Intelligent Edge: Leveraging Deep Imitation Learning for Mobile Edge Computation Offloading

Shuai Yu, Xu Chen, Lei Yang, Di Wu, Mehdi Bennis, and Junshan Zhang

Abstract

In this work, we propose a new deep imitation learning (DIL)-driven edge-cloud computation offloading framework for MEC networks. A key objective for the framework is to minimize the offloading cost in time-varying network environments through optimal behavioral cloning. Specifically, we first introduce our computation offloading model for MEC in detail. Then we make fine-grained offloading decisions for a mobile device, and the problem is formulated as a multi-label classification problem, with local execution cost and remote network resource usage consideration. To minimize the offloading cost, we train our decision making engine by leveraging the deep imitation learning method, and further evaluate its performance through an extensive numerical study. Simulation results show that our proposal outperforms other benchmark policies in offloading accuracy and offloading cost reduction. At last, we discuss the directions and advantages of applying deep learning methods to multiple MEC research areas, including edge data analytics, dynamic resource allocation, security, and privacy, respectively.

Introduction

With the development of emerging mobile applications (e.g., augmented reality, 3D gaming, and various Internet of things [IoT] applications), more and more mobile applications become resource-thirsty and delay-sensitive. To this end, the European Telecommunications Standards Institute (ETSI) provided a concept of multi-access edge computing (MEC) in their 5G standard [1]. In the MEC architecture, distributed MEC servers are located at the network edge to provide cloud-computing capabilities and IT services with low latency, high bandwidth, and real-time processing. The edge servers can be connected to the remote cloud through backhaul links to leverage the resourceful computation capacities and IT services of the remote cloud. By the use of the collaborative edge-cloud computation offloading between mobile users and servers, mobile users' communication overhead and execution delay can be significantly reduced.

Nevertheless, mobile devices usually fail to make the most appropriate fine-grained offloading decisions in real time, especially in the time-varying and uncertain MEC environments. On one hand, the wireless and backhaul links between the mobile devices and edge-cloud servers are time-varying and uncertain. On the other hand, the MEC server offers only limited radio, storage, and computational resources, especially in hotspot areas.

To this end, a new research area, called intelligent edge learning, is emerging [2, 3], which refers to the deployment of machine learning algorithms at the network edge. One of the key motivations of pushing machine learning toward the edge is to allow rapid access to the enormous real-time data generated by mobile users for fast training and fast response to real-time offloading requirements.

Recently, deep imitation learning (DIL) [4], which is the problem of training robotic skills from human demonstration, has attracted the attention of researchers in the field of robotics (e.g., autonomous driving, gesturing, and manipulation). Compared to traditional machine-learning-based offloading methods, DIL carries four advantages:

- Better performance with large data scale
- Noteworthy accuracy in decision making
- Fast inference speed
- Easy and quick to deploy

Thus, it makes sense to deploy a novel DIL-based offloading schedule in MEC-empowered 5G networks.

In this article, we study the issue of making rapid offloading decisions for a single mobile device in MEC network environments. Our objective is to minimize the offloading cost in a time-varying network environment, subject to network resource constraints. To this end, we propose an intelligent edge computation offloading framework to make fine-grained offloading decisions for the mobile device in the MEC network. The offloading decisions made by the mobile device comprehensively consider both the execution cost on the mobile device side and time-varying network conditions (including available communication and computation resources, wired and wireless channel conditions) on the MEC side.

In summary, the contributions of this article are summarized as follows. Based on behavioral cloning [4], which performs supervised learning from the observation of demonstrations (i.e., the optimal offloading decisions in this article), we design a DIL-based offloading model for the intelligent framework. Our model is first trained from learn-

Digital Object Identifier: 10.1109/MWC.001.1900232

Shuai Yu, Xu Chen (corresponding author), and Di Wu are with Sun Yat-sen University; Lei Yang is with the University of Nevada; Mehdi Bennis is with the University of Oulu; Junshan Zhang is with Arizona State University.

ing demonstrations in an offline manner. After a quick and easy deployment, our model can make near-optimal online offloading decisions at a very fast inference speed. We discuss potential directions and advantages for applying deep learning into multiple MEC research areas.

The rest of this article is organized as follows. We first introduce the related works in the following. Then we present our computation offloading model. Next, we formulate the optimization problem and describe the DIL-based offloading model. Simulation results are then shown. We further discuss directions and advantages of deep learning for MEC. Finally, we conclude the article in the final section.

Related Work

In this section, we first survey the traditional computation offloading strategies. Then we review the state-of-the-art machine-learning-based computation offloading strategies. Last but not least, we introduce DIL The related works are summarized in Table 1.

Methods	Related works	Advantages	Disadvantages
Traditional	[5, 6]	Performance guarantee	High complexity
Reinforcement learning	[7, 8]	Model-free	Curse of dimensionality
Deep reinforcement learning	[9]	Suitable for dynamic environments	Long online training time
Deep imitation learning	Our work	Quick and easy to deploy, fast online inference speed	Requires a large number of offline demonstrations

TABLE 1. Machine-learning-based computation offloading methods.

Traditional Computation Offloading Strategies

From the perspective of a mobile user in the MEC network, it needs to decide whether and where to offload its computational tasks to enhance its quality of service (QoS). However, in practical edge network environments, the decision making problem is sophisticated because the network environments are randomly uncertain and time varying. Traditional optimization approaches (e.g., game theory [5], Lyapunov optimization [6]) for making computation offloading decisions in edge computing environments has been widely studied. For example, Chen *et al*. [5] study the computation offloading problem in multi-user MEC environments. They prove that it is NP-hard to obtain a centralized optimal solution, and propose a game theoretic approach to achieve optimal offloading decisions in a distributed manner. The authors of [6] investigate the computation offloading issue for energy harvesting (EH) devices in MEC environments. They exploit Lyapunov optimization to jointly minimize the execution latency and task failure for EH devices. The main drawback of traditional computation offloading strategies is their high algorithm complexity, especially in the multi-user multi-server edge computing environments. Thus, it is hard to deploy the strategies to practical edge network environments.

Reinforcement-Learning-Based Computation Offloading Strategies

Reinforcement learning (RL) can solve the problem of how a decision engine chooses the optimal action through interacting with outside environments. The main objective of RL is to choose an action for each state of the system in order to maximize the long-term (delayed) cost. Thus, RL is suitable for the decision making problem of computation offloading in a stochastic and dynamic edge computing network. For example, Dinh et al. [7] studied the computation offloading problem in time-varying MEC environments. They consider a multi-user multi-MEC-server environment and propose a model-free reinforcement learning (RL) offloading scheme. The objective is to make mobile users learn their long-term offloading decisions to minimize their long-term cost. The authors of [8] proposed a Markov decision process (MDP)-based dynamic offloading framework in a single-user intermittently connected cloudlet network. Through a value iteration algorithm, their decision engine can obtain an optimal policy to minimize the long-term offloading costs (i.e., computation and communication costs). The main advantage of RL is that it can learn without a priori knowledge (i.e., the model-free feature). However, with the increase of the number of system and action states, the computational complexity of RL will grow rapidly (i.e., the curse of dimensionality problem). Besides, the performance of such offloading framework heavily relies on hand-crafted features (e.g., the pre-calculated transition probability of MDP).

Recently, researchers' attention has turned to deep reinforcement learning (DRL). DRL, which combines traditional reinforcement learning and deep learning, is an emerging area of machine learning research. DRL is based on representation learning to automatically extract features from massive raw data, and can be regarded as an ideal tool to predict computation offloading decisions. For example, the authors of [9] jointly optimize networking, caching, and computing resources for a vehicular network. Due to the high complexity of the joint optimization problem, they propose a DRL method to solve the problem. The main advantage of DRL for computation offloading relates to its online training manner, which is suitable in a dynamic network environment. However, the corresponding training time is very long.

Deep Imitation Learning

Deep imitation learning is an efficient approach to teach intelligent agents skills through learning demonstrations. The authors of [4] consider a virtual reality (VR) scenario to teach a PR2 robot to learn policies from robotic manipulation demonstrations. They show that high-quality robotic manipulation demonstrations play a key role in DIL. The advantages of DIL relate to its offline training and online decision making. Thus, a trained model can be deployed easily and quickly. However, the main limitation is that the training phase of DIL heavily relies on a large number of demonstrations, and it is hard to collect the demonstrations.

In this work, we propose a DIL-based computation offloading strategy for edge computing networks. We first generate high-quality demonstrations (i.e., the optimal offloading actions) and train our model in an offline manner. Then, after a

TL allows us to deal with variational environments by leveraging the already existing labeled data of some related task or domain. In practical edge computing scenarios, we can combine DIL and TL to deal with more complex tasks (e.g., finding optimal resource allocation schemes) that are based on already trained models.

quick and easy deployment, our model can make near-optimal online offloading decisions with a very fast online inference speed.

Note that DIL is a traditional supervised learning approach, and its training and evaluation operate in the same domain. If we want to apply a trained model to a new domain, we can retrain the model, or take advantage of transfer learning (TL) [10]. Transfer learning is the ability of a system to recognize and apply knowledge and skills learned in previous domains/tasks to novel domains/tasks. TL allows us to deal with variational environments by leveraging the already existing labeled data of some related task or domain. In practical edge computing scenarios, we can combine DIL and TL to deal with more complex tasks (e.g., finding optimal resource allocation schemes) that are based on already trained models.

Computation Offloading Model

We study the computation offloading for a single mobile device in a small cell-based MEC system. Note that the small cell-based MEC system consists of:

- Mobile devices
- MEC server, also called small cell cloud-enhanced e-Node B (SCceNB)
- Remote cloud

Thus, the mobile device can:

- Execute its computational tasks locally
- Offload its tasks to the SCceNB through a wireless link
- Offload its tasks to the remote cloud through wireless and backhaul links

Application Model

We model a mobile application $\mathcal{A}$ as a weighted directed graph $\mathcal{A} = (\mathcal{T}, \mathcal{D})$, where $\mathcal{T}$ represents the sub-tasks, and $\mathcal{D}$ the data dependencies (i.e., input and output data) between the sub-tasks. Then we split the application into multiple sub-tasks by fine-grained partitioning. Note that each sub-task of the application can be offloaded and executed independently.

We adopt a parameter tuple $\langle t, \xi_t, d_{t-1,t}, d_{t,t+1}\rangle$ to characterize the mobile application $\mathcal{A}$ for the mobile device, where t is the current sub-task, $\xi_t(t \in \mathcal{T})$ represents the workload of sub-task t. $d_{t-1,t}$ and $d_{t,t+1}$ denote the size of input and output data for sub-task t, respectively. Let ρ_t (in CPU cycles per byte), denote the complexity of sub-task t. It denotes the required CPU cycles a CPU core will perform per byte for the input data processed by sub-task t. Thus, ξ_t can be given as $\xi_t = \rho_t \cdot d_{t-1,t}$. Note that ξ_t is decided by the nature (e.g., algorithm complexity) of the sub-task t.

Execution Model

The mobile device can process the mobile application $\mathcal{A}$ locally. According to the application parameter tuple, the task execution time for the mobile device to execute sub-task t locally is decided by the computation capacity of the mobile device (in million instructions per second).

For the edge execution, the mobile device can establish a cellular link with the SCceNB and offload its own sub-tasks to the SCceNB via the radio access network (RAN). Based on the assumptions above, the delay for sub-task input and output data transfer through cellular transmission is determined by the data size of data exchange between sub-tasks and the cellular data rates. In addition, the edge execution time (i.e., for the SCceNB to execute sub-task t) is determined by the total computing resource of the available CPU cores.

For the remote cloud execution, the end-to-end (E2E) latency is decided by the RAN and core network as well as the backhaul between them. In this article, we consider that the E2E delay consists of wireless and wired delays. Let $\mathcal{W}$ denote the wired delay between the SCceNB and the remote cloud. Note that the delay consists of:

- The backhaul delay between SCceNB and the core network
- The processing delay of the core network
- The communication delay for data transmission between the core network and remote cloud/Internet

Problem Formulation

Decision Making Procedure

When the mobile device receives the offloading requirement of application $\mathcal{A}$, it first sends a message on the data size $\mathcal{D}$ of the sub-tasks for the application. The report also includes the current wireless channel state (e.g., the channel quality between the mobile device and the SCceNB).

After receiving the message, the SCceNB allocates m subcarriers ($m \in \mathcal{M}$) and n CPU cores ($n \in \mathcal{N}$) to each sub-task for the mobile device, according to the entire available computation and communication resources and the received message. Thus, the current system state of computation offloading can be denoted by $S = (\mathcal{T}, \mathcal{D}, \mathcal{N}, \mathcal{M}, \mathcal{W})$, which consists of the mobile device's task profiles, network resource status, as well as the wired delay status.

According to the observed system state S, the mobile device calculates the immediate costs of local-edge-cloud executions for each sub-task, and makes action decisions of either processing the sub-task locally or offloading to the edge-cloud side for the current mobile application $\mathcal{A}$.

Computation Offloading Optimization Problem

The system state of the MEC network is given as S. Assume that the action space for computation offloading optimization is $\mathcal{I} = \{I_t \in 0, 1, 2\}, t \in \mathcal{T}$, indicating that the mobile device can execute a sub-task t locally ($I_t = 0$) or offload the sub-task to SCceNB ($I_t = 1$) or to the remote cloud server ($I_t = 2$). Under current system state S, $E(S, I_t)$ denotes the execution cost of sub-task t, which is:

- The immediate local execution cost if sub-task t is executed locally
- The immediate edge offloading costs if the sub-task is executed at the SCceNB
- The immediate cloud offloading costs if the sub-task is executed at the remote cloud server

Apparently, the edge offloading cost consists of radio and computation resource usage cost, the SCceNB computation cost (i.e., task execution time), and the data transmission cost (i.e., transmission delay) for offloading. The cloud offloading cost consists of radio and wired resource usage cost, the remote cloud server computation cost (i.e., task execution time), and the data transmission cost (i.e., transmission delay) for offloading.

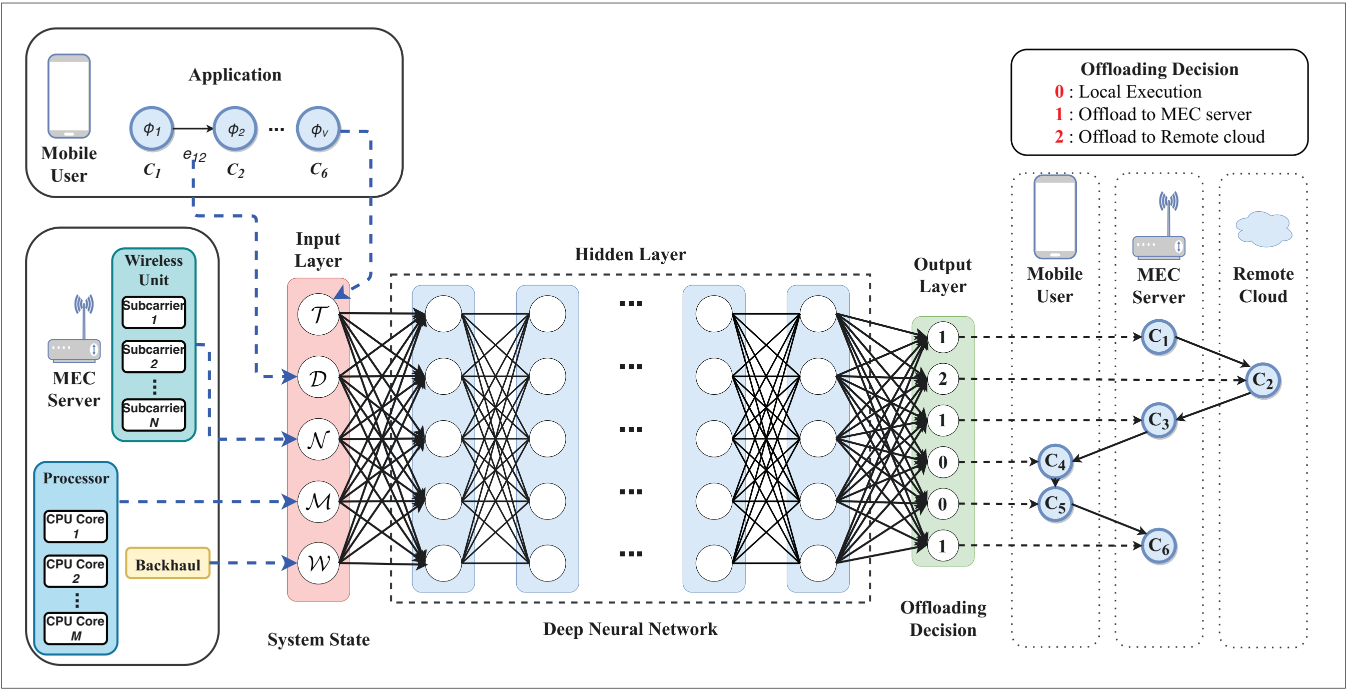

FIGURE 1. Proposed deep-imitation-learning-based offloading model.

Then the objective of the computation offloading optimization problem is to obtain a near-optimal offloading policy β* that can minimize the offloading cost given by $\Sigma_{t \in \mathcal{T}} E(S, I_t)$. Note that the offloading cost is the sum of costs for the sub-tasks of mobile application $\mathcal{A}$, which is not provided immediately. We can obtain the long-term cost when all the sub-tasks have been processed.

Deep Imitation Learning for Computation Offloading

The optimization problem of minimizing the offloading cost is a combinatorial optimization problem. Thus, it is impossible to achieve the optimal solution in real time by using standard optimization methods. Another possible approach is to utilize RL. Nevertheless, since the action space is defined over the combination of the execution selections for multiple sub-tasks, it suffers from the curse of dimensionality and hence converges very slowly in practical implementation.

To address these challenges, we explore a novel scheme of autonomous computation offloading decision by leveraging DIL. Intuitively, we first obtain the demonstrations (i.e., the optimal decision samples) by solving the computation offloading optimization problem in an offline manner. Then, using these demonstrations, we train a DIL model for imitating the optimal decision patterns and generate efficient online computation offloading decisions in real time.

Deep Multi-Label Classification Model for Computation Offloading

As shown in Fig. 1, the optimization problem can be formulated as a multi-label classification [11] problem. Assume that mobile application $\mathcal{A}$ consists of T sub-tasks. The input layer of our training model consists of the observation of the application features and network states. Our offloading decision in the output layer is a T-dimensional vector for the application. If a sub-task is offloaded, its value is 2 (cloud) or 1 (edge); otherwise, it is local. We define the multi-label offloading accuracy as the proportion of the predicted correct labels to the total number of labels. Through the accuracy, we can evaluate the output (i.e., predicted offloading actions) with respect to the optimal offloading actions.

Figure 2 illustrates the flowchart of our model. It consists of three phases: offline demonstration generation, offline model training, and online decision making. In the following, we describe these phases.

Offline Demonstration Generation: Based on behavioral cloning [4], imitation learning performs supervised learning through imitating demonstrations (i.e., optimal offloading action). Thus, the objective of this phase is to generate demonstrations to train our DIL framework. We acquire a large number of decision samples by leveraging the offline optimization scheme for solving the optimization problem. In general, when the decision space is:

- Small, we can use an exhaustive approach to obtain the optimal offloading decision by searching the whole action space (there are 3^T possibilities in the space).
- Medium, the problem can be solved by some mixed integer programming solver (e.g., CPLEX).
- Huge, we can leverage some approximate offline algorithms to obtain efficient decision samples.

Then the network state S as well as its optimal offloading decision are recorded as raw decision samples to train our framework in the next phase.

Offline Model Training: In this phase, we use the deep neural network (DNN) to extract and train the features of training data. We conventionally use the rectified linear unit (ReLU) as the activa-

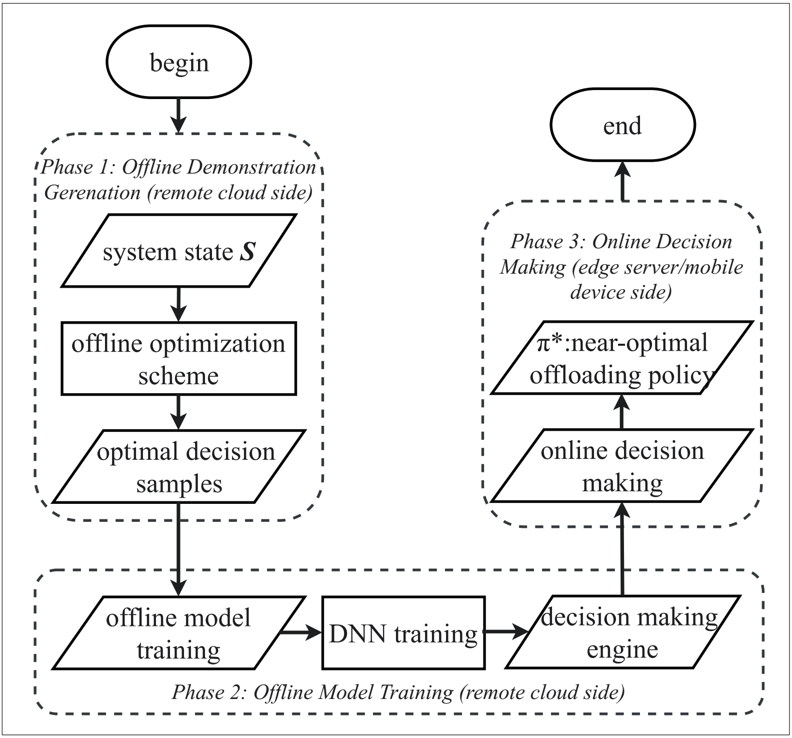
begin
Phase 1: Offline Demonstration Gerenation (remote cloud side)
system state S
offline optimization scheme
optimal decision samples
end
Phase 3: Online Decision Making (edge server/mobile device side)
π*:near-optimal offloading policy
online decision making
offline model training
DNN training
decision making engine
Phase 2: Offline Model Training (remote cloud side)

FIGURE 2. Flowchart of the proposed deep-imitation-learning-based offloading framework.

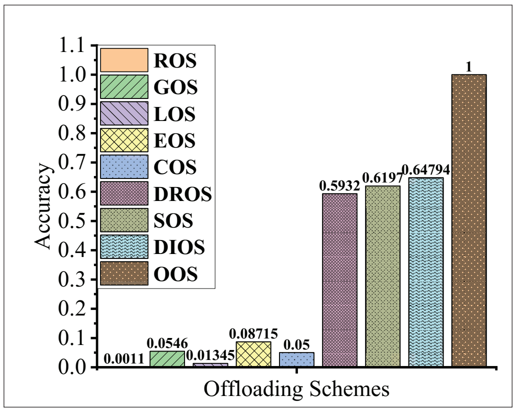

FIGURE 3. Comparison of offloading decision accuracy.

tion function for the hidden layers. Our offloading model inputs the system state S and outputs offloading decisions $I_t(t = 1, 2, ..., T)$. The sigmoid function is used as the output of our model. Note that it can be formulated as a multi-label classification problem to maximize the multi-label (i.e., predicted offloading actions) accuracy. We consider the cross-entropy loss [12] to measure the performance of the model, and use the Adam optimizer [13] to optimize the neural network. The output layer consists of T neurons that represent the offloading actions of the T sub-tasks. If an output neuron is less than 0.5, it denotes local execution; otherwise, offloading.

Online Decision Making: Once the offline model training phase of the DNN is finished, it can be used to make real-time computation offloading decisions in an online manner. At this time, the DNN outputs a sequence of offloading decisions for all sub-tasks of the mobile application. Based on the outputs, we can evaluate the offloading accuracy and offloading costs of our DIL model.

Complexity Analysis

Traditionally, using DIL to train an artificial intelligence (AI) model is computation-intensive, especially in the offline demonstration generation and offline model training phases. Fortunately, it can be done using historical data in an offline manner. Thus, we can offload the data to the resourceful remote cloud data center when the associated computational overhead is high.

In the offline demonstration generation phase, the complexity for this phase is $\mathcal{O}(|\mathcal{I}|^T)$, where $|\mathcal{I}|$ represents the size of the action space $\mathcal{I}$, and T denotes the number of sub-tasks for the mobile application. The complexity for the offline model training phase is only $\mathcal{O}(T^3Q^3)$, where Q represents the number of neurons in each hidden layer. After the offline training, our model can be deployed on either the mobile side or the edge server side, in order to make real-time offloading decisions. In the online decision making phase, our decision model has constant complexity $\mathcal{O}(1)$, which is highly scalable and real-time.

In order to alleviate the tension between resource-intensive DNNs and resource-poor edge servers, DNN compression can reduce the model complexity and resource requirement. Two typical DNN compression technologies can be used: weight pruning, which can remove redundant weights (i.e., connections between neurons) from a trained DNN, and data quantization, which can reduce the computation overhead by using a more compact format to represent layer inputs, weights, or both.

Proof-of-Concept Performance Evaluation

Simulation Setting

In order to evaluate the performance of our DIL-based offloading scheme, we consider a MEC network consisting of a mobile device and a MEC server. The number of CPU cores for the SCceNB is set to be 16 (i.e., $M = 16$). For the edge network, we consider the Rayleigh-fading environment, and the total bandwidth is divided into 256 subcarriers (i.e., $N = 256$). The wired (backhaul) delay between the SCceNB and the remote cloud is $\mathcal{W} \in [0.01, 0.02]$ s. FThe mobile application usually consists of a few sub-tasks to dozens of sub-tasks in reality. In this article, the mobile application consists of 6 sub-tasks (i.e., T = 6). The data dependencies and the workload for the sub-tasks follow uniform distribution, similar to [14]. Note that the random variables for different sub-tasks are independent.

In the offline demonstration generation phase, we use MATLAB to generate 100,000 demonstrations, which means that the mobile application is executed 100,000 times independently under various network environments. At the same time, the sample of the optimal offloading scheme can be obtained in this phase. In the online decision making phase, we evaluate the performance of our DIL-based offloading scheme (DIOS) by leveraging the Jupyter notebook. We consider the following eight benchmark schemes from the literature.

Optimal Offloading Scheme: We search the whole action space to find the optimal offloading scheme (OOS).

Local Offloading Scheme (LOS): The mobile application is executed on the mobile device locally. Thus the offloading decision variables are $I_t = 0$, ($t = 1, 2,..., T$).

Deep-Reinforcement-Learning-Based Offloading Scheme (DROS): This is a computation offloading scheme that is based on the DRL method [9].

Greedy Algorithm-Based Offloading Scheme (GOS): The mobile device chooses offloading actions through a greedy algorithm, which means that the mobile device chooses the sub-action that can maximize the offloading cost in each sub-task execution step.

Random Offloading Scheme (ROS): The offloading decisions are generated randomly.

Shallow Learning-Based Offloading Scheme (SOS): The number of hidden layers is set to be 1.

Edge Offloading Scheme (EOS): With coarse offloading strategies, the entire mobile application is offloaded to the MEC server side.

Cloud Offloading Scheme (COS): In coarse offloading strategies, the entire mobile application is offloaded to the remote cloud side.

Evaluation Results

Simulation results of our DIOS method are shown in Figs. 3–5.

Figures 3 and 4 report the offloading accuracy and corresponding offloading cost of different offloading schemes with respect to the OOS. Figure 3 shows that our DIOS outperforms other offloading schemes in offloading accuracy. At the same time, DIOS reduces the offloading cost on average by 19.80, 18.24, 23.17, 8.37, 13.61, 1.15, and 2.34 percent compared to the ROS, GOS, LOS, EOS, COS, DROS, and SOS schemes, respectively. Note that the EOS (offload computation to the edge) performs better than COS (offload computation to the remote cloud) and LOS (local execution). This proves that the MEC server can reduce energy cost on the mobile terminal side, as well as the backhaul usage on the remote cloud side.

Figure 5 shows the task execution time using different offloading schemes with respect to the OOS. Note that our DIOS reduces the execution time by 23.25, 8.77, 47.98, 17.73, 18.70, 11.36, and 15.14 percent compared to the ROS, GOS, LOS, EOS, COS DROS, and SOS schemes, respectively.

As a proof of concept, the numerical performance evaluation results above corroborate the feasibility and promise of the proposed DIL-driven computation offloading scheme. We are working on exploring other deep neural network architectures such as deep residual learning [15] for further performance gain and generalizing the approach to the challenging multi-MEC multi-user scenario.

Future Directions on Intelligent Edge Computing

In the sections above, we focus on the deep-learning-based computation offloading approach for a MEC system. In this section, we further introduce several potential directions for applying deep learning into multiple intelligent edge computing research areas, including edge data analytics, dynamic resource allocation, security, and privacy, respectively.

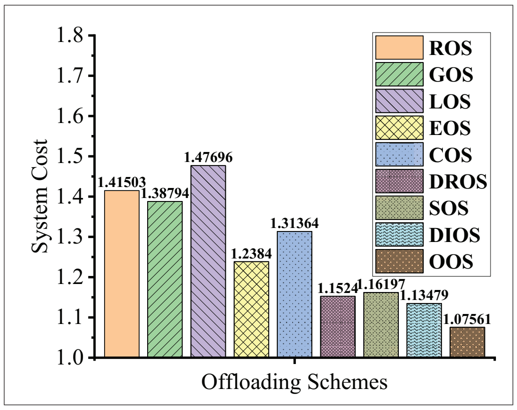

FIGURE 4. Comparison of offloading cost.

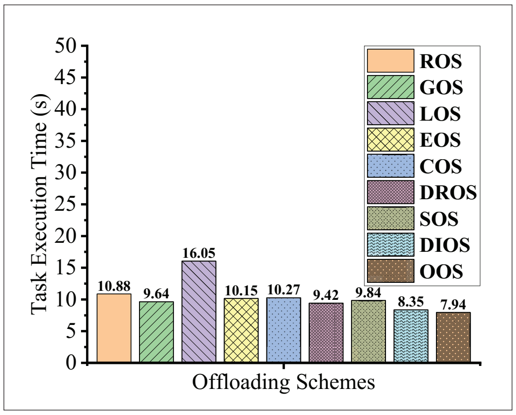

FIGURE 5. Comparison of task execution time.

Edge Data Analytics

Edge data analytics refers to the analysis of data from the distributed edge servers in a MEC system, and usually goes along with IoT applications and data caching.

IoT Application Scenario: Recently, MEC has received extensive attention in IoT scenarios, where inexpensive simple devices can generate huge volumes of raw data for big data processing. When considering the limited computation and storage resources of each single edge server, applying traditional machine learning and AI algorithms (usually compute-intensive) is inefficient. Thus, one huge problem in this scenario is how to process such big data in real time. We can apply deep learning in the MEC in order to improve the efficiency of data analyzing and processing. Deep learning can extract accurate information from the huge IoT data in such complex network environments. Compared to the traditional machine learning methods, deep learning outperforms in processing huge data, since it can precisely learn high-level features (e.g., faces and voices), extracts new features automatically for different problems, and takes much less time to infer information.

Data Caching Scenario: Data caching is one of the key features of a MEC system [1], and usually consists of content caching and computation caching. Content caching refers to caching popular contents (e.g., segments of popular movies) at the edge server in order to avoid retransmitting the same contents. This approach can significantly reduce the backhaul traffic and transmission delay, whereas computation caching denotes caching

Because deep learning can extract accurate information from the huge IoT data in such complex network environments. Compared to the traditional machine learning methods, deep learning outperforms in processing huge data, since it can precisely learn high-level features (e.g., faces and voices), extracts new features automatically for different problems, and takes much less time to inference information.

Deep learning can provide the privacy protection by transferring sensitive training data into intermediate data. Such intermediate data in DNN usually have different semantics compared to the sensitive training data. For example, after extracting the features by the DNN filter, hackers cannot obtain the original information from the hidden layer.

parts of popular computation result data (e.g., recognized face) that is likely to be reused by others. This approach can reduce not only the retransmission delay, but also the re-computation latency. We can apply the deep supervised learning (DSL) method to the edge servers to analyze and extract the features of the collected data from mobile devices. It makes more precise caching placement decisions than traditional machine learning approaches. Moreover, the popularity of different data is usually time-varying. Thus, we need to collect and process large amounts of data to obtain statistical inference from the data. Thanks to the model-free feature, we can maximize the long-term cache hit rate through DSL without knowledge of the data popularity distribution.

Dynamic Resource Allocation

Dynamic resource allocation (DRA) is a key technology to improve network performance in a dynamic environment. Note that the MEC performance is influenced by a variety of time-varying factors, including communication and computation resources, workloads of mobile users, data caching and power management policies, and so on, which is a huge project. Therefore, there is a strong demand on intelligent edge resource management to maximize long-term resource utilization. DRL has the potential to handle high-dimension state spaces of complicated control problems, and could be used to solve the DRA problem for MEC. It makes edge servers automatically and efficiently negotiate the most appropriate configuration directly from the complicated state space. Moreover, it can explore deep connections in the data and obtain accurate prediction of resource allocation schemes for MEC network.

Security and Privacy

Recently, security and privacy issues pose a tough challenge for the development of MEC. Security is becoming an increasingly important issue in MEC-based applications. Since edge servers are located at the edge and physically closer to attackers. MEC systems face multiple security threats such as wireless jamming, distributed denial of service (DoS) attacks, and smart attacks. Due to the sophistication and self-learning capability, deep learning provides more accurate and faster processing than shallow learning algorithms. It can play a key role in attack detection to deal with attacks. The privacy issue is another important threat for the cloud-based MEC system, where users risk exposing their sensitive data by sharing it and allowing edge data analytics. Moreover, MEC can provide location awareness services for cellular-network-based applications, which result in location privacy and trajectory privacy issues. Deep learning can provide privacy protection by transferring sensitive training data into intermediate data. Such intermediate data in DNN usually have different semantics compared to the sensitive training data. For example, as shown in Fig. 1, after extracting the features through the DNN filter, hackers cannot obtain the original information from the hidden layer.

Conclusion

In this article, we study the fine-grained computation offloading issues for a single mobile device within MEC networks, that is, a computation task can be executed on the mobile device locally, offloaded to an edge server, or offloaded to the remote cloud. In particular, we first introduce the application model and execution model, respectively. Then we present our offloading decision making procedure, and formulate the optimization problem to minimize the overall offloading cost. After that, we propose a deep-imitation-learning-based algorithm to obtain a near-optimal solution rapidly for the optimization problem. Numerical results confirm that our proposal achieves an offloading accuracy up to 64.79 percent and reduces at most 23.17 percent offloading cost at the same time. At last, we discuss the important directions and advantages of applying deep learning methods to multiple MEC research areas.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB1001703); the National Science Foundation of China (No. U1711265, No. 61972432); the Fundamental Research Funds for the Central Universities (No. 17lgjc40, 19LGZD37); the Program for Basic and Applied Basic Research Fund of Guangdong (No. 2019A1515010030); the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No. 2017ZT07X355); the Pearl River Talent Recruitment Program (No. 2017GC010465); the U.S. National Science Foundation (IIS-1838024); and the Guangdong Special Support Program (2017TX04X148).

References

[1] ETSI, "Multi-access Edge Computing (MEC)," 2018; https://www.etsi.org/technologies-clusters/technologies/ multi-access-edge-computing
[2] J. Park *et al.*, "Wireless Network Intelligence at the Edge," *CoRR*, vol. abs/1812.02858, 2018.
[3] E. Li, L. Zeng, Z. Zhou, and X. Chen "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing." IEEE Trans. Wireless Commun., vol. 19, no. 1, Jan. 2020, pp. 447–57.
[4] T. Zhang *et al.*, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," *Proc. 2018 IEEE Int'l. Conf. Robotics and Automation*, May 2018, pp. 1–8.
[5] X. Chen *et al.*, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Trans. Networking*, vol. 24, no. 5, Oct. 2016, pp. 2795–2808.
[6] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE JSAC*, vol. 34, no. 12, Dec. 2016, pp. 3590–3605.
[7] T. Q. Dinh *et al.*, "Learning for Computation Offloading in Mobile Edge Computing," *IEEE Trans. Commun.*, vol. 66, no. 12, Dec 2018, pp. 6353–67.
[8] Y. Zhang, D. Niyato, and P. Wang, "Offloading in Mobile Cloudlet Systems with Intermittent Connectivity," *IEEE Trans. Mobile Computing*, vol. 14, no. 12, Dec. 2015, pp. 2516–29.
[9] Y. He, N. Zhao, and H. Yin, "Integrated Networking, Caching, and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach," *IEEE Trans. Vehic. Tech.*, vol. 67, no. 1, Jan. 2018, pp. 44–55.
[10] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowledge and Data Engineering*, vol. 22, no. 10, Oct 2010, pp. 1345–59.
[11] G. Tsoumakas and I. Katakis, "Multi-Label Classification: An Overview," *Int'l. J. Data Warehousing and Mining*, vol. 2007, 2007, pp. 1–13.
[12] P. T. D. Boer *et al.*, "A Tutorial on the Cross-Entropy Method," *Annals of Operations Research*, vol. 134, 2002.
[13] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014; http://arxiv.org/abs/1412.6980.
[14] C. You *et al.*, "Asynchronous Mobile-Edge Computation Offloading: Energy-Efficient Resource Management," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, Nov 2018, pp. 7590–7605.

[15] K. He *et al.*, "Deep Residual Learning for Image Recognition," *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–78.

Biographies

Shuai Yu is currently a postdoctoral research fellow in the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. He received his Ph.D. degree from University Pierre and Marie Curie (now Sorbonne University), Paris, France, in 2018, his M.S. degree from Beijing University of Post and Telecommunications (BUPT), China, in 2014, and his B.S. degree from Nanjing University of Post and Telecommunications (NJUPT), China, in 2009. His research interests include wireless communications, mobile computing, and machine learning.

Xu Chen received his Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012. He was a postdoctoral research associate at Arizona State University, Tempe, from 2012 to 2014, and a Humboldt Scholar Fellow with the Institute of Computer Science, University of Goettingen, Germany, from 2014 to 2016. He is currently a full professor at Sun Yat-sen University, and the Vice Director of the National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He received the prestigious Humboldt Research Fellowship from the Alexander von Humboldt Foundation of Germany, the 2014 Hong Kong Young Scientist Runner-Up Award, the 2016 Thousand Talents Plan Award for Young Professionals of China, the 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, the 2017 IEEE ComSoc Young Professional Best Paper Award, the Honorable Mention Award from the 2010 IEEE International Conference on Intelligence and Security Informatics, the Best Paper Runner-Up Award from the 2014 IEEE INFOCOM, and the Best Paper Award from the 2017 IEEE ICC. He is currently an Associate Editor of the *IEEE Internet of Things Journal* and the *IEEE Journal on Selected Areas in Communications* Series on Network Softwarization and Enablers.

Lei Yang received his B.S. and M.S. degrees in electrical engineering from Southeast University, Nanjing, China, in 2005 and 2008, respectively, and his Ph.D. degree from the School of Electrical Computer and Energy Engineering, Arizona State University, Tempe, in 2012. He was a postdoctoral scholar at Princeton University, New Jersey, and an assistant research professor with the School of Electrical Computer and Energy Engineering, Arizona State University. He is currently an assistant professor with the Department of Computer Science and Engineering, University of Nevada, Reno. His research interests include big data analytics, edge computing and its applications in IoT and 5G, stochastic optimization and modeling in smart cities and cyber-physical systems, data privacy and security in crowdsensing, and optimization and control in mobile social networks. He was a recipient of the Best Paper Award Runner-Up award at IEEE INFOCOM 2014. He is currently an Associate Editor for *IEEE Access*.

Di Wu received his B.S. degree from the University of Science and Technology of China, Hefei, in 2000, his M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and his Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong in 2007. He was a postdoctoral researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a professor and the Assistant Dean of the School of Data and Computer Science at Sun Yat-sen University. His research interests include cloud/edge computing, multimedia communication, Internet measurement, and network security. He was a co-recipient of the IEEE INFOCOM 2009 Best Paper Award and the IEEE Jack Neubauer Memorial Award (2019).

Mehdi Bennis is currently an associate professor with the Centre for Wireless Communications, University of Oulu, Finland, and also an Academy of Finland Research Fellow. His main research interests are in radio resource management, heterogeneous networks, game theory, and machine learning in 5G networks and beyond. He has coauthored one book and published more than 200 research papers in international conferences, journals, and book chapters. He has been a recipient of several awards, including the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2016 Best Tutorial Prize from the IEEE Communications Society, the 2017 EURASIP Best Paper Award for the *Journal of Wireless Communications and Networks*, the all-University of Oulu award for research, and the 2019 IEEE ComSoc Radio Communications Committee Early Achievement Award. He is an Editor of *IEEE Transactions on Communications*.

Junshan Zhang received his Ph.D. degree from the School of Electrical and Computer Engineering), Purdue University, West Lafayette, Indiana, in 2000.,In 2000, he joined the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, where he has been the Fulton Chair Professor since 2015. His current research interests include information networks and data science, including communication networks, the Internet of Things (IoT), fog computing, social networks, and smart grid. He was a recipient of the ONR Young Investigator Award in 2005, the NSF CAREER Award in 2003, the IEEE Wireless Communications Technical Committee Recognition Award in 2016, the Kenneth C. Sevcik Outstanding Student Paper Award of ACM SIGMETRICS/IFIP Performance in 2016, the Best Paper Runner-Up Awards of IEEE INFOCOM 2009 and IEEE INFOCOM 2014, and the Best Paper Award at IEEE ICC 2008 and 2017. He was a TPC Co-Chair for a number of major conferences in communication networks, including IEEE INFOCOM 2012 and ACM MOBIHOC 2015. He was the General Chair of ACM/IEEE SEC 2017, WiOPT 2016, and IEEE Communication Theory Workshop 2007. He was an Associate Editor of *IEEE Transactions on Wireless Communications*, an Editor of the *Computer Networks Journal*, and an Editor of *IEEE Wireless Communications*. He currently serves as the Editor-in-Chief for *IEEE Transactions on Wireless Communications*, an Editor-at-Large for *IEEE/ACM Transactions on Networking*, and an Editor for *IEEE Network*. He was a Distinguished Lecturer of the IEEE Communications Society.