on latency and energy usage.

In consideration of the nature of delay-sensitive task requests, offloading task computing to the edge server significantly improves the user experience. As a result, we split the time into slots in which the task request execution delay is not regarded to exceed the given time slot duration.

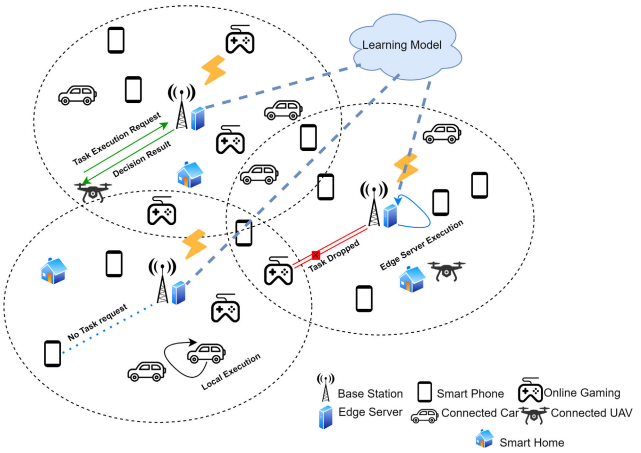


FIGURE 1. Multi-device supported edge computing system model for next generation wireless networks.

In each n-th time slot, we use the Rician wireless channel model for UAV types of users. Because usually, UAV-ground communications consist of more line-of-sight (LoS) links and some multiple paths due to scattering or reflection of ground buildings, trees, etc. [31]. So, Rician distribution fits well to a stronger line-of-sight environment while Nakagami-m distributions are utilized to model dense scatters. As a result, other channels are modeled as Nakagami-m fading among connected smart devices and edge servers. The probability density function (pdf) of the Rician distribution for UAV users can be given as

$$f_R(n) = \frac{(K+1)n}{\Omega} \exp\left(-K - \frac{(K+1)n^2}{\Omega}\right) I_0\left(2\sqrt{\frac{K(K+1)}{\Omega}}n\right) \tag{1}$$

where $I_0(\cdot)$ is the 0-th order modified Bessel function of the first kind, K is defined as Rician factor and Ω is the scale parameter [32]. The pdf of Nakagami-m model used for the rest of the connected devices can be given as

$$f_N(n; m, \Omega) = \frac{2m^m}{\Gamma(m)\Omega^m} n^{2m-1} \exp\left(-\frac{m}{\Omega}n^2\right) \tag{2}$$

in which $m \geq 1/2$ and for both Rician and Nakagami-m distributions $\Omega > 0$ and $n > 0$. Additionally, we assume that neither smart devices nor edge servers have a buffer to process task requests.

In this study, task execution decisions in smart connected devices or edge servers are modeled with respect to energy usage and execution length based on the model described in [17]. The task is executed locally if the smart connected device has enough energy to run the task on its own by the execution request deadline τ_x. Otherwise, if the edge server capacity is sufficient, task execution requests are offloaded to the edge servers. If neither of these two computational modes is acceptable for the task request execution, the computation task is dropped. Furthermore, the scenario in which the smart

connected device does not generate any request for computation is also taken into consideration as the main distinction unlike the existing literature including [18].

Moreover, task requests are assumed as L bits and modeled by independent and identically distributed (i.i.d.) Bernoulli processes [18]. Each task request is generated with probability ρ with values ranging between $0.7 \leq \rho \leq 1$. To identify whether a task request is generated or not a task request indicator is defined as $\zeta_i^n \in \{0, 1\}$. When one of the smart connected devices produces a task, the parameter ζ_i^n equals 1 at the n-th time slot where $n \in \mathcal{N}$ for this device. If $\zeta_i^n = 0$, the task is not generated by the i-th smart connected device.

We define four classes to indicate decision options for each task execution, where $i \in \{\mathrm{m}, \mathrm{s}, \mathrm{d}, \mathrm{n}\}$ denotes each class that belongs to the mobile, server, drop, and no request cases, respectively [17]. If the i-th smart device executes the task request on its own, the result of the class indicator is $C_{\mathrm{i,m}}[n] = 1$. Accordingly, if the task offloading to one of the edge servers is feasible, then $C_{\mathrm{i,s}}[n] = 1$. Furthermore, $C_{\mathrm{i,d}}[n] = 1$ is defined to represent task is dropped for the i-th device while $C_{\mathrm{i,n}}[n] = 1$ indicates task request is not generated at the n-th time arrival. As a consequence, the indicators for each of the four classes are represented as follows:

$$C_{\mathrm{i,d}}[n] + C_{\mathrm{i,n}}[n] + C_{\mathrm{i,m}}[n] + C_{\mathrm{i,s}}[n] = 1, \quad n \in \mathcal{N}. \quad (3)$$

Toward increasing the speed of the decision for task execution process with competing resources like time, energy, and capacity; task offloading decision from smart connected devices to the edge servers is more challenging than the scenarios where task is dropped or task is not generated from any devices. In this model, edge computing servers are positioned to execute task requests coming from multiple smart connected devices where delay-sensitive task requests are generated from all these types of devices. Once the task request is received from users, devices make decisions for computation execution in local or to offload remote servers mainly in terms of delay, energy consumption, and edge server capacity levels. Therefore, the execution models of devices and edge servers are explained comprehensively in the following two subsections.

A. CONNECTED DEVICE LOCAL EXECUTION MODEL

In the edge computing network supported by multiple devices, each device is considered to run L bits task locally. In the device execution model where we define $Z_l = LX_l$, which denotes all central processing unit (CPU) cycles required to complete the execution of the task on the smart connected device, and X_l is the number of CPU cycles taken to process one bit of data. By using these definitions, the delay cost for local task execution of each smart connected device is expressed as below:

$$D_{\mathrm{i,l}}[n] = \sum_{z=1}^{Z_l} \left(f_{i,z}[n]\right)^{-1}, \quad (4)$$

where we define maximum computation capacity of the end user in terms of CPU cycle frequency as $f_{\mathrm{l}}^{\max}$, i.e., $\forall z \in \{1, 2, \ldots, L\, X_{\mathrm{l}}\}, f_{i,z}^n \leq f_{\mathrm{l}}^{\max}, n \in \mathcal{N}, i \in \mathcal{K}$.

Moreover, the energy consumption due to task computation in each connected smart device is calculated as stated in [18] and [19]:

$$E_{\mathrm{i,l}}[n] = \kappa \sum_{z=1}^{Z} \left(f_{i,z}[n]\right)^2, \quad (5)$$

where κ presents the effective switched capacitance [33].

Additionally, EH devices have a better capacity to carry out computing activities, such as offloading to an edge server for execution or performing computation locally in the edge computing environment [34]. Therefore, in this study, the task execution decision process is customized to the EH approach to properly use renewable energy in smart connected devices. We assume that the energy that arrives $E_{i,H}[n]$ in each connected smart device is uniformly distributed, with the highest value of $E_{i,H}^{\max}$ satisfying $0 \leq e_{i,l}[n] \leq E_{i,H}[n]$, where $e_{i,l}[n]$ indicates the quantity of harvested energy.

B. EDGE TASK EXECUTION MODEL

Compared to the task execution ability of the local device, the edge server has a greater computing capability and a more consistent power supply than the local device. Therefore, in this model, a generated task is analyzed to see if the task might be offloaded to the edge server based on the duration of the task execution and smart connected device energy consumption with consideration of each edge server's capacity and channel conditions. When selecting the task execution computation mode, we assume that each task is only allocated to just one server. Accordingly, we define $s_{i,j}^n$ as a selection indicator where $s_{i,j}^n \in \{0, 1\}$. If the i-th smart connected device chooses the j-th edge server for offloading the task execution, the result is $s_{i,j}^n = 1$. To cover all edge server selection processes, the calculation is expressed as below:

$$\sum_{j=1}^{M} s_{i,j}^n = 1, n \in \mathcal{N}, i \in \mathcal{K}, j \in \mathcal{M}. \quad (6)$$

In the proposed UAV-integrated system model, we apply the Rician fading channel model for UAV-type smart connected devices to adequately address line-of-sight (LoS) links and scattering between UAVs and BSs and make the proposed solution more realistic [31], [35]. On the other hand, we exploit the commonly-adopted Nakagami-m distribution model which is known to capture fading of radio propagation and the fluctuations of various small-scale fading environments [36] to different types of smart connected devices besides UAVs in the proposed system model.

To express the data rate in the time slot n, we adopt Shannon's capacity calculation to our model as

$$R\left(h_{i,j}[n], P_i[n]\right) = B \log_2 \left(1 + \frac{|h_{i,j}[n]|^2 P_i[n]}{\sigma^2}\right), \quad (7)$$

where the transmit power of each smart connected device is defined as $P_i[n]$ with the maximum transmit power $P_i^{\max}$. Also, in this equation, $h_{i,j}[n]$ represents the channel gain between each smart connected device and edge server, B shows the allocated bandwidth to each device, and σ^2 is defined as noise power received at each edge server.

Additionally, in this edge task execution model, the task computing capabilities of edge servers are limited in terms of the connection capacity and required CPU cycles, which are defined as Z_s. To indicate the connection capacity of the edge servers, each available bandwidth of the edge server is split equally into N sub-bands and each device is assigned to one sub-band with ω MHz bandwidth within the corresponding time interval n. The corresponding limitation is evaluated as

$$\sum_{i\in\mathcal{K}} \mathrm{I}\left(I_i^n\right)\cdot s_{i,j}^n Z_{\mathrm{s}} \le f_{\mathrm{s}}^{\max} n, n\in\mathcal{N}, j\in\mathcal{M}, \tag{8}$$

where $f_{\mathrm{s}}^{\max}$ indicates the maximum amount of CPU cycle frequency of each server and we define the indicator function as $\mathrm{I}(\cdot)$. To add, execution latency on each edge server is ignored and the total delay equals to transmission latency of each task request.

As a result, we calculate the cost of transmission delay given below if the task is offloaded to the edge servers

$$D_{\mathrm{i,s}}[n] = \frac{L}{R\left(h_{i,j}[n], P_i[n]\right)}. \tag{9}$$

The energy consumption cost of each edge server is defined by

$$E_{\mathrm{i,s}}[n] = P_i[n]D_{\mathrm{i,s}}[n] = P_{i,m}[n]\frac{L}{R\left(h_{i,j}[n], P_i[n]\right)}. \tag{10}$$

It is worth noting that in this study, changes in the total battery energy level of each smart connected device are taken into consideration since they have an impact on the task execution process, which is mostly ignored in traditional mobile network systems.

IV. INTELLIGENT TASK CLASSIFICATION FRAMEWORK

A significant number of tasks are generated by IoT devices in which a variety of IoT applications offer latency-sensitive features. Therefore, it is critical to effectively handle the task execution process. In this paper, we focus on an AI-based classification strategy that has great potential for expediting task execution decisions, where ML approaches have a great ability to quickly utilize the available resources as opposed to traditional models. Thus, we present an intelligent fast task classification framework consisting of training and testing processes as shown in Figure 2.

In this section, we discuss how the number of tasks is generated in the IoT applications initially. After that, we explore dataset generation and task classification strategy for the proposed edge computing model. Then, we evaluate the edge server selection process to complete end to end framework.

A. DATASET GENERATION

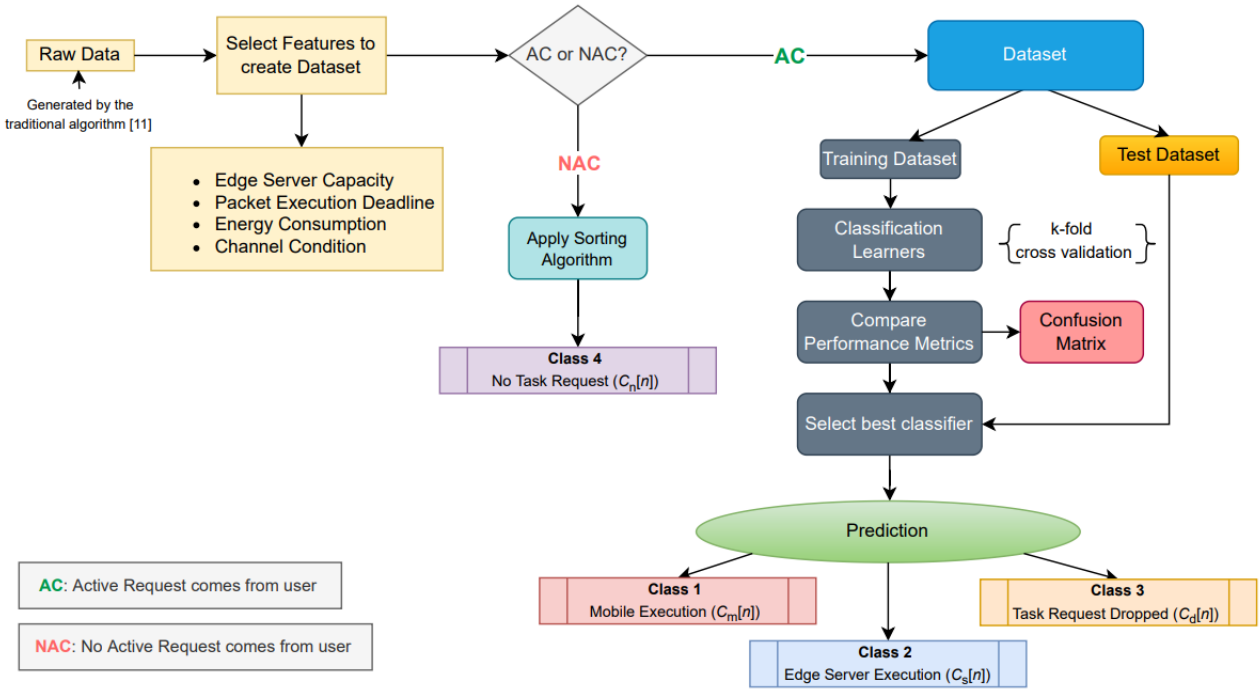
Optimizing cellular or IoT networks is challenging since the number of configurable parameters has increased dramatically over the years [37]. Data-driven ML algorithms are one option to provide automated solutions for IoT services. When the available data is large and multidimensional, ML or, more particularly, deep learning (DL) algorithms can be used to extract valuable features and classification of the data [16]. In this study, since multiple connected devices are assumed to be existed in the network with time-sensitive applications, various parameters play a role to determine offloading decisions in conventional optimization approaches which might lead to greater time delays. With the help of AI algorithms, we could utilize the limited number of input parameters to be used in the decision making process.

As illustrated in Figure 2, raw data which is considered as the input of the training process is generated based on our system model and using the Lyapunov Optimization based Dynamic Computation (LODCO) algorithm [18]. The raw data consists of a randomly generated number of samples within a particular time period. The same raw data is processed throughout the training and testing phases in order to compare classification performances. In the feature selection step, four main system parameters are chosen which are edge server capacity, packet execution deadline, energy consumption, and channel condition. These are selected as the main parameters to create a dataset to be processed by learning algorithms. The values or ranges of these parameters are defined in Table 1 which we take into account while creating the dataset. To add, these four parameters are randomly distributed in each time slot to get a randomized dataset and provide better prediction results from the learning process.

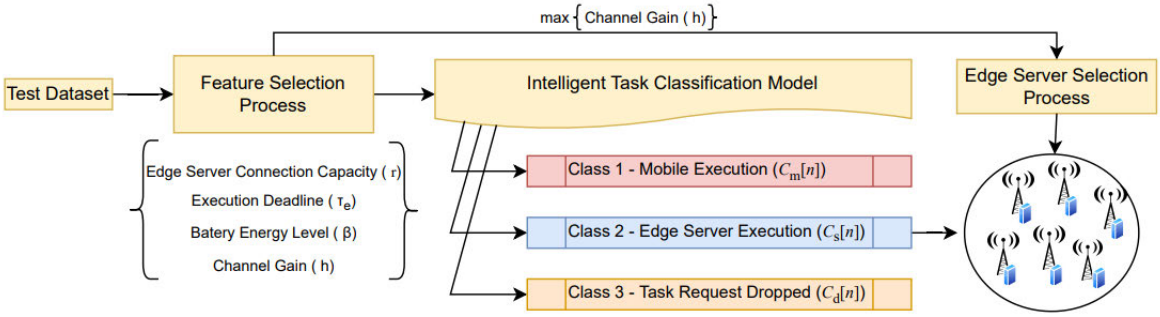
Furthermore, the first parameter of the system is the edge server connection capacity denoted by Γ_j^n, which refers to the number of users connected to each server in the n-th time slot. The second parameter is τ_i representing the execution deadline, where the task must be completed within the specified time period for each user. To define the battery energy level of each IoT device, we use β_i, which is randomly distributed for each user and used EH approach stated in System Model section. The final parameter represented by $h_{i,j}[n]$ is the gain in the wireless channel between each connected device and edge servers where the channel is modeled as Rician for UAVs and Nakagami for other users. Before the training dataset is generated, we apply a sorting algorithm to classify whether an active task request is received from users or not which is defined as an "Active Request" (AC) comes from

TABLE 1. System parameters for task classification process.

Parameters	Description	Value / Range
Γ_j^n	Edge server connection capacity	$\{0, 5\}$
τ_i	Execution deadline	$[10^{-3}, 15\times10^{-3}]$
β_i	Battery energy level	$[2\times10^{-5}, 2\times10^{-3}]$
$h_{i,j}[n]$,	Channel gain	Calculated subject to dist. n/a



(a) Training Process



(b) Testing Process

FIGURE 2. Task classification learning model framework for IoT applications in edge computing networks.

the user or "No Active Request" (NAC) comes from the user in Figure 2. The sorting algorithm labels the task as Class 4 ($C_{i,n}[n]$) at the n-th time slot if there is no request sent by users.) Note that our dataset is divided where 50% data points are used to train the multi-class ML algorithms while the other half is used to test the performance of the algorithms. Through this, we aim to identify effectively working classifier.

B. CLASSIFICATION LEARNING STRATEGY

The effective implementation of massively heterogeneous IoT networks relies heavily on network and resource management. In addition to resource allocation challenges, task management decisions play a pivotal role in implementing the heterogeneous IoT network environment [16]. Unlike traditional task management methods such as optimization models, and heuristics-based methods, ML approaches indicate the great potential for decision making where ML knows how to learn the variations, classify the conditions and forecast the results [38], [39].

In this study, we construct the ML model which includes producing the training samples, selecting informative features and classification of data. Once the model is trained, the test dataset can be classified according to preset classes as Class 1, Class 2, and Class 3 defined during training. We evaluate that the multi-class learning model predictions outperform the traditional optimization model, LODCO. To provide comprehensive findings over the dataset, decision trees (DT), ensemble learners (EL), k-Nearest Neighbors (kNN), support vector machine (SVM) and neural networks (NN) such as

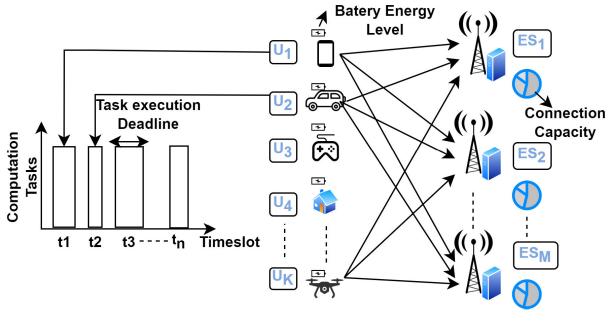


FIGURE 3. Illustration of simulation environment of the proposed system model where computation task is generated in each time slot and assigned to the suitable edge server.

shallow NN (SNN) and deep neural network (DNN) type of ML classifiers are compared, which are mainly used in the literature [2], [37], [40].

As shown in Figure 2 (a), the training model, raw data generated through LODCO algorithm is preprocessed before obtaining the extracted dataset. After partitioning the dataset as a training and test dataset we apply different kinds of classification learners such as DT, kNN, SVM, NN predicts class labels. After the execution of classification learners, we compare the performance metrics of each which are produced by confusion matrices. Then, the best classifier is selected according to the accuracy, precision, recall, and Macro F1 score results of the classifiers. Finally, the trained model is tested by the test dataset to predict other execution modes among local execution as Class 1, offloading execution as Class 2, and task dropping as Class 3 for each smart connected IoT device where we share performance results of classification learners in the performance evaluation section.

In Figure 2 (b), we illustrate the end-to-end testing model of our framework. The test dataset is similarly classified based on the pre-defined classes in the training model. Additionally, if the user is labeled Class 2, which refers to edge server execution mode, we apply the edge server selection process to identify which server executes the regarding task for the user. To accomplish the matching of the server, we use the maximum value of channel gain as $\max\{h_{i,j}[n]\}$. In the performance evaluation section, the highest results above 98% are obtained for this mapping. These scores show that channel gain is the best fit for the edge server selection process.

V. PERFORMANCE EVALUATION & RESULTS

In this section, we first describe evaluation metrics that give better comparison results between ML models, such as accuracy, precision, recall, and Macro F1 scores. Secondly, a case study is provided to explain the simulation environment. Finally, simulation results are demonstrated in terms of performance metric results of ML models, speed rate, and training time consumption measures including comparison with traditional techniques.

A. EVALUATION PROCESS

Based on the proposed system model and experimental design, ML algorithms are evaluated using the following performance indicators which are calculated by the confusion matrix (CM). Accuracy (ACC) in classification is an important parameter that indicates the classification success rate. The model performs better if the classification accuracy is higher. Therefore, in our evaluation, accuracy is one of the key performance metrics to select the best classifier and is described as

$$\text{ACC} = \frac{\sum_i \mathbf{K}_{ii}}{\sum_{i,j} \mathbf{K}_{ij}}, \tag{11}$$

where $\mathbf{K} \in \mathbb{R}^{4\times4}$ is the CM of a classifier obtained from the test samples. The rows and columns of the matrix are denoted as target and prediction, respectively.

Secondly, precision (PRE) and recall (RCL) indicators are defined to collect more comparable information between classifiers. It should be noted that if the dataset is unbalanced, the ACC is insufficient to display with respect to performance results. Thus, these metrics are critical to determining the actual performance of the system. PRE and RCL are calculated respectively as

$$\text{PRE}_i = \frac{\mathbf{K}_{ii}}{\sum_i \mathbf{K}_{ij}}, \tag{12}$$

$$\text{RCL}_i = \frac{\mathbf{K}_{ii}}{\sum_j \mathbf{K}_{ij}}. \tag{13}$$

Finally, the Macro F1 metric is described as below in terms of PRE and RCL values by using the corresponding formula

$$F_i = \frac{2 \times \text{PRE}_i \times \text{RCL}_i}{\text{PRE}_i + \text{RCL}_i} \tag{14}$$

and then, F_1^{macro} is generated as

$$F_1^{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} F_i, \tag{15}$$

which is the main comparison metric for the evaluation process.

B. CASE STUDY

In this section, we describe the simulation environment of our study based on Figure 3. We assume $K = 12$ smart connected IoT devices where two of them are UAVs and $M = 6$ edge servers to execute $L = 1000$ bits task according to the classification decision. As given in Table 1, each edge server has $\Gamma_j^n = 5$ connection capacity at each time slot. The execution deadline differentiates for UAV type of users, which is ($\tau_i \in [1, 5]$ ms) where tasks of other users must be executed in ($\tau_i \in [5, 15]$ ms). The battery energy level, β_i, is between $[2 \times 10^{-5}, 2 \times 10^{-3}]$ joule. To perform the simulations, we set time slot as $n = \{2000\}$ where the raw data is generated by 144000 samples which is processed throughout the training and testing phases, and the outcomes are shown in Figure 4, Table 2 and Table 3. In addition, we expand the

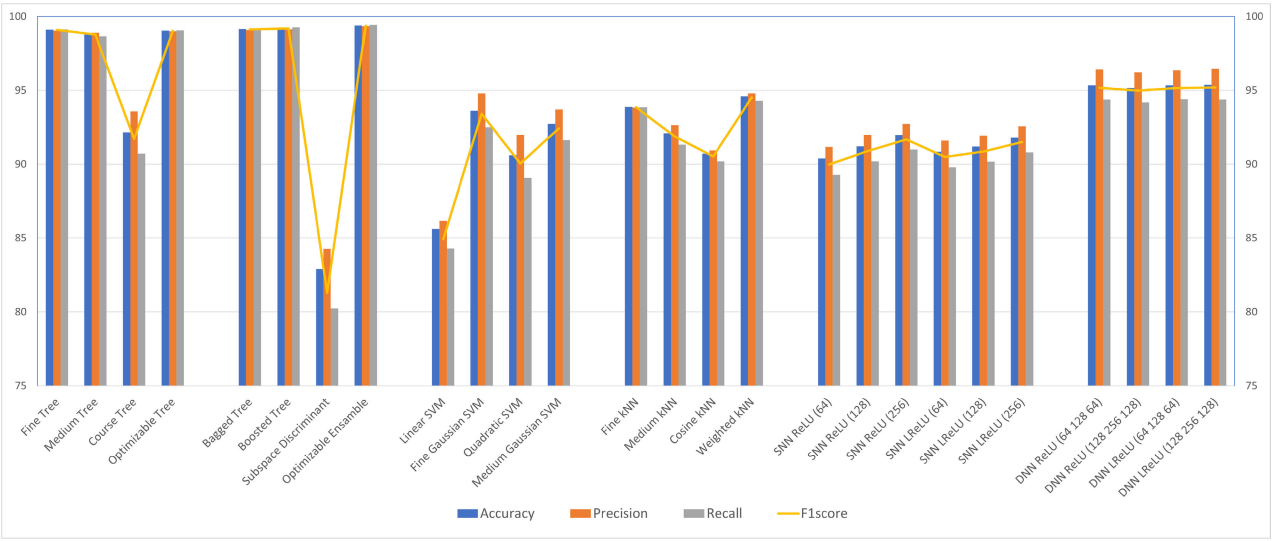


FIGURE 4. Comparison of learning methods for task classification in edge computing networks.

time frame as $n = \{1000, 2000, 4000\}$ to demonstrate edge server selection process simulation results across a broader range as illustrated in Figure 5. Furthermore, we provide user mobility where all users are uniformly distributed within [10, 100] meters in each time slot n.

In classification, different versions of learning models such as DT, EL, SVM, kNN, SNN, and DNN are used in this study. The number of leaves is primarily considered in the functions that control the depth of the trees in DT models where Fine Tree, Medium Tree, Coarse Tree has maximum 100, 20, 4 number of splits to make many fine distinctions between classes. In EL models, Bagged Tree uses Random forest bag ensamble method while Boosted Tree uses Adaboost for classification. Subspace Discriminant EL method creates boundaries between classes to find linear combinations of features. Besides, Bayesian Optimization (BO) approach is used in Optimizable Tree and Optimizable Ensamble learning models since BO is one of the most useful non-convex optimization frameworks for tuning hyperparameters and is commonly preferred in literature for proper parameter selection [41], [42]. In addition, SVM models differ depending on the kernel type and scale that linear, quadratic and Gaussian based kernel functions are utilized to separate data points of one class from other classes. Furthermore, the different versions of kNN models are formed by the number of neighbors and distance. In Fine kNN, the number of neighbors is set to 1 whereas in Medium kNN neighbors is set to 10. In Cosine kNN, Cosine distance metric is used for classification while Weighted kNN uses a distance weight, and the number of neighbors are set to 10 for both. When it comes to ANN models, SNN has a structure with a single hidden layer. The number of neurons in this hidden layer varies between 64, 128, and 256. For DNN, a three-layer ANN structure is chosen and the number of neurons of the layers is determined as {64,128,256}, {128, 256, 128}, and {64, 128, 64}. For the activation function of SNN and DNN, ReLU and LeakReLU(0.01) functions are separately used in both ANN classifiers.

The training of ML models and speed tests are carried out on a system with 16 GB RAM and a 2.6 GHz 6-core processor. To ensure fairness in experimental findings, we run the program 10 times and use average results. For each classification ($k = 5$)-fold cross-validation is used for computing the performance metrics. Additionally, a large variety of hyper-parameters are also swept to illustrate parameter space variation.

C. SIMULATION RESULTS

To validate the proposed system model and the intelligent task classification framework, extensive simulations are performed where different versions of DT, EL, SVM, kNN, SNN and DNN are used as classifiers.

Figure 4 presents a broad statistical comparison of ML models including specialized DT, kNN, SVM algorithms, and SNN and DNN algorithms in terms of accuracy, precision, recall, and Macro F1 scores. It can be concluded from the graph that most of the DT and EL methods have better performance over SVM, kNN, SNN and DNN ML models. When comparing the DT models, since the Fine Tree (FT)

TABLE 2. Speed and F1 score comparison of ML models for all types of IoT devices.

Classifier	Sample/Sec	Ratio to LODCO	F_1^{macro}
LODCO	$275,188 \pm 10,280$	$\times 1$	-
Fine Tree	$0,012 \pm 0,003$	$\times 22242$	99.08
Optimizable Ensemble	$0,016 \pm 0,007$	$\times 16450$	99.37
Weighted kNN	$0,026 \pm 0,011$	$\times 10270$	94.52
Fine Gaussian SVM	$0,090 \pm 0,004$	$\times 3054$	90.01
SNN^{ReLU}	$0,137 \pm 0,012$	$\times 1996$	91.67
DNN^{LReLU}	$0,181 \pm 0,017$	$\times 1516$	95.19

TABLE 3. Speed and F1 score comparison of ML models for UAV users.

Classifier	Sample/Sec	Ratio to LODCO	F_1^{macro}
LODCO	$254,61 \pm 9,137$	$\times 1$	-
Fine Tree	$0,015 \pm 0,003$	$\times 16739$	99.56
Optimizable Ensemble	$0,021 \pm 0,006$	$\times 11940$	96.23
Weighted kNN	$0,036 \pm 0,013$	$\times 7.049$	96.23
Fine Gaussian SVM	$0,107 \pm 0,016$	$\times 2375$	96.09
SNN^{ReLU}	$0,175 \pm 0,021$	$\times 1449$	95.47
DNN^{LReLU}	$0,263 \pm 0,037$	$\times 968$	97.46

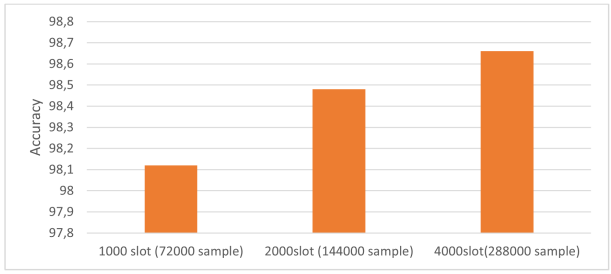


FIGURE 5. Example of edge server selection results for FT learning method.

has many leaves over the other tree models it has greater performance. In EL models, Optimizable Ensemble (OE) has the highest score due to using hyperparameter optimization methods. It can be concluded from the graph that Optimizable Ensemble (OE) and Fine Tree (FT) have a greater performance over other learners where F1 score values are 99.37% and 99.08%, respectively. SVM and kNN models, on the other hand, have inferior performance results compared to FT and OE where weighted kNN and fine Gaussian SVM are performed better in their group by 94.52% and 90.01% F1 score values correspondingly. Besides, both linear, quadratic or Gaussian based hyperplanes in SVM and Fine, Medium and Cosine types of kNN learners do not perform effectively for classification. When we compare DL methods, DNN has promising results by providing statistical values around 95% while each SNN model has moderate results around 90% in all performance metrics. In the group of SNN, 256-layered with ReLU function activated SNN gives better results in F1 scores as 91.67% while 128-256-128 layered LReLU function activated DNN has 91.67%.

In the proposed Intelligent Task Classification Framework, we consider the highest scores of the performance metrics to select the best fit for the classification over the heterogeneous IoT environment including delay-sensitive applications. It is worth noting that a close approximation to the traditional LODCO optimization model, which is assumed as 100% score, is crucial to reveal the success of the proposed framework. Therefore, we select the best performed classifiers in the group of DT, EL, SVM, kNN, SNN and DNN models which are FT, OE, weighted kNN, fine Gaussian SVM, RelU function activated 256-layered SNN and LRelU function activated 128-256-128 layered DNN.

Statistical and execution time comparison of chosen ML models for all types of smart connected IoT devices is presented in Table 2 while Table 3 only provides the results for UAV users. It can be obtained from these two tables, all ML models have impressive speed rates across to the traditional method LODCO. As shown in Table 2, LODCO algorithm completes classification in around 275.2s while FT achieves outstanding result with around 0.012s. Similarly, it can be deduced from Table 3, all ML models can provide classification results less than a second while FT is the best in speed with a time of 0.015s. When we compare the F1 score values, the OE is the first learning model and the FT is the second model to select for the classification of IoT devices. Although the rest of the models, such as kNN, SVM, SNN, and DNN, are faster, they still fall far short of the DT F1 scores. It can be concluded that our framework provides us to select the best fit classifier to achieve greater performance results and is able to adapt multi-device environment and fast to response latency sensitive applications such as in UAVs.

Furthermore, as testing process is carried out in accordance with the chosen best fit classifier, greater statistical rates of classifiers have a positive impact on the effectiveness of the edge server selection process in our framework. Figure 5 presents the results of the edge server selection process using the FT learning approach, which is utilized owing to its higher speed and F1 scores. Once the edge server execution mode is selected, maximum channel gain is used to pick the correct server. It can be stated that our proposed system achieves greater performance results in an end-to-end manner with an accuracy rate above 98% in edge server selection even in smaller and larger datasets while taking into account execution time delays.

VI. CONCLUSION

Thanks to smart connected devices and IoT applications, users and things can connect to any service at any location. From a user point of view, latency is one of the most critical parameters for using the applications effectively. The proposed intelligent task classification framework appears to be a promising method for implementing real-world IoT devices in an edge network environment and accelerating the task decision process, which is critical for latency-sensitive IoT applications in edge computing networks. Furthermore, our approach provides end to end edge server task execution mechanism where a limited number of system parameters are utilized to accomplish the results. Simulations show that the proposed approach achieves computation offloading decision to edge servers in a fast and accurate way by using ML models when it is compared to the conventional optimization method.

As part of future work, it would be interesting to specialize the environment not only for UAV type of users but also for all types of IoT devices to demonstrate more realistic deployments. Another extension would be have more users and a massively connected device environment, and apply different learning mechanisms for offloading decisions by considering both edge and center cloud environment constraints. Additionally, the mobility scenario would be enhanced and

tailored for vehicles rather than considering the random distribution of each time slot as in this paper. Furthermore, the energy gathering capabilities of the devices and their effects on the edge computing environment might be focused on as a next step.

REFERENCES

[1] Ericsson. (2022). *White Paper, a Research Outlook Toward 2030, 6G Connecting a Cyber-Physical World*. [Online]. Available: https://www.ericsson.com/4927de/assets/local/reports-papers/white-papers/6g–connecting-a-cyber-physical-world.pdf

[2] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13849–13875, Sep. 2021.

[3] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[4] C. Sun, H. Li, X. Li, J. Wen, Q. Xiong, and W. Zhou, "Convergence of recommender systems and edge computing: A comprehensive survey," *IEEE Access*, vol. 8, pp. 47118–47132, 2020.

[5] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, Oct. 2020.

[6] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[7] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, 4th Quart., 2018.

[8] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[9] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[10] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2017.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[12] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2131–2165, 4th Quart., 2021.

[13] Y. Li, F. Qi, Z. Wang, X. Yu, and S. Shao, "Distributed edge computing offloading algorithm based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 85204–85215, 2020.

[14] M. Zamzam, T. Elshabrawy, and M. Ashour, "Resource management using machine learning in mobile edge computing: A survey," in *Proc. 9th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2019, pp. 112–117.

[15] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[16] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1251–1275, 2nd Quart., 2020.

[17] B. Atan, N. Calik, S. T. Basaran, M. Basaran, and L. Durak-Ata, "Learning-based fast decision for task execution in next generation wireless networks," in *Proc. 28th Int. Conf. Telecommun. (ICT)*, Jun. 2021, pp. 1–5.

[18] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[19] N. Sprecher et al., "Harmonizing standards for edge computing—A synergized architecture leveraging ETSI ISG MEC and 3GPP specifications," ETSI, Sophia Antipolis, France, White paper no. 36, 2020.

[20] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[21] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.

[22] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, and D. Z. Du, "Reliability-aware offloading and allocation in multilevel edge computing system," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 200–211, Mar. 2021.

[23] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[24] X. Zhou, Y. Gao, C. Li, and Z. Huang, "A multiple gradient descent design for multi-task learning on edge computing: multi-objective machine learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 121–133, Jan. 2022.

[25] C. Gong, F. Lin, X. Gong, and Y. Lu, "Intelligent cooperative edge computing in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9372–9382, Apr. 2020.

[26] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-based context-aware resource allocation for edge-computing-empowered industrial IoT," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4260–4277, May 2020.

[27] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.

[28] K. Kim, Y. M. Park, and C. Seon Hong, "Machine learning based edge-assisted UAV computation offloading for data analyzing," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 117–120.

[29] H. Maleki, M. Basaran, and L. Durak-Ata, "Reinforcement learning-based decision-making for vehicular edge computing," in *Proc. 29th Signal Process. Commun. Appl. Conf. (SIU)*, Jun. 2021, pp. 1–4.

[30] J. Ren and S. Xu, "DDPG based computation offloading and resource allocation for MEC systems with energy harvesting," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC Spring)*, Apr. 2021, pp. 1–5.

[31] Z. Qin, H. Wang, Z. Wei, Y. Qu, F. Xiong, H. Dai, and T. Wu, "Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17290–17307, Dec. 2021.

[32] A. Abdi, C. Tepedelenlioglu, M. Kaveh, and G. Giannakis, "On the estimation of the K parameter for the Rice fading distribution," *IEEE Commun. Lett.*, vol. 5, no. 3, pp. 92–94, Mar. 2001.

[33] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 13, nos. 2–3, pp. 203–221, Aug. 1996.

[34] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, and X. Chen, "Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 2154–2165, Jun. 2021.

[35] C. You and R. Zhang, "3D trajectory optimization in Rician fading for UAV-enabled data harvesting," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3192–3207, Jun. 2019.

[36] G. Qu, J. Zhou, Z. Sheng, H. Yu, and Y. Ren, "Reliability-guaranteed admission control for mobile computation offloading under Nakagami fading channel," *IEEE Wireless Commun. Lett.*, vol. 10, no. 10, pp. 2195–2199, Oct. 2021.

[37] K. Sharma and X. Wang, "Toward massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 426–471, 1st Quart., 2019.

[38] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When cellular networks meet artificial intelligence," *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 175–183, Oct. 2017.

[39] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018.

[40] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, Dec. 2011.

[41] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
[42] P. I. Frazier, "A tutorial on Bayesian optimization," 2018, *arXiv:1807.02811*.

BESTE ATAN received the B.S. degree (Hons.) in electronics and communication engineering from the İzmir İnstitute of Technology, İzmir, Turkey, in 2011, and the M.S. degree in telecommunication engineering from Istanbul Technical University, İstanbul, Turkey, in 2014, where she is currently pursuing the Ph.D. degree. She is also working as a Radio Access Networks Solution Architect with Umlaut Communications GmbH, Germany. She is also a member of the Information and Communication Research Group, Istanbul Technical University. Her current research interests include 5G and beyond networks, network optimization, edge computing, resource allocation, and scheduling.

MEHMET BASARAN received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Istanbul University, Istanbul, Turkey, in 2008 and 2011, respectively, and the Ph.D. degree in electronics and communication engineering from Istanbul Technical University (ITU), Istanbul, in 2018. He was a Research Assistant at ITU. He was a Visiting Researcher at RWTH Aachen University, Aachen, Germany, in 2017, in the context of the FET EU Horizon 2020 Project. He was a Research and Development Operations Manager at the ITU Vodafone Future Laboratory, from 2018 to 2021. He has been a Research Professional with Siemens Turkey, since April 2021. His research interests include next generation communication systems, cognitive radio networks, and signal processing for wireless communications.

NURULLAH CALIK received the M.Sc. and Ph.D. degrees in electronics and communication engineering from Yildiz Technical University, Turkey, in 2013 and 2019, respectively. He worked as a Postdoctoral Researcher at the Informatics Institute, Istanbul Technical University. He is currently an Assistant Professor with the Department of Biomedical Engineering, Istanbul Medeniyet University, Turkey. His research interests include large-scale data analysis, signal and image processing, AI applications in engineering, deep learning regression, optimization, and surrogate modeling.

SEMIHA TEDIK BASARAN received the B.Sc., M.Sc., and Ph.D. degrees in electronics and communication engineering from Istanbul Technical University (ITU), Istanbul, Turkey, in 2011, 2013, and 2019, respectively. From 2012 to 2019, she was a Research Assistant at the Wireless Communications Research Group, ITU. She was a Visiting Researcher at the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany, in 2017. From 2019 to 2020, she was an Experienced Researcher at Ericsson Research Turkey. She is currently an Assistant Professor with Istanbul Technical University. Her research interests include random network coding, mobility management, and over-the-air computation. Her Ph.D. thesis has been rewarded one of the Best Ph.D. Thesis by the IEEE Turkey Section, in 2019.

GULDE AKKUZU received the B.S. degree in electronics and communication engineering from Istanbul Technical University, Istanbul, Turkey, in 2019. She is currently pursuing the M.Sc. degree in communication engineering with the Technical University of Munich. She is also working as a Digital HW Development Engineer with Rohde&Schwarz, Munich, Germany. Her current research interests include artificial intelligence, neural networks for resource-constrained devices, HW/SW design, and embedded systems.

LUTFIYE DURAK-ATA received the B.S., M.S., and Ph.D. degrees in electrical engineering from Bilkent University, Ankara, Turkey, in 1996, 1999, and 2003, respectively. She was with the Statistical Signal Processing Laboratory, Korean Advanced Institute of Science and Technology (KAIST), from 2004 to 2005, and the Department of Electronics and Communications Engineering, Yildiz Technical University, Istanbul, Turkey, from 2005 to 2015. Since 2015, she has been with the Informatics Institute, Istanbul Technical University, Istanbul, where she is currently a Full Professor. Her research interests include adaptive and statistical signal processing and communications theory.