ACCEPTED FROM OPEN CALL

In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning

Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen

Abstract

Recently, along with the rapid development of mobile communication technology, edge computing theory and techniques have been attracting more and more attention from global researchers and engineers, which can significantly bridge the capacity of cloud and requirement of devices by the network edges, and thus can accelerate content delivery and improve the quality of mobile services. In order to bring more intelligence to edge systems, compared to traditional optimization methodology, and driven by the current deep learning techniques, we propose to integrate the Deep Reinforcement Learning techniques and Federated Learning framework with mobile edge systems, for optimizing mobile edge computing, caching and communication. And thus, we design the "In-Edge AI" framework in order to intelligently utilize the collaboration among devices and edge nodes to exchange the learning parameters for a better training and inference of the models, and thus to carry out dynamic system-level optimization and application-level enhancement while reducing the unnecessary system communication load. "In-Edge AI" is evaluated and proved to have near-optimal performance but relatively low overhead of learning, while the system is cognitive and adaptive to mobile communication systems. Finally, we discuss several related challenges and opportunities for unveiling a promising upcoming future of "In-Edge AI."

Introduction

With the increasing quantity and quality of rich multimedia services over mobile networks, there has been a huge increase in the traffic and computation for mobile users and devices in recent years, which imposes a huge amount of workload on today's already-congested backbone networks and mobile networks.

Naturally, the emerging idea of Mobile Edge Computing (MEC) is proposed [1] as a novel paradigm for easing the burden of backbone networks by pushing the computation/storage resources to the proximity of the user equipments (UEs). On the other hand, MEC circumvents the long propagation delays introduced by transmitting data from mobile devices to remote cloud computing infrastructures, and hence is able to support latency-critical mobile and Internet of Things (IoT) applications. Specifically, edge nodes, that is, base stations equipped with computation/storage capability, could deal with the computation and content requests of UEs, and consequently this scheme improves the quality-of-service (QoS) of mobile network operators (MNOs) and the quality-of-experience (QoE) of UEs and relieves the load of backbone networks, and pressure of clouds (data centers) [2].

Fulfilling the requirement of QoE of UEs is not trivial even by virtue of MEC. The key and difficult point lies in that computation offloading requires wireless data transmission and might bring about the congestion of wireless channels, and hence raises the decision making or optimization problem on the whole communication and computation integrated system, that is, how to jointly allocate communication resources and computation resources of edge nodes.

Several pioneering works have been proposed and realize quite good results in their assumed settings based on convex optimization, game theory and so on [3, 4] . Nevertheless, considering the particular use cases in MEC, these optimization methods may suffer from the following issues.

Uncertain Inputs: They assume that some key information factors are given as inputs, but actually some of them are difficult to obtain due to variant wireless channels and privacy policies.

Dynamic Conditions: The dynamics of the integrated communication and computation system are not well addressed

Temporal Isolation: Most of them do not consider the long-term effect of current decisions on resource allocation except for Lyapunov optimization, viz., in a highly time-varying MEC system, most proposed optimization algorithms are optimal or close-to-optimal only for a snapshot of the system. In a word, the problem that exists in the resource allocation optimization of the MEC system is *"lack of intelligence."*

In view of the increasing complexity of mobile networks, for example, a typical 5G node is expected to have 2000 or more configurable parameters, a recent new trend is to optimize wireless communication by Artificial Intelligence (AI) techniques [4, 5], including but not limited to the application of AI for the physical layer (PHY), the data link Layer, and traffic control [6]. In particular, related studies on edge computing and caching, such as [7, 8], have shown that rein-

Digital Object Identifier:
10.1109/MNET.2019.1800286

Xiaofei Wang, Yiwen Han, and Chenyang Wang are with Tianjin University; Qiyang Zhao is with Huawei Technology; Xu Chen is with Sun Yat-sen University; Min Chen (corresponding author) is with Huazhong University of Science and Technology.

0890-8044/19/$25.00 © 2019 IEEE

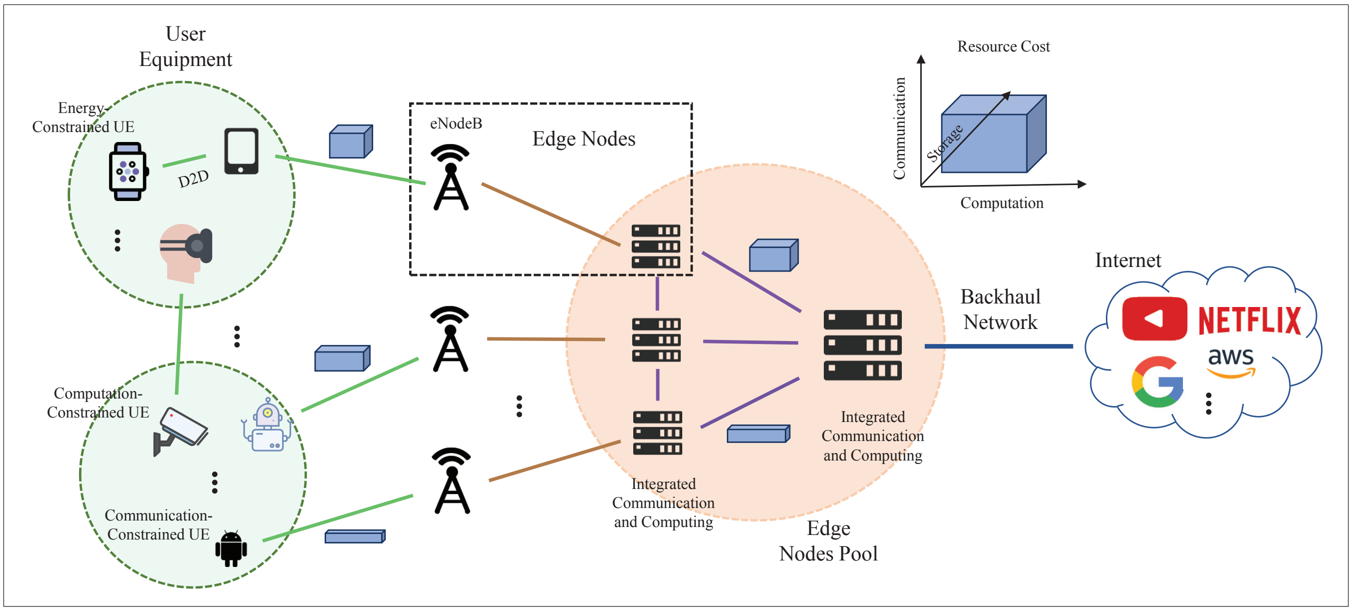

FIGURE 1. Framework of AI-supported mobile edge system with cognitive ability.

forcement learning [9] (include Deep Q-Learning [10] in Deep Reinforcement Learning) has the potential to be effective in joint resource management. But [7] is based on Q-Learning, which is not feasible for the practical MEC system where the state-action space is tremendous and the focus of [8] is on vehicular networks. Besides, none of these works has thought over:

- In what form the training data shall be gathered (whether in a distributed or a centralized way).
- Where the reinforcement learning agent should be placed and trained (whether in UEs, edge nodes or remote cloud infrastructures).
- How the update process of reinforcement learning agents should be proceeded and collaborated.
- The privacy protection of training data.

Therefore, in this article, we use Deep Reinforcement Learning (DRL) to jointly manage the communication and computation resources, and both cases of computation offloading and edge caching among the MEC system (illustrated in Fig. 1) is discussed. In addition, Federated Learning [11] is introduced as a framework for training DRL agents in a distributed manner while:

- Largely reducing the amount of data that should be uploaded via the wireless uplink channel.
- Reacting cognitively to the mobile communication environment and conditions of cellular networks.
- Adapting well with heterogeneous UEs in a practical cellular network.
- Preserving personal data privacy.

To the best of our knowledge, we are the first group to study the application of DRL coupled with Federated Learning for intelligent joint resource management of communication and computation in MEC systems. Our contribution can be summarized as follows:

- We discuss the methodology of utilizing DRL (specifically, Deep Q-Learning), and Distributed DRL for optimizing Edge Caching and Computation.
- We propose the "In-Edge AI" framework to further utilize "Federated Learning" for a better deployment of intelligent resource management in the MEC system.
- We provide proof-of-concept evaluation and verify that the proposed scheme has advantages in the balance of performance and cost.

In addition, we also discuss opportunities and challenges to hopefully unveil the upcoming future of edge architecture supporting various AI-based applications.

Optimizing the Edge by DRL

We intend to use AI techniques (particularly DRL) as the method of cognitive computing for building an intelligentizing mobile edge computing, caching and communication system. The cognitive process among protocol stacks of wireless communication is given in Fig. 2, where we partition the whole process into three main parts.

Information Collecting: Sense and collect the indispensable observing data for cognitive computing among the MEC system, including but not limited to the usage of communication and computation resources, wireless environments and intensities of UEs' request.

Cognitive Computing: By making use of the observing data of the system, cognitive computing is performed to fuse the massive observed data and further give the decision of scheduling.

Request Handling: The MEC system deals with the request of UEs on the basis of the scheduling decision given by cognitive computing.

In this section, two representative use cases in the MEC system are investigated.

DRL over the MEC System for Caching

Recently, we have observed the emergence of promising mobile content caching and delivery techniques, by which popular contents are cached in the intermediate servers (or middleboxes, gateways or routers) so that demands from users for the same content can be accommodated easily without duplicate transmissions from remote

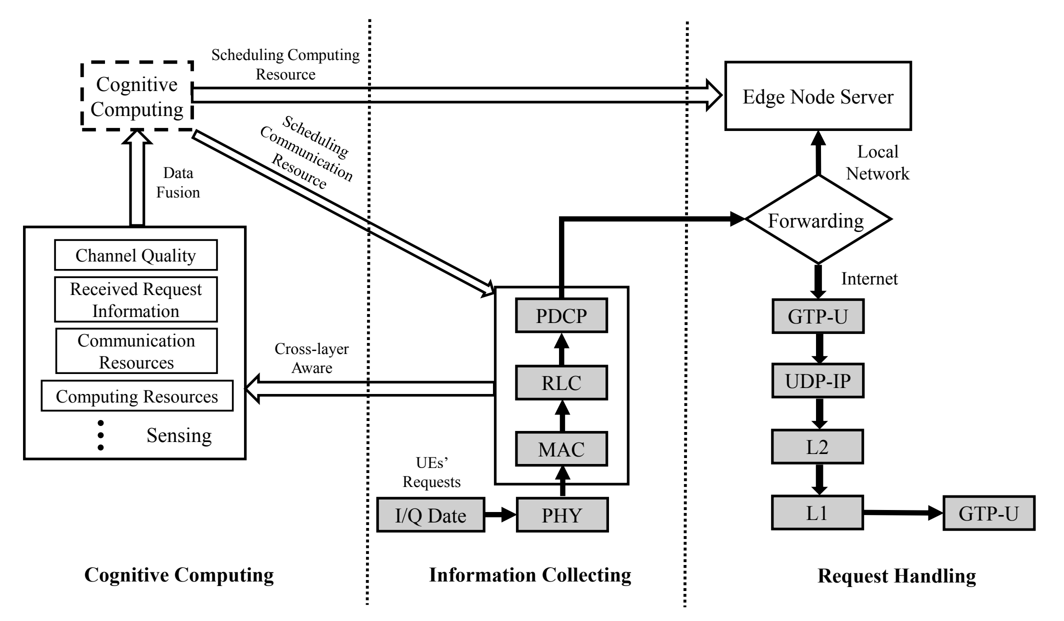

FIGURE 2. Procedure of utilizing cognitive computing in mobile edge system among protocol stacks.

cloud servers, and hence significantly reducing redundant traffic.

Thereinafter, we focus on the scenario of caching contents in edge nodes. In the MEC system depicted by Fig. 1, there is a library of F popular content files, denoted as $\mathcal{F} = \{1, ..., F\}$, that all mobile users may request in the system. The content popularity is defined as $(P_f)_{F\times 1}$, which is the probability distribution of content requests from all users. Content popularity indicates the common interests of all users in the network. In related works, the content popularity is always described by MZipf distribution. Moreover, for simply asserting the efficiency of DRL in edge caching, we assume that the content popularity changes slowly and all contents have the same size. For each request, the DRL agent in the edge node can make a decision to cache or not cache, and if yes, the agent determines which local content shall be replaced. We assume that all content popularity, user preference and average arrival rate of requests are static during a relatively long period. We model the cache replacement problem in all edge nodes as a Markov Decision Process (MDP) and use DRL to solve it, and the results will be shown later.

DRL over the MEC System for Computation Offloading

Communication Model: As illustrated in Fig. 1, considering an environment where a set of UEs $\mathcal{N} = \{1, ..., N\}$ is covered by a set of base stations $\mathcal{B} = \{1, ..., B\}$, UEs could choose to offload their intensive computation tasks to an edge node via the wireless channel or execute these tasks locally. There are M wireless channels and the set $\mathcal{M} = \{1, ..., M\}$ denotes channels of one base station. Specifically, among the decision choice $a_n \in \{0\} \cup \mathcal{M}$, UE n could choose to offload the computation to the edge via a wireless channel a_n or compute its tasks locally as a decision $a_n = 0$. For the purpose of simulating the variation of wireless channels, the channel gain state between a UE and a base station (belongs to an edge node) is independently picked from a finite state space, by which the channel state transitions are modelled as a finite-state discrete-time Markov chain. In this wireless scenario, the achievable data rate could be evaluated by the Shannon-Hartley theorem.

Computation Model: In order to demonstrate the long-term effects within the MEC system, computation tasks are generated according to Bernoulli distribution across the time horizon. A computation task is represented by (μ, ν), where μ, ν denote the size of computation input data (in bits) and the total number of CPU cycles needed to complete the computation task, respectively. All these generated tasks are stored in a task queue and executed sequentially on the UE or edge node according to a FIFO (first-in first-out) principle. When the task is executed locally, the computation execution time of it is given as $d_L = \nu/f_L$, where f_L is the the computation capability (i.e., CPU cycles per second) of the UE and determined by the amount of energy the UE decides to allocate. When the task is scheduled to be executed on the edge node, the execution time of this offloaded task can be calculated as $d_E = \nu/f_E$, where f_E is the computation capability the edge node allocated to the UE and the formula $f_E >> f_L$ holds (the computing performance of edge nodes is much stronger than the UE's).

Problem Formulation for Computation Offloading: For efficiently performing computation offloading over the MEC system, the UE shall make a joint communication and computation resource allocation decision in terms of a control action (c, e), where $c \in \{0\} \cup \{\mathcal{M}\}$ is the computation offloading decision denoting the UE chooses to execute the task locally ($c = 0$) or to offload the task via which wireless, and e denotes the amount of allocated energy for wireless communication and local computation. In the MEC system, our attention in this section is focused on how to improve the task executing experience (viz., QoE) of UEs.

High complexities of the MEC system will be introduced especially when dealing with

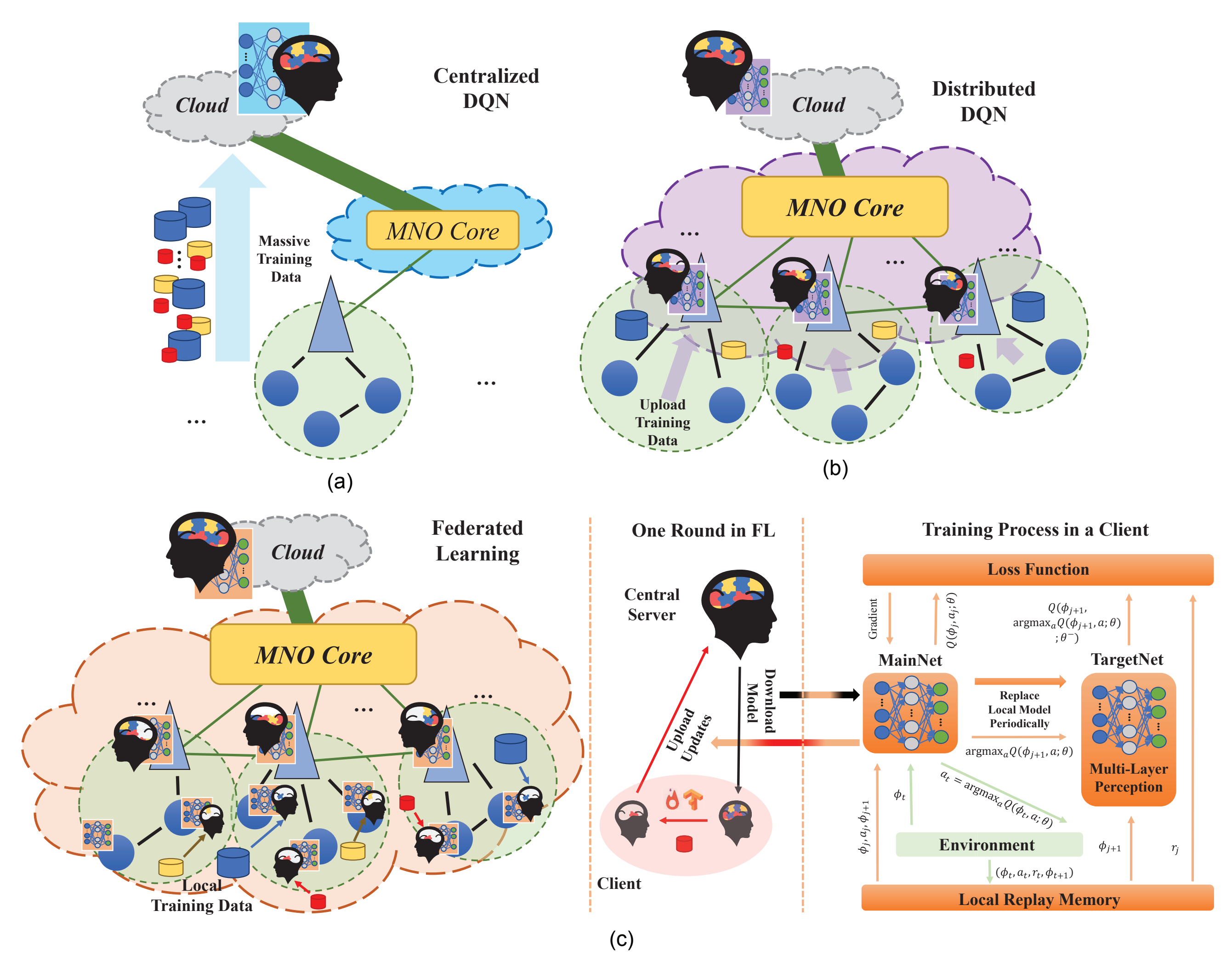

FIGURE 3. Centralized DRL v.s. Distributed DRL v.s. DRL with Federated Learning over mobile edge system: a) centralized DRL; b) distributed DRL; c) DRL with Federated Learning.

task offloading problems and the difficulty of acquiring the global necessary information when involving massive UEs. Hence, a DRL algorithm such as Deep Q-Leaning is utilized here as the agent, which handles the joint control action of the UE. The whole problem could be summarized as that the UE decides a joint wireless channel selection and energy allocation decision according to a stationary control policy $\boldsymbol{\Phi} = (\Phi_c(\boldsymbol{\Upsilon}), \Phi_e(\boldsymbol{\Upsilon}))$, while it keeps observing the network state $\boldsymbol{\Upsilon}$ which involves the task queuing state, the cumulative energy consumption of the UE, the occupying wireless channel of the UE and qualities of all wireless channels. In addition, we define an immediate utility function $u(\boldsymbol{\Upsilon}, (c, e))$ to evaluate the QoE of UEs, which is inversely proportional to the execution delay of tasks (include the wireless transmission delay and the computation delay), the task queuing delay, the energy consumption of the UE and the count of task dropping and failing. By taking advantage of Deep Q-Learning and its improved version, for example, Double DQN [12], the control policy $\boldsymbol{\Phi} = (\Phi_c(\boldsymbol{\Upsilon}), \Phi_e(\boldsymbol{\Upsilon}))$ could be trained and achieves increased utility of UEs for long-term performance optimization.

In-Edge AI with Federated Learning

Previously, two use cases of DRL in the MEC system are put forward. However, one key challenge is still pending in practice, namely the deployment of a DRL agent. Taking the computation offloading use case as an example, *if the DRL agent is trained on edge nodes or remote cloud servers just as depicted in Fig. 3a*, due to the wireless communication characters of the MEC system, three deficiencies existed:
- The training data is large in quantity when considering massive UEs, and it will increase the burden of uplink wireless channels.
- The training data which should be uploaded to edge nodes or cloud is privacy-sensitive, and it might cause potential privacy accidents.
- If training data is transformed for privacy consideration, server-side proxy data is less relevant than on-device data.

If the DRL agent is trained on the UE individually, there are still another two defects:
- The computation capability of a UE is relatively weak, and it will take a long time or is even impossible to train the DRL agent on large-scale data.

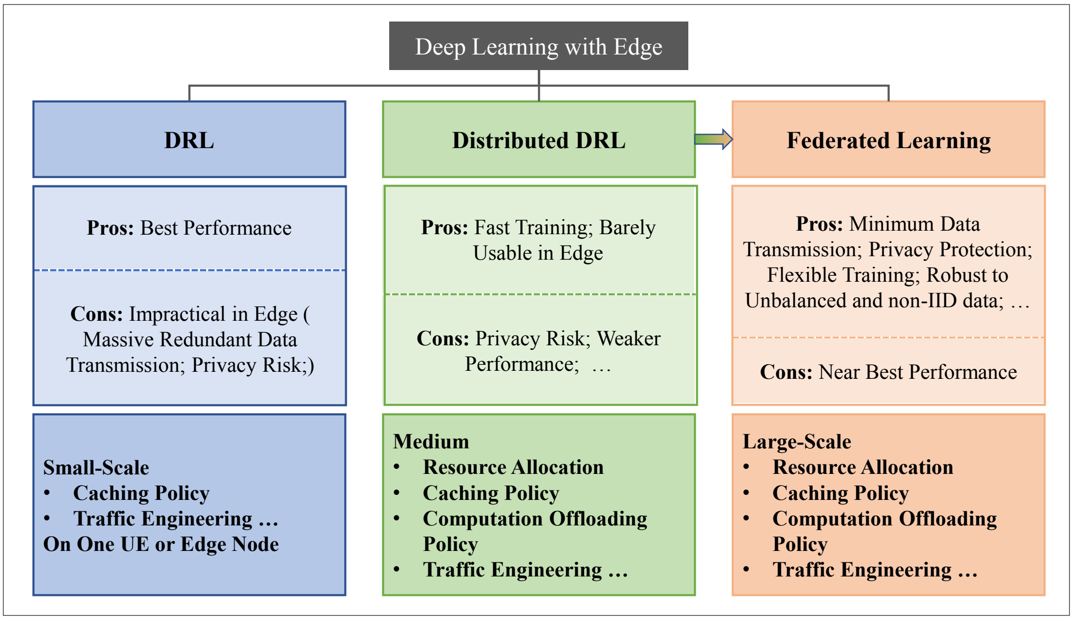
Deep Learning with Edge
DRL
Distributed DRL
Federated Learning
Pros: Best Performance
Cons: Impractical in Edge (Massive Redundant Data Transmission; Privacy Risk;)
Pros: Fast Training; Barely Usable in Edge
Cons: Privacy Risk; Weaker Performance; …
Pros: Minimum Data Transmission; Privacy Protection; Flexible Training; Robust to Unbalanced and non-IID data; …
Cons: Near Best Performance
Small-Scale
• Caching Policy
• Traffic Engineering …
On One UE or Edge Node
Medium
• Resource Allocation
• Caching Policy
• Computation Offloading Policy
• Traffic Engineering …
Large-Scale
• Resource Allocation
• Caching Policy
• Computation Offloading Policy
• Traffic Engineering …

FIGURE 4. Taxonomy of applying Deep Reinforcement Learning in mobile edge system.

- The training process of a DRL agent may introduce extra energy consumption of a UE.

DRL techniques require intensive computation capacity for finding the optimal solutions. In particular, if there are huge amounts of data factors, parameters, and criteria for optimizing resources over large-scale MEC systems (an operator's network in cities), advanced distributed Deep Learning (DL) approaches should be utilized. Hence, thinking about training DRL agents in a distributed fashion efficiently is natural. As the Fig. 4 illustrates, although keeping and training a DRL agent in every UE and edge node can achieve the best performance, it is only practical to apply distributed DRL in the MEC system, because there will not be enough time and data for training. However, most distributed DRL architectures, described by Fig. 3b, could not handle unbalanced and non-IID data and cope with the privacy issues [11]. In addition, they usually reduce the performance of DRL agents when UEs and network states are heterogeneous. Therefore, Federated Learning (FL) is introduced in this article for training DRL agents in MEC systems because it can deal with several key challenges below, which differentiates it from other distributed DL approaches [11].

Non-Independent and Identically Distributed (Non-IDD): The training data (transition memories in DRL) on the UE is based on the wireless environment it experienced and its own computation capability and energy consumption. Hence, any individual training data of a UE will not be able to represent the training data of all UEs. In FL, this challenge could be met by merging the updates of models with FedAvg in [11].

Limited communication: UEs are often unpredictably off-line or allocated with poor communication resources. Nevertheless, using additional computation could decrease the consumption of communication rounds needed to train a model. In addition, FL only asks a part of clients, in one round, to upload their updates, which handles the circumstance where clients are often unpredictably off-line.

Unbalanced: Some UEs may have more computation tasks to be handled and some may experience more states of mobile networks, resulting in varying amounts of training data among UEs. Also, this challenge could be handled by the FedAvg algorithm.

Privacy and Security: The information that needs to be uploaded for FL is the minimal update necessary to improve the DRL agent. Further, techniques of secure aggregation and differential privacy could be applied naturally, which could avoid privacy-sensitive data being contained in local updates. Nonetheless, privacy and security are not our focus in this work; more information about these issues could be found in the references of [11]

Integration of Federated Learning within Edge for In-Edge AI

Between AI in the cloud and AI in UEs, there is an "edge." With the aid of the FL framework, edge nodes equipped with AI computation in the "edge" could combine the cloud and massive UEs together and form a powerful AI entity with strong cognitive ability provided by massive UEs and edge nodes. Throughout this envisioned architecture, each edge node can support AI tasks on the system level dynamically, not only for its own but for the global optimization and balancing of the whole MEC system.

Use cases of edge caching and computation offloading are taken to represent In-Edge AI of integrating FL.

From Edge to Cloud: Considering edge caching in the MEC system, the DRL agent employed in an edge node makes decisions on caching appropriate content according to requested contents of UEs dynamically. However, space-time popularity dynamics of requesting contents put forward the requirement for collaborated edge caching. Thus, the cloud server belonging to MNOs could be the central server for coordinating the edge nodes. Among all edge nodes, each of them keeps a DRL agent and updates it by its own local training data.

From UEs to Edge: In the scenario of computation offloading, each UE shall decide whether one computation task should be offloaded to edge nodes, offloading the task via which wireless channel and the energy consumption according to the inference result of the DRL agent in it. Though UEs such as mobile phones, industrial IoT devices, and smart vehicles are able to perform some AI computation, the computation capability and the energy consumption still limit their abilities in AI computing (DRL training on large-scale data). Therefore, we proposed to use all edge nodes in the "edge" as an integral server for coordinating massive UEs covered by them. By virtue of this scheme, UEs with relatively weak computation capability could be able to hold a complex DRL agent.

The methodology of tackling two aforementioned scenarios is similar. For training a general DL model, as depicted in Fig. 3c, FL iteratively solicits a random set of clients (distributed devices that train the DL model) to download parameters of the DL model from a certain server, perform the training process on the downloaded model with their own data, and upload only the new model parameters to the server, which aggregates uploaded updates of the client to further improve the model. The process inside an individual client is also illustrated in Fig. 3c, where symbols are taken as the same version in [9] for accessible reading. Specifically, the client trains its model, which is downloaded from the central server before, based on local training data, and uploads the updated weight of MainNet backward when it is available.

To summarize, FL enables the resource-constrained edge to compute devices (include UEs and edge nodes) to learn a shared model, while keeping the training data local.

Pros and Cons of Federated Learning for In-Edge AI

The core benefit of FL lies in distributing the quality of knowledge across a large number of devices without necessarily centralizing the training data. Extended by this core benefit, several particular benefits are brought further in the MEC system.

Cognitive: Server-side proxy data is less relevant than on-device data. In the MEC system, massive UEs and edge nodes could act as perceptrons and acquire various, abundant and personalized data for updating the global DL model. On the perspective of UEs, these data could include the quality of the wireless channel, the remaining battery life and the energy consumption, the immediate computation capability and so on. Concerning edge nodes, the cognitive data could include the computation load, the storage occupation, the number of wireless communication links, the task queue states waiting for handling, and and so on. Thus, using FL based on these raw data instead of centralized DL renders the MEC system more cognitive.

Robust: FL inherently could address key issues about the availability of UEs and edge nodes and the unbalanced and non-IID data. Consequently, the performance of In-Edge AI shall not be easily affected by the unbalanced data and sometimes the poor communication environment. Further, its ability to handle non-IDD data allows massive UEs in different wireless environments to train their own DL model without being concerned about the negative effect.

To some extent, our proposed In-Edge AI is a future-oriented concept. We envision a near future where most mobile devices, particularly smartphones, are endowed with the ability of not only inferring but also training the Deep Learning model.

Flexible: In FL, additional computation could be used to decrease the number of communication rounds to train a model. An effective way to add computation is increasing computation per UE, that is, adding more local SGD updates per round. Hence, a trade-off between computation and communication existed. Specifically, powerful (both in computation and energy) UEs could decide to perform more mini-batches in training to further decrease the communication cost and vice versa.

Certainly, due to the limitation of federated optimization, FL could not outperform the centralized DL trained model, but could still achieve the near best performance, which will be demonstrated later.

Practicability Discussion

To some extent, our proposed In-Edge AI is a future-oriented concept. We envision a near future where most mobile devices, particularly smartphones, are endowed with the ability of not only inferring but also training the Deep Learning model. As is well known, even the state-of-the-art edge chips, such as Edge TPU (brought by Google and powered by TensorFlow Lite), could only support elementary training processes in DL. Therefore, the practicability should be discussed with considering both practical deployment and delay requirement.

Deploy Challenges: Learning takes a long time of training as well as inferring according to the required accuracy level. Obviously, the DRL model should not be deployed directly while setting weights of neural networks at random. Otherwise, the MEC system will be paralyzed because the DRL model could only make random decisions at preliminary exploration.

Nonetheless, this may be solved if the DL model is not trained from scratch. We are now working on using transfer learning to boost the training process in MEC systems. The basic idea is to simulate the wireless environment and requests of UEs. Just as evaluating and adjusting the antenna settings in a simulated testbed, the simulated environment is used to train an off-line Deep Reinforcement Learning agent. Then, the established DRL model could be distributed to mobile UEs.

Delay Requirement: Learning always takes a long time of training as well as inferring according to the required accuracy level. However, if using transfer learning, there will be less time or computation consumption on the side of training.

Based on the pre-established DRL, the requirement of training (not inferring) is alleviated, that is, the DRL agent could reach the satisfied accuracy level after a small number of mini-batches or even directly inferring. When the wireless environment or the pattern of UEs' requests is changed, the UE could also take advantage of the established DRL to adjust neural networks in its DRL model.

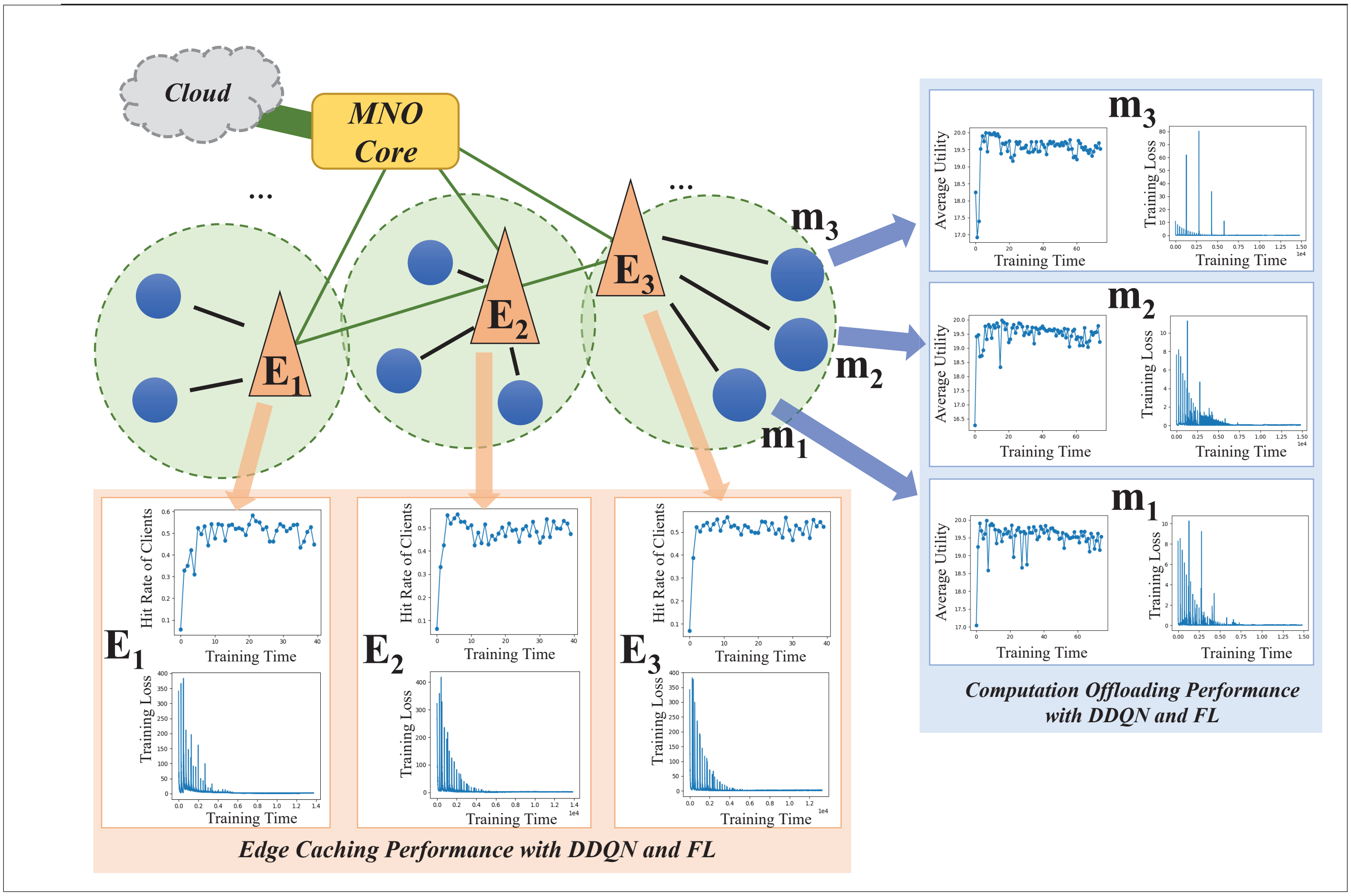

FIGURE 5. Capability of Federated Learning in edge caching and computation offloading.

Unlike using an enormous number of hidden layers and neurons in CNN or RNN, the neural network in DRL is merely a multi-layer perceptron (MLP), as shown in Fig. 3c. In our simulated scenarios, the MLP has only one hidden layer with 200 neurons. Therefore, once the model has been trained, the inferring process could be performed quickly due to the low complexity of the MainNet in DRL agents.

Our work [13] aims to meet the challenge of running DL on resource-constrained mobile devices. By incorporating this work, the requirement on computation capacity of UEs could be relaxed along with decreasing the inferring delay.

Data-Driven Evaluation of Proof-of-Concept In-Edge AI Framework

Experiment Settings

For investigating the performance of In-Edge AI with FL, two simulations on edge caching and computation offloading are presented, respectively. Among all simulations, the time horizon is discretized into time epochs.

We capture Xender's trace for one month (from 01/09/2016 to 30/09/2016), including 9,514 active mobile users, conveying 188,447 content files, and 2,107,100 content requests [14], of which the content popularity distribution can be well fitted by a Zipf distribution with ∂ = 1.58. In edge caching simulations, we use this mined Zipf distribution as the content popularity distribution to generate the content request of UEs, and consider the cooperation of six edge nodes. Once an edge node receives a request from UE, the local DRL agent in it will make a decision to cache which content or not cache, and then obtain the reward of this action for its own training. In edge caching, the cloud server belonging to MNOs is the central server for coordinating the edge nodes.

On investigating the capabilities of DRL coupled with FL over the MEC system for computation offloading, we suppose the whole bandwidth ω = 5 MHz of an edge node is divided into 10 wireless channels, and take 10 UEs as the clients in the FL framework to individually train their DRL agents and merge them among edge nodes. The channel gain states between the UE and the edge node are from a common finite set, which quantifies the quality of the wireless channel into six levels. Throughout the simulation, the number of tasks generated on each UE follows a Bernoulli distribution with average arrival rate λ_T per time epoch.

As for the DRL settings in UEs, edge nodes, and cloud, we choose the Double DQN algorithm and select tanh as the activation function and Adam optimizer. We used a single-layer fully-connected feedforward neural network, which includes 200 neurons, to serve as the target (TargetNet) and the eval (MainNet) Q network. Other parameter values in Double DQN are set as follows: replay memory capacity M = 5000, mini-batch size B = 200, discount factor γ = 0.9, exploration probability ε = 0.001, learning rate

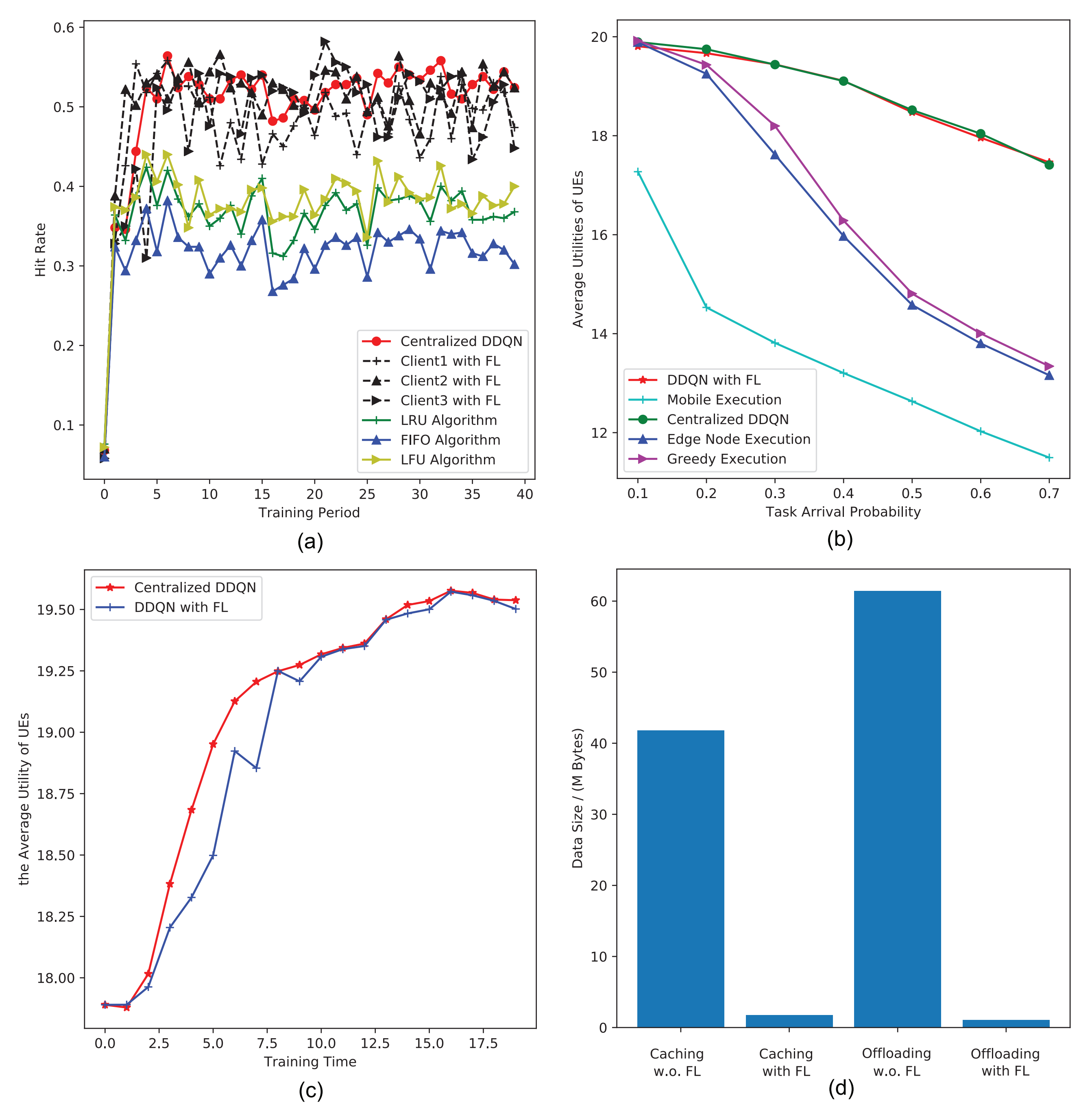

FIGURE 6. Performance Evaluation of Double DDQN with and without Federated Learning: a) performance comparison in edge caching; b) performance comparison in computation offloading; c) detailed performance comparison in computation offloading; d) comparison of transmission cost.

$\zeta = 0.005$ and the period of replacing the target Q network is $\phi = 250$. In addition, to build a baseline for the performance of the DRL agent with FL, we construct a centralized DRL agent for comparison and it is assumed to be able to receive all data used for reinforcement learning.

Evaluation Results

To elucidate the performance of our In-Edge AI framework, experiments on edge caching and computation offloading are carried out under various settings.

Figure 5 depicts the performance of DDQN with FL both on edge caching and computation offloading. Three edge nodes (E1, E2 and E3) and three UEs (m1, m2, and m3) are chosen for exhibiting the capabilities of their own DRL agents. On the perspective of edge caching, E_1, E_2, and E_3 are chosen to cache contents, and hit rates of them are improving and finally maintain within a certain range along with the process of training DRL agents. In the simulation of computation offloading, the average utilities of three UEs m_1, m_2, and m_3 are increasing with the decrease of training loss, and also maintain within a certain range at last. This means that the efficiency of handling offloaded tasks in the MEC system is improved.

Because the coordinating server in FL only executes tasks of merging updates instead of taking over the whole training, the computation load of clients is inevitably heavier because of the local training process. This will cause more energy cost on UEs or burden the computation of both UEs and edge nodes, and these issues are still open questions.

Further, Fig. 6 gives the details of performance comparison as follows.

•In Fig. 6a, the edge caching performance of the DDQN with FL is compared to the centralized DDQN, and some other caching policies, such as known LRU (least recently used), LFU (least frequently used) and FIFO (first in, first out). It can be easily seen that the specific performance of FL is near close to the results of centralized DDQN in terms of achieved hit rates of three clients (edge nodes), and outperform LRU, LFU, and FIFO.

•In the experiment on computation offloading, the performance of the DDQN with FL is compared to the centralized DDQN, and another three baseline computation offloading policies, that is, mobile execution, edge node execution, and greedy execution. Here, mobile execution, edge node execution and greedy execution mean that the UE processes all computation tasks locally, all computation tasks are offloaded from the UE to edge nodes, and the UE makes decisions on executing a computation task locally or offloading it to edge nodes with the aim of maximizing the immediate utility, respectively. In Fig. 6b, it can be observed that the DDQN with FL achieves the near performance of centralized DDQN and is superior to another three baseline policies.

•On investigating the detailed performance in the training process, we assume that the capability of wireless communication is not the hinder, that is, both massive training data for centralized DDQN and lightweight model updates for DDQN with FL can be uploaded to the target position within a discretized time epoch. As illustrated in Fig. 6c, the performance of centralized DDQN is better than that of DDQN with FL at the beginning of training. However, once the model merging of FL has been processed several times, the performance of DDQN with FL becomes near to that of centralized DDQN. Certainly, if the client wishes to obtain the expected performance using DDQN with FL, it must take time to wait for the model merging, that is, exploiting training results of other clients. Nonetheless, this experiment assumes an ideal wireless environment. In practice, massive training data are actually unable to be uploaded without any delay. Therefore, performing DDQN with FL is more practical in MEC systems, at least for now when wireless resources are also the major consideration.

•We also gather statistics of the total wireless transmission until the DRL agent is well trained both in the scenario of edge caching and computation offloading. In the framework of FL, every client only needs to upload the update of its model. Without the FL framework, viz., using centralized DRL, UEs must upload the whole training data via wireless channels and thus consume more communication resources, which is demonstrated by Fig. 6d.

With no doubt, FL must trade something for its advantages. Specifically, because the coordinating server in FL only executes tasks of merging updates instead of taking over the whole training, the computation load of clients is inevitably heavier because of the local training process. This will cause more energy cost on UEs or burden the computation of both UEs and edge nodes, and these issues are still open questions.

Opportunities and Challenges

We hereby discuss a few essential promising research directions of In-Edge AI, and highlight several pending problems from the perspective of elaborating and widening the usage of AI and Edge Computing.

Accelerating AI Tasks by Edge Communication and Computing Systems

As mentioned previously, it is important to optimize the edge of mobile communication systems by AI techniques, but it is also important to exploit specialized methods of optimizing learning computation tasks by the support of edge [13]. DRL can be treated as a type of specialized edge computing task at the service-level. How to find correct edge nodes to collaborate, and how to allocate appropriate resources, for the large amount of AI tasks with various priorities, deadlines, scales, different requirement on CPU, memory and so on are also indispensable. Interestingly, this is the reverse direction of the aforementioned Federated Learning, but also a kind of application of the "In-Edge AI." Potential research topics may include game theoretic algorithms for the competition of edge nodes and mobile users over the multi-dimensional resource for AI acceleration by the edge, and the dynamic and adaptive splitting of AI tasks becomes quite challenging.

Efficiency of In-Edge AI for Real-time Mobile Communication System toward 5G

In 5G, the defined Ultra-Reliable Low-Latency Communications (URLLC) scenarios strongly desire very small delays and high reliability of the mobile system. However, general deep learning-based optimization and prediction schemes take quite a long running time of recursions for converging to the results, which is inappropriate for mobile edge systems, particularly the system-level edge computing tasks desiring rapid responses in the scale of millisecond. In order to manage the guaranteed QoS of delays and bandwidth for caching, communication and computation, In-Edge AI should be able to provide differentiated support for various types of services, and fine-grained collaborative scheduling of the AI tasks (or split ones) over the edge nodes and mobile devices should be accelerated in nearly real-time, which is critical but remain unsolved in current literature. Furthermore, the integration of the FL framework with AI chipset hardware should be explored as well.

Incentive and Business Model of In-Edge AI

In-Edge AI is a tight confederation among mobile operators, SPs/CPs, and mobile users, while capable entities may contribute more to the overall optimization of the edge caching and computing tasks, but a large number of UEs may rely on the

AI of edge nodes and other UEs. Also, AI computation requirements from SPs and CPs should be satisfied across edge nodes of different mobile operators. Hence, a more reliable and accurate design of the incentive framework of In-Edge AI becomes interesting, which should further provide a loop of the digital copyrights of the content and services. Blockchain techniques may be integrated into the In-Edge AI framework [15], but how to distribute the huge computation load of the proof of work over the edge system and how to evaluate the contribution of In-Edge AI computation on heterogenous scenarios are still unexplored.

Conclusions

In this article, we discussed the potential of integrating the Deep Reinforcement Learning techniques and Federated Learning framework with the mobile edge system and optimizing the mobile edge computing, caching and communication with it. We performed experiments on investigating the scenarios of edge caching and computation offloading in a mobile edge system, and the "In-Edge AI" is evaluated and proved to have the ability to achieve near-optimal performance. For our future work, we will concentrate on not only optimizing learning computation tasks by the support of edge but also scheduling the AI tasks over the edge nodes and mobile devices in a fine-grained and collaborative manner.

For our future work, we will concentrate on not only optimizing learning computation tasks by the support of edge but also scheduling the AI tasks over the edge nodes and mobile devices in a fine-grained and collaborative manner.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFC0809803 and 2017YFE0123600); the Huawei HIRP OPEN Project (HO2018095224); the National Science Foundation of China (U1711265); and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (2017ZT07X355)

References

[1] ETSI, "Mobile Edge Computing — Introductory Technical White Paper," Sept. 2014.

[2] X. Wang *et al.*, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems," *IEEE Commun. Mag.*, vol. 52, no. 2, Feb. 2014, pp. 131–39.

[3] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE JSAC*, vol. 36, no. 3, Mar. 2018, pp. 587–97.

[4] Y. Mao *et al.*, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, Aug. 2017, pp. 2322–58.

[5] Q. Mao, F. Hu, and Q. Hao, "Deep Learning for Intelligent Wireless Networks: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, Fourth Quarter 2018, pp. 2595–2621.

[6] F. Tang *et al.*, "On Removing Routing Protocol from Future Wireless Networks: A Real-Time Deep Learning Approach for Intelligent Traffic Control," *IEEE Wireless Commun.*, vol. 25, no. 1, Feb. 2018, pp. 154–60.

[7] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 1, Feb. 2018, pp. 180–90.

[8] Y. He, N. Zhao, and H. Yin, "Integrated Networking, Caching, and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, Jan. 2018, pp. 44–55.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA, USA: MIT Press, 2016.

[10] V. Mnih *et al.*, "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, Feb. 2015, pp. 529–33.

[11] H. B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proc. Int'l. Conf. Artificial Intelligence and Statistics*, Apr. 2017.

[12] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," *Proc. 30th AAAI Conf. Artificial Intelligence*, vol. 16, 2016, pp. 2094–2100.

[13] E. Li, Z. Zhou, and X. Chen, "Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy," *Proc. ACM SIGCOMM*, MECOMM Workshop, 2018.

[14] X. Li *et al.*, "Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks: Modeling, Optimization, and Design," *IEEE JSAC, Special Issue on Caching for Communication Systems and Networks*, vol. 36, no. 8, Aug. 2018, pp. 1768–1785.

[15] Z. Xionget al., "When Mobile Blockchain Meets Edge Computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

Biographies

Xiaofei Wang [S'06, M'13, SM'18] (xiaofeiwang@tju.edu.cn) is currently a professor at Tianjin University, China. He received M.S. and Ph.D. degrees from the School of Computer Science and Engineering, Seoul National University, in 2008 and 2013, respectively. He received the B.S. degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology in 2005. He is the winner of the IEEE ComSoc Fred W. Ellersick Prize in 2017. His current research interests are social-aware multimedia service in cloud computing, cooperative backhaul caching and traffic offloading in mobile content-centric networks.

Yiwen Han [S'18] (hanyiwen@tju.edu.cn) is a Ph.D. student at Tianjin University, China. He received the B.S. and M.S. degrees from Nanchang University and Tianjin University, China, in 2015 and 2017, respectively. His current research interests include edge computing, wireless communication, and machine learning.

Chenyang Wang [S'17] (chenyangwang@tju.edu.cn) received his B.S. and M.S. degrees from Henan Normal University, Xinxiang, Henan province, China. He is now a Ph.D. student in the School of Computer Science and Technology at Tianjin University. His research interests include mobile edge computing, caching, offloading and D2D wireless networks.

Qiyang Zhao is currently a senior research engineer at Huawei Technologies, China. He received B.S. and Ph.D. degrees from Xidian University and the University of York in 2005 and 2013, respectively. He is specialized in end-to-end network slicing, radio resource management, control and user planes, network orchestration, optimization, reinforcement learning, transfer learning, neural network and data analytics, plus comprehensive skills in mathematical modeling, system level simulation and large-scale prototype development.

Xu Chen [M'12] (chenxu35@mail.sysu.edu.cn) received his Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012. He was a postdoctoral research associate with Arizona State University, Tempe, from 2012 to 2014, and a Humboldt Fellow with the University of Goettingen, Germany, from 2014 to 2016. He is currently a professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. He received the 2017 IEEE ICC Best Paper Award, the 2014 IEEE INFOCOM Best Paper Runner-Up Award, and the 2014 Hong Kong Young Scientist Runner-Up Award.

Min Chen [SM'09] (minchen2012@hust.edu.cn) has been a full professor in the School of Computer Science and Technology at HUST since 2012. He is the Chair of the IEEE Computer Society STC on Big Data. His Google Scholars Citations reached 17,000+ with an h-index of 64. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017, and the IEEE Jack Neubauer Memorial Award in 2019. His research focuses on IoT sensing, 5G networks, healthcare big data, and cognitive computing.