Security-Aware Data Offloading and Resource Allocation For MEC Systems: A Deep Reinforcement Learning

Ibrahim Elgendy [1], Ammar Muthanna [2], Mohammad Hammoudeh [1], Hadil Ahmed Shaiba [1], Devrim Unal [1], and Mashael Khayyat [1]

[1]Affiliation not available
[2]SPbSUT

October 30, 2023

Abstract

The Internet of Things (IoT) is permeating our daily lives where it can provide data collection tools and important measurement to inform our decisions. In addition, they are continually generating massive amounts of data and exchanging essential messages over networks for further analysis. The promise of low communication latency, security enhancement and the efficient utilization of bandwidth leads to the new shift change from Mobile Cloud Computing (MCC) towards Mobile Edge Computing (MEC). In this study, we propose an advanced deep reinforcement resource allocation and securityaware data offloading model that considers the computation and radio resources of industrial IoT devices to guarantee that shared resources between multiple users are utilized in an efficient way. This model is formulated as an optimization problem with the goal of decreasing the consumption of energy and computation delay. This type of problem is NP-hard, due to the curseof-dimensionality challenge, thus, a deep learning optimization approach is presented to find an optimal solution. Additionally, an AES-based cryptographic approach is implemented as a security layer to satisfy data security requirements. Experimental evaluation results show that the proposed model can reduce offloading overhead by up to 13.2% and 64.7% in comparison with full offloading and local execution while scaling well for large-scale devices.

Security-Aware Data Offloading and Resource Allocation For MEC Systems: A Deep Reinforcement Learning

Ibrahim A. Elgendy, Ammar Muthanna, *Member, IEEE*, Mohammad Hammoudeh, *Senior Member, IEEE*, Hadil Ahmed Shaiba, Devrim Unal, and Mashael Khayyat

***Abstract*—The Internet of Things (IoT) is permeating our daily lives where it can provide data collection tools and important measurement to inform our decisions. In addition, they are continually generating massive amounts of data and exchanging essential messages over networks for further analysis. The promise of low communication latency, security enhancement and the efficient utilization of bandwidth leads to the new shift change from Mobile Cloud Computing (MCC) towards Mobile Edge Computing (MEC). In this study, we propose an advanced deep reinforcement resource allocation and security-aware data offloading model that considers the computation and radio resources of industrial IoT devices to guarantee that shared resources between multiple users are utilized in an efficient way. This model is formulated as an optimization problem with the goal of decreasing the consumption of energy and computation delay. This type of problem is NP-hard, due to the curse-of-dimensionality challenge, thus, a deep learning optimization approach is presented to find an optimal solution. Additionally, an AES-based cryptographic approach is implemented as a security layer to satisfy data security requirements. Experimental evaluation results show that the proposed model can reduce offloading overhead by up to 13.2% and 64.7% in comparison with full offloading and local execution while scaling well for large-scale devices.**

***Index Terms*—Computation Offloading, Mobile-edge Computing, 5G, Security, Deep Reinforcement Learning.**

I. Introduction

TODAY, the Internet of Things (IoT) network technology is fully embraced into virtually every aspect of our lives. Advances in sensor and communication technologies lead to the proliferation of complex, delay- and computation-intensive industrial IoT applications that often generate and process large volumes of data [1]. Such applications include, efficient manufacture inspection, virtual/augmented reality, image recognition, Internet of Vehicles (IoV) and e-Health [2]–[4]. To alleviate the resource constrains of mobile IoT devices and meet the communication/processing delay requirement complex computations can be offloaded to more resourceful devices [5].

Cloud computing was firstly exploited as a resource-rich service for mobile devices via the Mobile Cloud Computing (MCC) paradigm. MCC provides flexible processing, storage and services capabilities while reducing battery consumption. High latency is considered one of the key challenges facing MCC, especially in real-time and delay-sensitive applications. Additionally, security poses a critical challenge that faces MCC, where applications data and services may be vulnerable to many types of attacks during various stages of data transmission and processing [6].

Mobile Edge Computing (MEC) was recently introduced as a viable and promising solution to address MCC's challenges. In MEC, the computation capabilities of the cloud are pushed to the edge of the radio access network, which is in close proximity to mobile devices, resulting in a cost-efficient and low-latency architecture [7], [8]. Application domains such as predictive maintenance of industrial machines benefit from the MEC provision to provide fast and highly localised feedback to modify a live representation of the world [9].

Numerous approaches and models for computation offloading in MEC emerged in the literature with the goal of decreasing the consumption of energy, reducing computation latency and/or allocating radio resources efficiently [10]–[14]. Obtaining an optimum offloading solution in complex and dynamic multi-user wireless MEC system is a challenging task. Additionally, the security threats encountered during data transmission have not been addressed in most offloading approaches in the literature [15]. Moreover, the lack of adequate data protection controls can quickly overshadow the advantages of the MEC paradigm. Motivated by these aforementioned considerations, we present a deep reinforcement learning model to handle performance optimization in a multi-user and multi-task MEC systems as well as a security layer for protecting data during edge server transmission. The main contributions of our paper are summarized as follows:

- A new security layer that uses the standard AES cryptosystem.

I.-A. Elgendy is with the School of Computer Science and Technology, Harbin Institute of Technology, China and with Department of Computer Science, Faculty of Computers and Information, Menoufia University, Egypt. E-mail: ibrahim.elgendy@hit.edu.cn.

A. Muthanna is with the St. Petersburg State University of Telecommunication, 193232 St. Petersburg, Russia and with Peoples' Friendship University of Russia (RUDN University), 117198 Moscow, Russia. E-mail: ammarexpress@gmail.com.

M. Hammoudeh is with the Department of Computing, Manchester Metropolitan University, Manchester M15 6BH, U.K.. E-mail: M.Hammoudeh@mmu.ac.uk.

H. A. Shaiba is with the College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. E-mail: HAShaiba@pnu.edu.sa.

D. Unal with Qatar University, College of Engineering, KINDI Center for Computing Research, Doha, Qatar. E-mail: dunal@qu.edu.qa.

M. Khayyat is with the University of Jeddah, College of Computer Science and Engineering, Department of Information systems and Technology, Jeddah, Saudi Arabia. E-mail: Mkhayyat@uj.edu.sa.

- Formulating a combination model of computation offloading, security and resource allocation as an optimization problem with the goal of decreasing the total time and energy overhead of mobile devices.
- Transforming the formulated problem into an equivalent form of reinforcement learning, in which all the possible solutions are modeled as state spaces and the movement between different states as actions. Then, a Deep-Q-Network-based algorithm has been proposed for solving this problem and obtaining the near-optimum solution in an efficient way.

The reminder of this study is organized as follows. The related works on offloading strategies are introduced in Section II. In Section III, our system model is presented and the formulation of our optimization problem is defined. Then, the Deep-Q-Network-based proposed algorithm is presented in Section IV. Section V presents the experimental evaluation and discussion. Finally, this study is concluded in Section VI and the future work directions are presented.

II. Related Work

Numerous optimization models and approaches for computation offloading in MEC environment have been proposed in the literature. Some of these models handle only multi-user single-task MEC systems, e.g. [16], whereas others deal with multi-user multi-task environments, e.g., [17]. In addition, offloading conventional methods such as Lyapunov and convex optimization techniques [18] have been used to solve these models, whereas new algorithms based on artificial intelligence and deep learning have recently emerged [11], [19]–[21]. This section will review a brief overview of the common offloading optimization models.

A. Conventional Optimization Methods

Minimizing the total consumption of energy under a latency constraint for a multi-user, single-task MEC environment is the objective of [22]. The authors formulated an optimization problem to jointly optimize the resources of computation and communication and the decisions of offloading. Further, an efficient algorithm based on separable semi-definite relaxation approach is developed for obtaining the near-optimum solution for this problem. However, this work neglects the deadline delay requirement for the computation tasks. Tuysuz *et al.* [23] proposed a novel approach for addressing the video streaming mobility based on quality of experience (QoE), which can be deployed at the MEC servers. More precisely, this method first generates a session on the basis of QoE level and collects a set of information from the user. Afterward, three core manipulations have been performed to maintain the quality of experience level for each mobile device and to balance the load between mobile users based on user locations and their mobility via handover operations.

Nur *et al.* [24] applied the caching concept with computation offloading for a multi-user system, in which the application code and their related data for the completed tasks are cached at the edge server for the next execution. To reduce the energy and delay costs, [24] considers the priority for the computation task which is calculated by task popularity, deadline, data size and computing resource. Nevertheless, the common drawback of [24] is the absence of security mechanisms to protect application's data from attacks during the transmission.

Dai *et al.* have addressed the computation offloading for multi-user environment with multi-task in [25] and [26]. Specifically, in [25], a new offloading framework of two-tier is proposed for a heterogeneous network. An optimization problem is formulated with the aim of decreasing the overall consumption of energy and MEC servers in which computation offloading, user association, allocation of transmission power and allocation of computation resource are considered. Furthermore, an algorithm is developed to find the optimum offloading decision. Whereas in [26], the authors have jointly considered the resource allocation and offloading along with mobility factors of vehicular edge computing systems. The load among vehicular edge computing servers is balanced by selecting the optimal offloading decision for the computation tasks while maximizing the system utility is the main goal. However, the main drawback in [25] and [26] is that the security and privacy of data during the offloading process are not considered.

The authors of [27] and [28] presented solutions to effectively secure applications data on MEC systems for computation offloading. Similarly, Meng *et al.* [27] presented a secure and efficient offloading framework for MCC, by regular renewing of the server key and random padding are jointly combined to protect against timing attacks. In addition, a hybrid and queuing model based on Markov chain is utilized to optimize security and performance. Whereas, Elgendy *et al.* [28], introduced a new security layer based on the AES cryptographic algorithm with a genetic algorithm to protect application data during transmission. However, management of offloading and processing in [27] are achieved via cloud data center, which results in increased delay. However, [28] only addressed a multi-user single-task environment and used a computationally prohibitive method for solving the associated offloading problem, especially for large-scale environments.

B. Deep Learning Methods

Deep learning algorithms are widely used in offloading for multi-user environments [11]. For example, an offloading scheme based on deep reinforcement learning for devices of IoT was proposed in [29] with the goal of minimizing the total system overhead. Specifically, the level of battery, the predicted amount of the consumed energy and the capacity of the channel are used in the optimal edge server for offloading the computation tasks. Then, a deep-Q-Network learning-based algorithm is proposed to decrease the dimensionality of the states space and to accelerate the learning speed. However, in [29], the application data is not protected from cyber-attacks during the transmission process.

A stochastic policy of computational offloading for a multi-user and multi-server environment was proposed in [30]. In this work, the task arrival, computation resources and the time-varying communication qualities between mobile users and

the edge server are jointly considered. The authors formulated a Markov decision process as a problem whose aims is to increase the long-term utility performance of the entire system. Then, two efficient algorithms based on double Deep Q-Network are proposed to address the curse-of-dimensionality. In [31], Dai *et al.* proposed a novel artificial intelligence empowered vehicular network architecture for IoV which can intelligently orchestrate the edge computing as well as caching resources. In addition, they jointly formulate the edge computing and caching as a Markov decision process problem and design a Deep Deterministic Policy Gradient (DDPG) algorithm to locate the computation resources in an efficient manner. However, in [31], the popular contents are shared between the vehicles at the edge caching which are vulnerable to different types of attacks.

More recently, Huang *et al.* [32] proposed a framework based on deep reinforcement learning for an online computation offloading, where the resource allocation and the offloading decision are jointly formulated as a non-convex problem. The aim is to increase the rate of computation in wireless networks. Then, a deep reinforcement learning-based online algorithm is developed for solving this problem via decomposing it into two sub-problems, namely, decision of offloading and allocation of resource. In addition, for rapid algorithm convergence, an order-preserving quantization method and an adaptive procedure are designed. Meanwhile, a multi-user with a multi-task offloading model for IoT was proposed in [33], in which the latency of service, energy consumption and success rate of task are jointly formulated to enhance the QoE-oriented computation offloading. However, the common drawback of [32], [33] is the absence of security mechanisms to protect application's data from attacks during the transmission.

It is evident from the literature review that computation offloading was investigated for multi-user environment in which conventional methods and deep learning are used to solve these problems. However, handling the security issue in a MEC system, especially a multi-user environment with a multi-task is not addressed. In this class of systems, most mobile applications send multimedia services and generate a substantial data which may be offloaded via the mobile networks. This motivates this study of jointly considering the resource allocation challenge and offloading for an environment of a multi-user and with a multi-task. In addition, we attempt to address the data security requirement during transmission to protect against various types of attacks.

III. System Model

We study a multi-user MEC system with a single wireless base station and N mobile devices, represented by a set $\mathcal{N} = \{1, 2, \ldots, N\}$, as shown in Fig. 1. In addition, an edge server is associated with the wireless base station to provide computational and storage services. Furthermore, each mobile device has a set of $\mathcal{M} = \{1, 2, \ldots, M\}$ different types of computation tasks requirements that need to be accomplished locally or will be transmitted and executed remotely through a wireless channel. In our study, a quasi-static approach is assumed in which the number of users does not change through the offloading period whereas it may vary over different periods [28].

The next subsections are present the modeling of communication, computation and security followed with more details on the formulation of our optimization problem.

A. Communication Model

The assumed environment has a set of $\mathcal{N} = \{1, 2, \ldots, N\}$ users that are connected to a single wireless base station via a wireless channel. Each mobile device has a set of $\mathcal{M} = \{1, 2, \ldots, M\}$ computationally intensive tasks that need to be completed either locally or remotely. Our aim is to reduce the system overhead in terms of communication/processing time and consumption of energy.

We refer $a_{i,j} \in \{0, 1\}$ as the offloading decision for the computation task j of user i. Specifically, $(a_{i,j} = 0)$ indicates that the mobile device i selects to execute its computation task j locally, while $(a_{i,j} = 1)$ indicates that the device i selects to transmit and execute its computation task j remotely. So, we define $A = \{a_{1,1}, a_{1,2}, \ldots, a_{N,M}\}$ as the profile decision of offloading for users.

Subsequently, in the offloading case, the data rate of uplink for the user i can be expressed as follows:

$$r_i = Blog_2(1 + \frac{p_i g_i^2}{\theta_0 B}) \tag{1}$$

where B and p_i refer the bandwidth and the power of user i transmission and g_i and θ_0 refer the gain and the density of power noise.

Consequently, the simultaneous offloading of mobile devices is limited by the following bandwidth constraints:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} a_{i,j} r_i \leq R \tag{2}$$

In this study, an Orthogonal Frequency Division Multiple access (OFDM) method is considered for addressing the transmission of multi-users at the same cell where the uplink transmission interference of intra-cellular is significantly reduced [28]. Furthermore, the consumption overhead for transmitting the result is neglected due to the small output size (result) of the computation task in comparison with the input data size [34].

B. Computation Model

This section presents the computation model for our system model that is composed of N number of mobile devices in which each device has an M number of intensive computation tasks that need to be completed. We use a tuple $\{I_{i,j}, C_{i,j}, \tau_{i,j}\}$ to represent a computation task requirement in which $I_{i,j}$, $C_{i,j}$ and $\tau_{i,j}$ denote the input size of data for each task (code and parameters), cycles of CPU needed to accomplish the task and the maximum tolerable delay for task j completion of user i. The values of $I_{i,j}$ and $C_{i,j}$ depend on the nature of the application which is obtained using a program profiler [35].

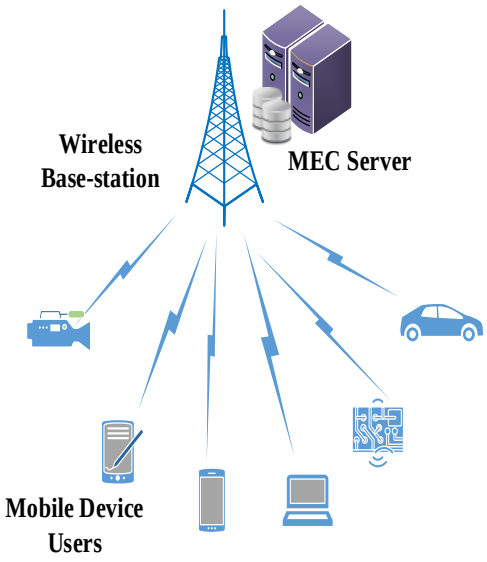

Fig. 1: An illustration of the assumed system model.

In the following subsections, the computation overhead for local and edge server computing approaches will be introduced with respect to both time of execution and consumption of energy.

1) Local Execution Approach: In local execution approach, each user i decides to execute its task j locally on its computation resources. So, the consumption of energy and time for processing the task j of user i locally can be calculated as follows:

$$T^l_{i,j} = \frac{C_{i,j}}{f^l_i} \tag{3}$$

$$E^l_{i,j} = \xi_i C_{i,j} \tag{4}$$

where f^l_i and ξ_i denote the computational capability (CPU cycles/seconds) and the CPU cycle's consumed energy of user i.

2) Edge Server Execution Approach: In the edge server execution approach, the task j of user i will be transmitted and processed remotely. Therefore, the consumption of energy and time for offloading and executing task j of user i remotely, i.e., task transmission and execution, can be calculated as follows:

$$T^e_{i,j} = \frac{I_{i,j}}{r_i} + \frac{C_{i,j}}{f^e_i} \tag{5}$$

$$E^e_{i,j} = p_i \frac{I_{i,j}}{r_i} \tag{6}$$

where f^e_i denotes the capability of computation for edge (CPU cycles/seconds) which is allocated to each user i. This study assumed that the edge server's computational resources are equally shared between all users.

C. Security Model

During offloading of computation tasks and their related data to an edge server, the offloaded data may be vulnerable to different types of attacks. In order to eliminate the data security risks, a new layer is introduced to fulfil the data security requirements. AES is used to encrypt/decrypt application data during transmission due to its efficient security and performance [36].

First, each user receives the offloading decision from the edge server which determines if the mobile user will offload their computation task or not. For the offloading decision case, the user is issued with a secret key to encrypt the transmitted data using 128-bit AES before transmitting the encrypted data to the edge server. Afterwards, the edge server uses the same key to decrypt the received data and then executes the computation task upon this data. Finally, the edge server sends the result back to the user.

We denote $\beta_i \in \{0, 1\}$ as the decision of security for user i. Specifically, $(\beta_i = 0)$ refers that the computation task's data of a user i will be offloaded without encryption. Whereas, $(\beta_i = 1)$ indicates that the computation task's data of each user i will be encrypted using our security layer before being transmitted to the edge. Therefore, we define $\beta = \{\beta_1, \beta_2, \ldots, \beta_N\}$ as a security profile. Accordingly, the extra-overhead for applying this layer could be defined as follows:

$$t^{sec}_{i,j} = \frac{\eta_{i,j}}{f^l_i} + \frac{\delta_{i,j}}{f^e_i} \tag{7}$$

$$e^{sec}_{i,j} = \xi_i \eta_{i,j} \tag{8}$$

where $\eta_{i,j}$ and $\delta_{i,j}$ refer the CPU cycles needed for encrypting and decrypting the data at user i and edge server, respectively [37], [38].

Moreover, regarding the security, computation and communication models the total consumption of time and energy for processing a tasks j of the user i can be defined as:

$$T_{i,j} = \left[(1 - a_{i,j})T^l_{i,j} + a_{i,j}T^r_{i,j}\right] \tag{9}$$

$$E_{i,j} = \left[(1 - a_{i,j})E^l_{i,j} + a_{i,j}E^r_{i,j}\right] \tag{10}$$

where $T^r_{i,j}$ and $E^r_{i,j}$ refer the total time and energy for our model with security consideration which can be expressed as follows:

$$T^r_{i,j} = \Big[\beta_i(t^{sec}_{i,j} + T^e_{i,j}) + (1-\beta_i)T^e_{i,j}\Big] \tag{11}$$

$$E^r_{i,j} = \Big[\beta_i(e^{sec}_{i,j} + E^e_{i,j}) + (1-\beta_i)E^e_{i,j}\Big] \tag{12}$$

Finally, from Eq.(9) and Eq.(10), the total time and energy overhead can be calculated as follows:

$$\kappa_{i,j} = w^t_i T_{i,j} + w^e_i E_{i,j} \tag{13}$$

where w^t_i and $w^e_i \in [0,1]$ refer to parameters for the consumption of time and energy for user i.

D. Problem Formulation

In this section, an optimization model for a multi-user environment with a multi-task is formulated with the goal of decreasing the total system overhead for users with respect to communication/processing time and energy. The formulation is given as follows:

$$\begin{aligned}
\min_{a} \quad & \Big[\sum_{i=1}^{M}\sum_{j=1}^{N}\kappa_{i,j}\Big] & \\
\text{s.t} \quad & \Big[E_{i,j} - E^l_{i,j}\Big] \le 0 & C1\\
& T_{i,j} \le \tau_{i,j} & C2\\
& \sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j} r_i \le R & C3\\
& \sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j} f^e_i \le F & C4\\
& a_{i,j} \in \{0,1\}, \qquad \forall_{i,j} & C5
\end{aligned} \tag{14}$$

The first two constraints are the energy and time limits for each computation task j. C_3 and C_4 constraints are the uplink data rate capacity and CPU computation capacity of an edge server node where F is the total CPU resources at each edge server. Finally, constraint C_5 ensures that the variable of decision offloading is binary.

Eq.(14) is considered as a linear problem where the optimal solution can be given by obtaining the offloading decision vector's values a. However, as a is considered as a binary variable, then, the set of feasible and the objective is considered as a non-convex, which makes the solving for this problem difficult, especially for a huge users' number. This is due to the problem of curse of dimensionality, in which problem size increases rapidly as the number of users increase [39]. Therefor, an deep reinforcement learning-based algorithm is proposed to obtain the near-optimum values for a.

IV. Problem Solution

A. Reinforcement Learning

Reinforcement learning is considered as a variant of machine leaning that allows a system to learn how to behave within an unknown dynamic environment and make different decisions in an optimal way without explicitly being programmed or human intervened. Fig. 2 shows a general illustration of a reinforcement learning scenario in which the agent, environment, state, action and reward are considered the main components. It is observed from the figure that, at time step t, the agent receives an observation regarding state s_t and chooses an action a_t which translates the agent from state s_t to a new state s_{t+1} on the basis of the policy $\pi = P(a_t|s_t)$. Then, the agent obtains a reward r_t and transitions to the state s_{t+1} on the basis of function of reward and transition probability of state which are defined as $R(s,a)$ and $P(s_{t+1}|s_t,a_t)$ respectively [40]. Subsequently, these steps are repeated until the agent reaches to the terminal state, where maximizing the expected cumulative rewards is the main goal which is defined as $R_t = \sum_{k=0}^{\infty}\gamma^k r_{t+k}$ with a discount factor $\gamma \in [0,1]$.

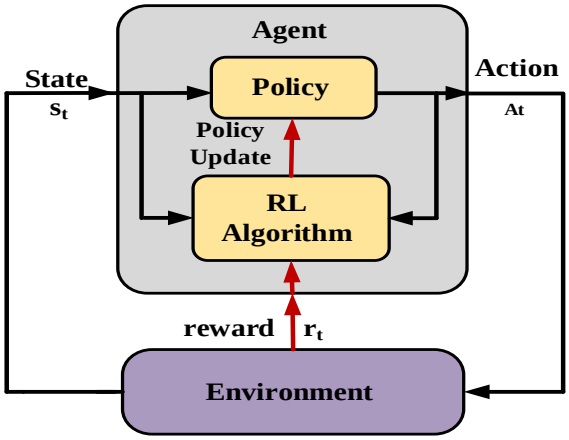

Fig. 2: Reinforcement Learning Illustration

The Q-learning algorithm is one of the most popular reinforcement learning algorithms where its learning method is defined based on recording a Q-value in the form of Q-table. This table declares the state-action pairs in which the row's headers represent the system states S, the column's headers represent the system actions A whereas the cell value represents the quality value, $Q(s,a)$, of taking an action from that state having a long-term accumulative reward. $Q(s,a)$ is calculated as:

$$Q(s',a') = Q(s,a) + \alpha\Big[r(s,a) + \gamma \max Q(s',a') - Q(s,a)\Big] \tag{15}$$

where $Q(s,a)$ and $Q(s',a')$ denote the current and the new Q values for that state and action respectively. In addition, $r(s,a)$ denotes the reward value obtained when selecting the action a at state t. $\max Q(s'a')$ denotes the maximum expected future reward obtained given the new state s' and all possible actions at that state. Finally, α and γ denote the learning rate and discount factor respectively. In this study, the computation offloading decision $a_{i,j}$ is used to represent the state $s = \{a_{i,j}\}$ while the corresponding movement among different states represent the action space A; this will be discussed with more details in the following subsection.

Regarding our optimization problem in Eq.(14), the Q-learning algorithm is not considered as effective for obtaining the optimal solution as the complexity of the problem increases rapidly as the number of users and their computation tasks

increase; this leads to an increase in the state-action pairs. Moreover, it becomes difficult to store and compute the corresponding Q value for the Q table and solving this problem becomes computationally prohibitive as the number of state-action pairs increases exponentially [39]. Therefore, Deep Q-Network (DQN) is considered to handle the Q-learning limitation through estimating the Q-value function instead of storing the Q-table as we will show in the next subsection.

B. Deep Q-Network

DQN is one of the effective reinforcement learning algorithm in which the neural network with parameter ω is used to approximate the function of Q-value and to generate the values for action as shown in Fig. 3. For DQN, the state is given as an input for the neural network and the Q-value is generated as the output, for all actions. In addition, ϵ-greedy strategy is used to select the action. A random action is selected for $\epsilon \in (0,1)$, i.e., exploration, and $a = arg\max_{a_t} Q(s(t), a(t); \omega)$ for 1-ϵ probability, i.e., exploitation.

In this study, an efficient DQN algorithm is proposed for solving our optimization problem and obtaining the near-optimum offloading decision. This problem is presented in Eq.(14). The optimization problem firstly, needs to be transformed into an equivalent reinforcement learning form, in which all the possible solutions are modeled as state spaces and the movement between different states as actions. In addition, the rewards value can be calculated based on the objective function. Consequently, the state space, actions and reward for the problem can be defined as follows:

- **State:** State space S is represented by the computation offloading decision $X = \{a_{1,1}, a_{1,2}, \ldots, a_{N,M}\}$ which is a $1 \times NM$ vector. Therefore, at an arbitrary index t, the system state can be defined as follows:

$$s(t) = \{a_{1,1}(t), a_{1,2}(t), \ldots, a_{N,M}(t)\} \tag{16}$$

- **Action:** The action space A is represented by the movement between two different states. Additionally, in this study, the system action can be defined as an index-selection within the state vector length in which the agent can move from the current state to a specific neighboring state based on the selected index. Specifically, a variable v is defined to denote the index of selection, in which $v = 1, 2, \ldots, NM$, and the action $a(t) = \{a_v(t)\}$ is considered as $1 \times NM$ vector.
- **Reward:** The agent gets a reward $R(s, a)$, at each step t, on the basis of a state s and after executing an action a which is considered as a scalar feedback signal for indicating how well the agent is doing. While the system state $s(t)$ represents the computation offloading decision, the objective function in our problem, $Z(t)$, can be derived based on the state $s(t)$ and can be denoted as follows:

$$Z_{s(t)}(t) = (\{a_{i,j}(t)\}) \tag{17}$$

where $\{a_{i,j}(t)\}$ is given by the state $s(t)$ according to the definition in Eq.(16). Additionally, based on the values of $Z_{s(t)}(t)$ and $Z_{s(t+1)}(t+1)$, the reward of the state-action pair $(s(t), a(t))$ is defined as follows:

$$r_{s(t),a(t),s(t+1)} = \begin{cases} 1, & Z_{s(t)}(t) > Z_{s(t+1)}(t+1) \\ \text{-1} & Z_{s(t)}(t) < Z_{s(t+1)}(t+1) \\ 0 & Z_{s(t)}(t) = Z_{s(t+1)}(t+1) \end{cases} \tag{18}$$

In this study, a pre-classification step has been applied on the state space in which the computation tasks that do not satisfy the completion time deadline constraints, i.e., $T^l_{i,j} <= \tau_{i,j}$), must be forced to execute locally on the mobile device, i.e, $a_{i,j} = 0$.

As shown in Fig. 3 and Algorithm (1), the DQN can be used to solve our optimization problem in Eq.(14). Firstly, given state, action and reward, the evaluation and target Q-network are initialized with random numbers ω and ω', respectively. Also, the replay memory Y is initialized with a capacity L. Then, for each episode k, an initial state s_{init} is chosen. Afterward, for each time step t and based on the ϵ strategy, the evaluation network generates a random action $a(t)$ for $\epsilon \in (0,1)$ probability and $a = arg\max_{a_t} Q^{pre}(s(t), a(t); \omega)$ for 1-ϵ probability. Then, on the basis of Eq.(18), the reward $r(t)$ as well as the next state $s(t+1)$ are obtained. In addition, the transition $(s(t), a(t), r(t), s(t+1))$ is stored in the experience replay Y. Consequently, for updating the evaluation network, a sample random minibatch of transitions $(s(k), a(k), r(k), s(k+1))$ is selected from experience replay Y and the predicted and labeled Q values, Q^{pre} and Q^{lab}, are calculated respectively as $Q(s(t), a(t); \omega)$, $r(t) + \gamma \max'_a Q^{tar}(s(t+1), a'(t); \omega')$ using evaluation and target networks shown in Procedure 1. This study is adopted as a loss function of a neural network which can calculate the loss between predicted and labeled Q values. In addition, Gradient Decent Algorithm (GDC) [41] is used to minimize this value. Finally, the parameter ω' of target network is updated every C steps.

V. Experimental Evaluation and Analysis

This section firstly introduces the setup of experimental. Afterward, an extensive discussion on the simulation results is presented to critically assess our proposed model's performance.

A. Experiment Setup

Our simulation is undertaken using a personal computer, which has an Intel® CPU 3.4 GHz Core(TM) i7-4770 with 16 GB RAM capacity. Python for development. The software environment is TensorFlow and Numpy with preinstalled Python 3.6 on Windows 10 Professional 64-bit [42]. A Multi-user environment with a multi-task is considered in which we have five users. The system bandwidth, noise and transmission power are set to 20 MHz, -100dBm and 100 mW, respectively. Each mobile user has a face recognition application as an example which consists of three independent computation tasks, namely, face detection, pre-processing and feature extraction and classification. The data size is distributed uniformly in $(0, 10)$MB, while the cycles of CPU

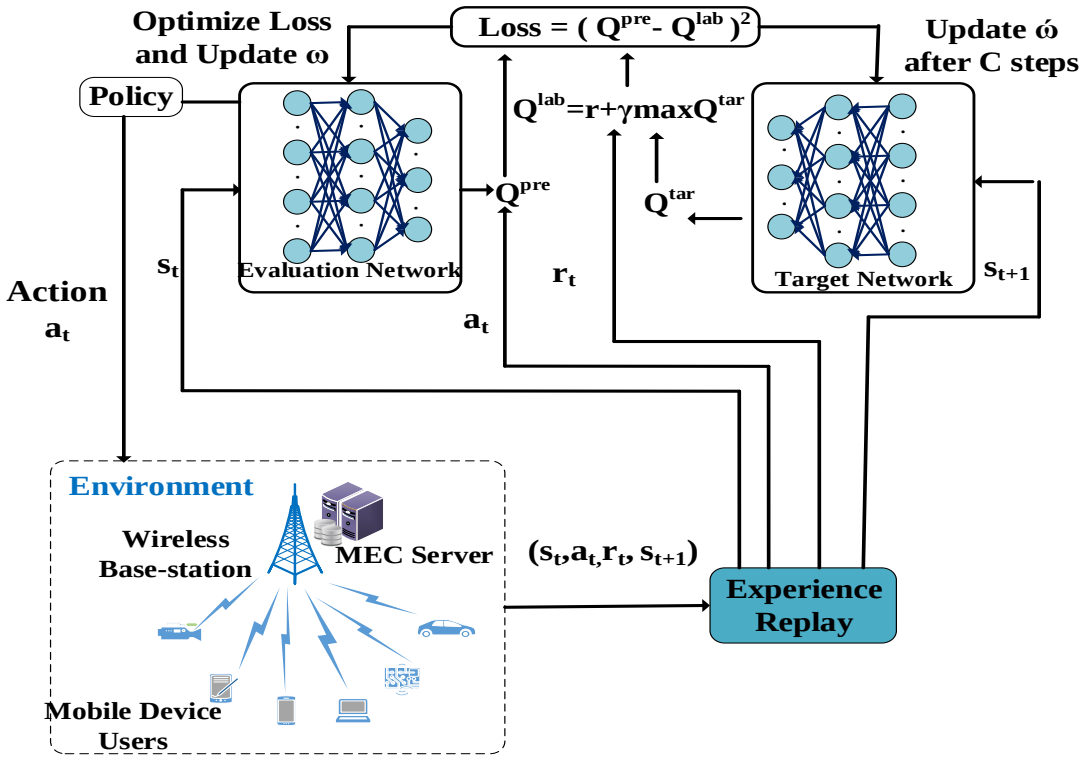

Fig. 3: Deep Q-network (DQN) based MEC System

are set to 1000 cycles/bit. The user's capability is assigned randomly within the $\{0.5, 0.6, \ldots, 1.0\}$GHz set, while edge server' CPU computational capability is set to $100GHz$. We also assume that the channel bandwidth, the transmission power of each device and background noise are $20MHz$, $100mW$ and $-100dBm$ respectively. The energy consumption for each mobile device is uniformly distributed within $(0, 20$ x $10^{-11})$J/cycle [34]. For the DQN algorithm, the episode, size mini-batch and replay memory are set to 20000, 32 and 512. While, the discount factor, learning rate, and $\epsilon-$ greedy values are set to 0.99, 0.01 and 0.1 respectively.

Finally, to verify the performance of our algorithm, five different policies are introduced:

- **Unsecure DQN**: Our model is applied without security layer addition.
- **Secure DQN**: Our model is applied after adding the security layer.
- **Local Execution**: All the computation tasks will be processed locally.
- **Full Offloading**: All the computation tasks will be processed remotely.
- **Random Offloading**: A random set of computation tasks are selected to be processed remotely while the remaining tasks will be executed locally.

B. Experiment Results

1) Convergence Performance: This subsection studies the convergence performance of the proposed algorithm, in which different values of each parameter are tested and the proper value will be selected for the next simulation.

Fig. 4 demonstrates the convergence performance of the total cost over different values of learning rate, in which the leaning rate can be used to adapt the updating speed of ω. Figure shows that, with the 0.01 value, the process of convergence is becoming faster than 0.001 value and this speed increases the value of learning rate increases. However, with the large value of learning rate i.e., 0.1, the convergence process can not converge well in which it will be fallen into a local optimum solution. Therefore, it is important to choose the appropriate learning rate value suitable for specific situations. Regarding this, we set 0.01 as a learning rate value, which is the most appropriate value.

Fig. 5 depicts the effects of different memory sizes on the convergence performance. Through the figure, we shows that with the smaller value of memory size, the convergence is becoming faster, but a local optimum solution is obtained instead of global one. Therefore, in the following simulations, the size of the replay memory is set to 1024 which is the most appropriate value.

Fig. 6 demonstrates the convergence performance of the proposed algorithm over different values of batch size in which the batch size can be utilized to determine the experience samples' number which are extracted from the memory at each training interval. From the figures, the batch size is set to 32 in the next simulations.

2) System Performance: This subsection presents and discusses the simulation results of our proposed model. First,

Algorithm 1 DQN based Computation Offloading Algorithm

Initialize the evaluation and target Q network parameters with random weights ω and ω', respectively. ($\omega' = \omega$)
Initialize replay memory Y with capacity L
for each episode $k \leq 1, 2, \ldots, K$ **do**
  Choose an initial state s^{init}
  for each step t **do**
    Generate a random number $\varphi \in [0, 1]$
    if $\varphi < \epsilon$ **then**
      Randomly select an action $a(t)$. ▷*Exploration*
    else
      Set $a(t) = arg\max_a Q^{pre}(s(t), a(t); \omega)$ where Q is estimated. ▷*Exploitation*
    end if
    Execute the action $a(t)$ and Calculate $Z_{s(t)}(t)$ according to Eq. (18)
    if $Z_{s(t)}(t) > Z_{s(t+1)}(t+1)$ **then**
      Set $r(t) = 1$
    else if $Z_{s(t)}(t) < Z_{s(t+1)}(t+1)$ **then**
      Set $r(t) =$ -1
    else
      Set $r(t) = 0$
    end if
    Save transition $(s(t), a(t), r(t), s(t+1))$ in Y
    Execute **Procedure 1** for updating the evalution network
    Reset $\omega' = \omega$ after each C steps.
  end for
end for

Procedure 1 Evaluation Network Updating

Extract a sample random mini-batch of transitions $(s(k), a(k), r(k), s(k+1))$ from Y.
if $s(k+1)$ is terminal state **then**
  Set $Q^{lab} = r(k)$.
else
  Calculate the label Q-value Q^{lab}:
  $Q^{lab} = r(k) + \gamma \max_{a(k+1)} Q^{tar}(s(k+1), a(k+1); \omega')$.
end if
Optimize the parameter ω using gradient descent algorithm which minimize the loss:
$(Q^{lab} - Q^{pre}(s(k), a(k); \omega))^2$.

the overhead of processing the computation tasks under the defined five policies over the different value of users is seen in Fig. 7. It is demonstrated from the figure that with 3 users, the our proposed DQN algorithm's overhead with and without security addition is equal to the full offloading policy and less than the other two policies. In addition, with the increasing the users' number, our model with and without security addition is able to achieve a lower overhead relative to full offloading policy. This is because the communication channels are shared and overloaded thereby lead to increase the communication time with users' number increasing. Moreover, our model can optimally select which computation tasks should be offloaded and which should not while minimizing the total cost of users

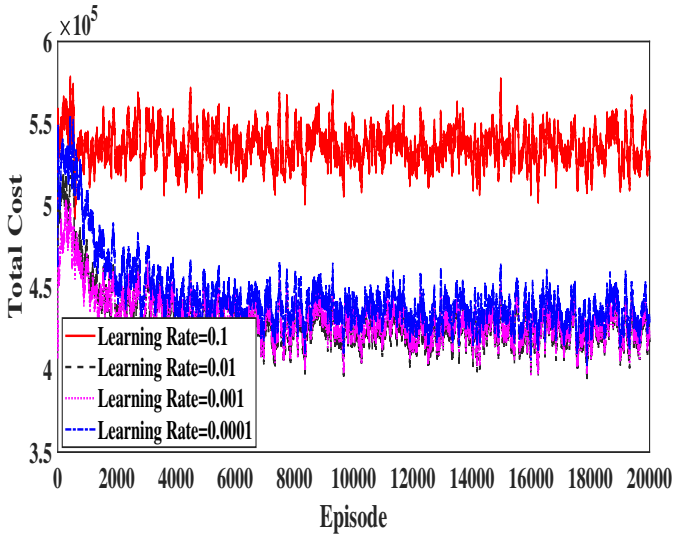

Fig. 4: Convergence performance versus different values of learning rate

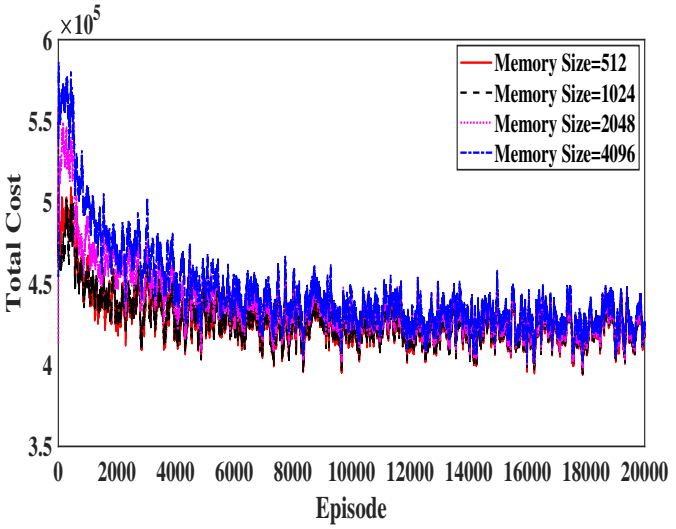

Fig. 5: Convergence performance versus different values of memory size

via the deployment of task offloading and security.

Similarly, Fig. 8 illustrates the total cost of executing the computation tasks under five different policies versus different data size for each task. As seen in this figure, the total cost of the five policies increases with the increasing size of input data for each task. Additionally, our DQN algorithm with and without a security layer outperforms the other policies. Moreover, the full offloading policy curve increases much more rapidly than the other four polices with the increasing size of input data for each task. This is because of as the size of data that is transmitted increases, the communication time also increases which leads to a significant increase in the total cost of the entire system.

Finally, Fig. 9 shows the total overhead of processing the computation tasks for different MEC server's capacity. It is seen in this figure that the policy of local execution is not

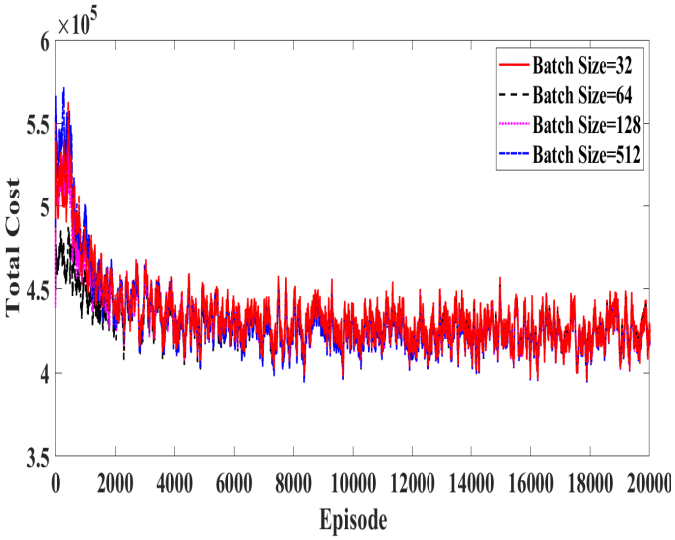

Fig. 6: Convergence performance versus different values of batch size

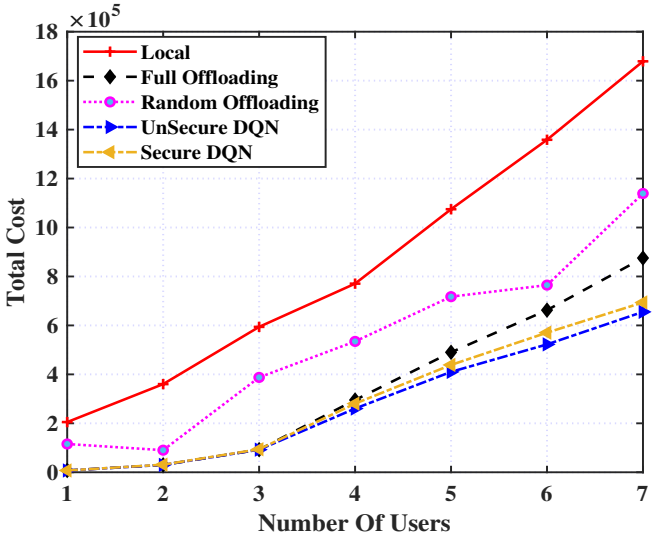

Fig. 7: The total Cost over different number of mobile users

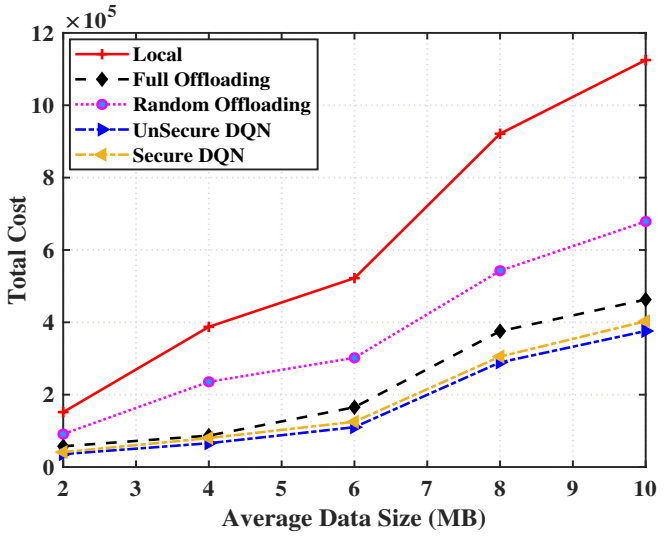

Fig. 8: Total Cost over different data size

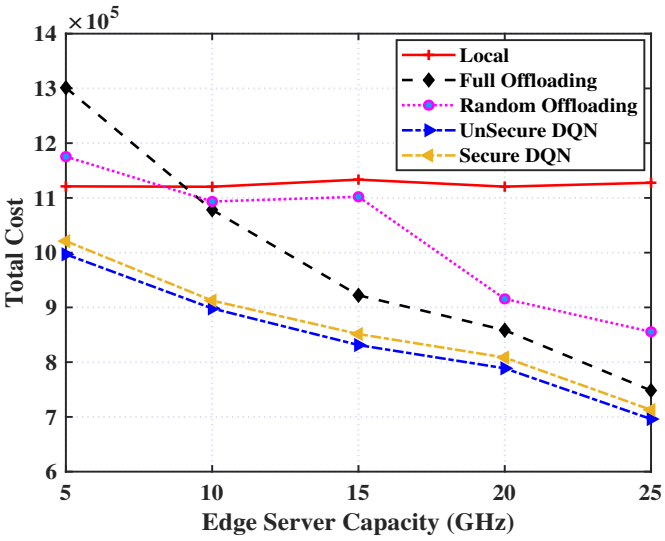

Fig. 9: Total Cost over different MEC capacity

impacted by MEC capacity, whereas the total cost of the other policies gradually declines with the MEC capacity's increasing. This is attributed to the shorter time of execution as the users can be allocated more resources, whereas the MEC resources are not used in the policy of local execution.

VI. Conclusion

Our study proposed a resource allocation and security-aware data Offloading model for a multi-user environment with a multi-task. A new efficient security layer is introduced using the AES algorithm to protect the communicated data against attacks. In addition, a combination model of security, resource allocation and computation offloading is formulated as a problem with the goal of reducing the total time and energy overhead of mobile users. Furthermore, to practically obtain the optimum solution, an equivalent form of reinforcement learning is given, in which the state space is defined using all available solutions and the movement between different states is used to define the actions. Then, an efficient algorithm based on DQN has been proposed for solving this problem and finding the optimum solution. Simulation results demonstrate that the proposed model can achieve performance gains of up to 13.2% and 64.7% of overhead in comparison with full offloading and local execution approaches. Additionally, our DQN-based approach was proven to scale well for the networks with a large-scale.

For a future work, an new layer of compression will be added to our model. This addition will compress the transmission data size to reduce the transmission time and enhance the overall system performance. Additionally, mobile users' mobility will be managed in an efficient manner, in which each user can move dynamically among different edge servers within an offloading period.

Acknowledgment

This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University

through the Fast-track Research Funding Program.

REFERENCES

[1] M. Hammoudeh, R. Newman, C. Dennett, S. Mount, and O. Aldabbas, "Map as a service: A framework for visualising and maximising information return from multi-modalwireless sensor networks," *Sensors*, vol. 15, no. 9, pp. 22 970–23 003, 2015.

[2] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[3] M. Hammad, P. Pławiak, K. Wang, and U. R. Acharya, "Resnet-attention model for human authentication using ecg signals," *Expert Systems*, p. e12547, 2020.

[4] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and sla violation in cloud computing," *IEEE Access*, vol. 6, pp. 55 923–55 936, 2018.

[5] S. Wan, X. Li, Y. Xue, W. Lin, and X. Xu, "Efficient computation offloading for internet of vehicles in edge computing-assisted 5g networks," *The Journal of Supercomputing*, pp. 1–30, 2019.

[6] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng, "Mobile cloud computing: Challenges and future research directions," *Journal of Network and Computer Applications*, vol. 115, pp. 70–85, 2018.

[7] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2737–2746, 2020.

[8] A. Abuarqoub, M. Hammoudeh, and T. Alsboui, "An overview of information extraction from mobile wireless sensor networks," in *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2012, pp. 95–106.

[9] M. Khayyat, A. Alshahrani, S. Alharbi, I. Elgendy, A. Paramonov, and A. Koucheryavy, "Multilevel service-provisioning-based autonomous vehicle applications," *Sustainability*, vol. 12, no. 6, pp. 2497–2513, 2020.

[10] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Transactions on Industrial Informatics*, 2020.

[11] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133 – 3174, 2019.

[12] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile iot networks," *IEEE Transactions on Network and Service Management*, 2020.

[13] W.-Z. Zhang, I. A. Elgendy, M. Hammad, A. M. Iliyasu, X. Du, M. Guizani, and A. A. Abd El-Latif, "Secure and optimized load balancing for multi-tier iot and edge-cloud computing systems," *IEEE Internet of Things Journal*, 2020.

[14] A. Alshahrani, I. A. Elgendy, A. Muthanna, A. M. Alghamdi, and A. Alshamrani, "Efficient multi-player computation offloading for vr edge-cloud computing systems," *Applied Sciences*, vol. 10, no. 16, p. 5515, 2020.

[15] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[16] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[17] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, pp. 1446–1465, 2019.

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

[19] P. Pławiak, M. Abdar, J. Pławiak, V. Makarenkov, and U. R. Acharya, "Dghnl: A new deep genetic hierarchical network of learners for prediction of credit scoring," *Information Sciences*, vol. 516, pp. 401–418, 2020.

[20] M. Khayyat, I. A. Elgendy, A. Muthanna, A. Alshahrani, S. Alharbi, and A. Koucheryavy, "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," *IEEE Access*, 2020.

[21] H. M. Zahera, M. A. Sherif, and A.-C. Ngonga Ngomo, "Jointly learning from social media and environmental data for typhoon intensity prediction," in *Proceedings of the 10th International Conference on Knowledge Capture*, 2019, pp. 231–234.

[22] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[23] M. F. Tuysuz and M. E. Aydin, "Qoe-based mobility-aware collaborative video streaming on the edge of 5g," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7115 – 7125, 2020.

[24] F. N. Nur, S. Islam, N. N. Moon, A. Karim, S. Azam, and B. Shanmugam, "Priority-based offloading and caching in mobile edge cloud," *Journal of Communications Software and Systems*, vol. 15, no. 2, pp. 193–201, 2019.

[25] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[26] ——, "Joint offloading and resource allocation in vehicular edge computing and networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[27] T. Meng, K. Wolter, H. Wu, and Q. Wang, "A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks," *Pervasive and Mobile Computing*, vol. 45, pp. 4–18, 2018.

[28] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Generation Computer Systems*, vol. 100, pp. 531–541, 2019.

[29] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

[30] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[31] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 12–18, 2019.

[32] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, 2019.

[33] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang, "Edge qoe: Computation offloading with deep reinforcement learning for internet of things," *IEEE Internet of Things Journal*, 2020.

[34] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[35] X. Lyu and H. Tian, "Adaptive receding horizon offloading strategy under dynamic environment," *IEEE Communications Letters*, vol. 20, no. 5, pp. 878–881, 2016.

[36] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[37] I. Elgendy, W. Zhang, C. Liu, and C. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[38] I. A. Elgendy, M. El-kawkagy, and A. Keshk, "Improving the performance of mobile applications using cloud computing," in *2014 9th International Conference on Informatics and Systems*. IEEE, 2014, pp. 109–115.

[39] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[40] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[41] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.