IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 40, No. 4, August, 2023, pp. 1329-1339
Journal homepage: http://iieta.org/journals/ts

A Deep Reinforcement Learning Approach for Efficient Image Processing Task Offloading in Edge-Cloud Collaborative Environments

Ming Sun[1,2*], Tie Bao[1], Dan Xie[1], Hengyi Lv[2], Guoliang Si[2]

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2] Changchun Institute of Optics, Precision Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China

Corresponding Author Email: sunm19@mails.jlu.edu.cn

https://doi.org/10.18280/ts.400403

Received: 8 March 2023
Revised: 12 June 2023
Accepted: 18 June 2023
Available online: 31 August 2023

Keywords:
edge computing, task offloading, image processing, multiple users, edge-cloud collaborative, cost efficiency

ABSTRACT

In the wake of the burgeoning Internet of Things (IoT) era and the increasing prevalence of image-based applications on mobile platforms, a significant demand for computing resources has been witnessed. While traditional cloud computing has been limited by substantial transmission distances and notable response delays, mobile edge computing, where communication, computation, and storage resources are situated on edge devices, has emerged as a superior alternative. In this context, the challenge of offloading image processing tasks for multiple users, especially considering the collaboration of edge servers under computational and communication resource constraints, is investigated. A primary objective is to strike a balance between energy consumption and task delays, thereby aiming to curtail the total associated costs. The novel framework introduced, termed as Image Collaborative Task Offloading System using Deep Reinforcement Learning (I-CTOS-DRL), is specifically designed for image processing tasks in edge-cloud collaborative scenarios. Through the integration of a set updating mechanism, complications arising from interactions with neighboring edge servers are effectively diminished. Simultaneously, a heuristic algorithm was constructed to identify the most viable servers for task offloading purposes. Building on this foundation, a pioneering methodology for image processing task offloading was devised, leveraging fully connected neural network training. Evaluations conducted extensively indicate that the proposed strategy outperforms established benchmarks in terms of efficiency.

1. INTRODUCTION

In the current digital era, remarkable developments in image-based applications have been observed. As the proliferation of smartphones, intelligent devices, and IoT gadgets has been documented, a heightened intricacy in real-time image processing has emerged [1, 2]. Often, traditional cloud computing architectures were identified as insufficient to address the evolving needs of these complex image processing tasks, faced with challenges such as increased latency and limited computational capacity.

The impetus for the present research was derived from the pressing need to facilitate real-time image processing across various sectors, spanning healthcare, entertainment, autonomous vehicles, and industrial automation. A mounting urgency has been discerned for the formulation of an adept system proficient in offloading extensive image processing tasks for myriad users spread across diverse geographic terrains.

1.1 Problem definition and contextualization

At the heart of the study lies the investigation of task offloading within the framework of edge-cloud collaboration. Interactions between edge devices, the central cloud data center, and neighboring edge devices were examined, focusing on enhancing image processing efficiency. The intricacy of this paradigm can be exemplified in a multi-user environment, characterized by concurrent image processing tasks (e.g., multiple colored users in various regions as illustrated in Figure 1). If tasks were processed locally, notable delays could be experienced, attributable to the limited computational capacities of individual devices. Although offloading to an edge node or cloud data center might reduce processing time, subsequent challenges, including transmission delay and queuing time, could be encountered.

Figure 1. A motivation example

1.2 Proposed framework: I-CTOS-DRL

In response to these challenges, the I-CTOS-DRL framework has been proposed. The salient contributions of this study include:

- A keen understanding of image processing requirements, with the offloading problem articulated to optimize total costs, encompassing delays and energy expenditure. Collaboration among edge servers was underscored to facilitate efficient real-time image processing.
- The introduction of the I-CTOS-DRL framework, an innovative method utilizing DRL and a heuristic algorithm to reduce inherent complexities. An innovative task updating mechanism for image processing was incorporated, promoting fluid collaboration among edge servers.
- A tailored simulator was devised to rigorously appraise the I-CTOS-DRL methodology. Evaluations corroborated its prowess in managing real-time image processing tasks across distinct sectors.

1.3 Applications and future directions

The I-CTOS-DRL exhibits considerable potential for applications in domains like augmented/virtual reality, digital medical intervention, autonomous driving, and advanced industrial control. A shift in the understanding and application of image processing within the realm of contemporary digital technology is anticipated.

This study augments the existing body of literature by offering a robust solution to the challenges posed by real-time image processing via edge computing. Fresh perspectives in the image processing domain have been unearthed, paving the way for future innovations.

1.4 Conclusion

This research illuminates a significant stride in image processing via edge computing, thereby enhancing the technological panorama. By delineating and addressing the intrinsic challenges of real-time image processing, a prospective era where digital interactions become more intuitive, prompt, and significant is envisioned.

2. RELATED WORK

2.1 Edge computing and task offloading: A focus on image processing

Edge computing has been documented as a pivotal shift, channeling computational resources to the network's periphery. This shift has enabled a multitude of applications, spanning from the IoT to autonomous vehicles, and importantly, image processing. At the heart of this evolution, task offloading has been observed, where the relocation of computationally intensive tasks to edge servers was optimized, yielding a marked reduction in latency [3].

(a) Image Processing within the Edge Computing Paradigm

Emerging task offloading techniques have been linked to the ascension of image processing capabilities in edge computing. Significant advancements in this domain are evidenced by:

- Medical Imaging: Task offloading in ultra-dense networks, as studied by Chen and Hao [4], was found to minimize delays in medical image processing. Concurrently, battery conservation was achieved through software-defined networks.
- Video Analytics: Through their research, Xia et al. [5] established both exact solutions and approximation algorithms, facilitating the offloading of real-time video processing tasks within multi-cell mobile edge computing environments.

(b) Integration of Machine Learning in Image Processing

The incorporation of machine learning has been acknowledged as a transformative step in the domain of intelligent image processing. Key developments include:

- Machine Learning in Task Offloading: A decision tree-based design for image classification was presented by Guo et al. [6].
- Disease Recognition: Enhanced recognition capabilities for early-stage brown spot disease in paddy leaves were attained through a CNN-based model, as documented by Upadhyay and Kumar [7].
- Alzheimer's Classification: It was reported by Thayumanasamy and Ramamurthy [8] that the DenseNet-169 model exhibited superior performance in Alzheimer's disease classification using brain MRI scans.
- Underwater Image Enhancement: A deep underwater image enhancement network utilizing a convolutional neural network algorithm was put forward by El Rejal et al. [9], showcasing enhancements in visibility, contrast, and the overall quality of deep-sea images.

(c) Efficiency-Driven Strategies in Image Processing

Prioritizing efficiency has remained a cornerstone of edge computing strategies tailored for image processing. Noteworthy strategies encompass:

- Power and Delay Reduction: Efforts to minimize system power consumption and delays during image execution were pursued by Keshavarznejad et al. [10], utilizing meta-heuristic techniques.
- Tomato Seedling Recognition: Zhang [11] introduced an information acquisition method leveraging the Cycle-Consistent Adversarial Network. This method was demonstrated to achieve a recognition accuracy ranging between 91% to 97%, bolstering the automation capabilities of tomato transplanters.

2.2 Evolution of task offloading within edge-cloud interfaces

The rapid proliferation of mobile devices has been perceived to amplify the demands on edge networks. In response to these increasing pressures, several innovative solutions have been proposed, as outlined below:

(a) Blockchain-Integrated Architectures for Image Data Processing

- A blockchain-facilitated edge-cloud computing architecture was put forward by Wu et al. [12]. This framework aimed to bridge the gap between mobile cloud computing and mobile edge computing, seeking to streamline the efficiency of image data processing.

(b) Defensive Measures Against Distributed Denial of Service Attacks

- In an effort to counteract volumetric-based Distributed Denial of Service attacks targeting cloud and edge computing networks, Yudhana et al. [13] integrated the Packet Filtering Firewall and Circuit Level Gateway Firewall. It was observed that these measures led to a substantial decline in both traffic and server resource usage, with reductions ranging between 64%-98.88% and reaching up to 96% respectively.

(c) Advanced Techniques for Image Edge Detection

• Moussa and Douik [14] ventured into refining edge detection methodologies for image processing. By leveraging information theory combined with metaheuristic and intelligent algorithms, a considerable improvement in execution time was achieved when compared to conventional operators.

(d) 3D Visual Image Recognition in Athletic Training

• An innovative 3D visual image recognition technique based on contourlet domain edge detection was elucidated by Wang [15]. This method, specifically tailored for recognizing and rectifying athletes' improper motions during training, showcased noticeable advancements in the accuracy of motion evaluations.

(e) Profit-Centric Offloading in Mobile Edge Computing Servers

• Tunga et al. [16] unveiled a strategy aimed at maximizing intrinsic profit when offloading tasks to Mobile Edge Computing servers. Given the constraints of fixed memory capacities and the necessity for low latency, the Ant Colony Optimization model was adeptly employed, demonstrating its efficacy in real-time applications.

2.3 Reinforcement learning's role in edge-based image processing

The incorporation of reinforcement learning into image processing at the edge has been increasingly examined. A notable instance includes the work by:

• Qu et al. [17-19], wherein a deep meta-reinforcement learning-based offloading algorithm was developed. This algorithm, tailored for intricate image processing decisions, harnessed the capabilities of several parallel deep neural networks.

2.4 Opportunities and challenges ahead

Despite the significant strides observed in amalgamating edge computing, task offloading, and image processing, persistent challenges remain. These primarily relate to the optimization of action and state spaces. Such challenges hint at a vast scope for deeper exploration, suggesting that a synthesis of reinforcement learning, task offloading, and image processing in the paradigm of edge-cloud cooperation might offer promising avenues of investigation.

2.5 Envisaging the evolution of task offloading in image processing in edge-cloud interfaces

Efforts have been directed toward enhancing task offloading methodologies grounded in Deep Reinforcement Learning (DRL), especially for applications in image processing. The predominant goal here revolves around reducing both latency and energy expenditure within the ambit of edge-cloud collaborative computing. This endeavor is perceived to be a pivotal stride in refining edge computing frameworks and task offloading mechanisms specifically for image processing applications.

3. MODELS AND PROBLEM FORMULATION

3.1 The edge-cloud system architecture

In this research, an overarching topology of the edge-cloud framework, with an emphasis on image processing tasks, is delineated by the notation $\boldsymbol{G} = \{\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{U}\}$.

• Cloud Data Center ($\boldsymbol{C}$): Situated distally from the user group and acting as the pinnacle of the system's hierarchy, the cloud data center, designated by **C**, is recognized as the central hub for expansive image processing undertakings.

• Edge Nodes and Base Stations: It is observed that the user cluster interacts with edge nodes via wireless channels and base stations. These edge layer entities, with their heterogeneous and autonomous stances, span across diverse regions. The notational representation, $\boldsymbol{M} = \{m_k\}$, has been adopted to capture the spread of edge nodes facilitated by various operators. Every individual edge node, given by m_k, is noted to be connected to its respective base station with a predetermined computing potential reserved for image data tasks. The computing capacity of each m_k is subsequently denoted by f_k.

• Adjacent Edge Servers: These are depicted by $\boldsymbol{A}(m_k)$. It is inferred that the selection of neighboring edge servers for cooperation can substantially enhance the effectiveness of image processing and dissemination.

• User Layer: Within this structure, the set of users interfacing with the edge node m_k is represented as $\boldsymbol{U}^{m_k} = \{u_i^k\}$. Users are found to establish connections with the closest edge node, which in turn augments the system's capability to swiftly address image processing demands.

• End Devices: Such devices, embodying users within this schema, range from computers and smartphones to smart wristbands. A continuous stream of atomic tasks is generated by these entities. These tasks, ranging from image recognition to filtering and enhancement, are intrinsic to image processing. The computational capacity of u_i^k, enveloping the essentialities of image processing, is indicated by l_i^k.

For a comprehensive understanding, Table 1 has been curated to enlist pivotal notations, with a special focus on those crafted to elucidate the specifics of image processing tasks.

Table 1. Symbol definition

Symbols	Definitions
$\boldsymbol{G}$	Topology of edge-cloud architecture, $\boldsymbol{G} = \{\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{U}\}$
$\boldsymbol{M}$	Set of edge nodes in $\boldsymbol{G}$, where $\boldsymbol{M} = \{m_k\}$
m_k	The k^{th} edge node in $\boldsymbol{M}$
f_k	The computing capacity of edge node m_k
$\boldsymbol{U}^{m_k}$	Set of users located in the area of m_k, where $\boldsymbol{U}^{m_k} = \{u_i^k\}$
u_i^k	The i^{th} user located in the area of m_k, where $u_i^k \in \boldsymbol{U}^k$
l_i^k	The computing capacity of u_i^k
χ_i^k	Set of tasks, including image processing tasks, where $\chi_i^k = \{w_i^k, d_i^k, \delta_i^k, \tau_i^k\}$
w_i^k	Task workload produced by user u_i^k, including image processing workload
d_i^k	Data size of user u_i^k, specifically related to image data size
δ_i	Ratio of output to input data volume user u_i, for image transformation tasks
τ_i	The maximum tolerant delay of user u_i, for image processing
$\boldsymbol{A}(m_k)$	The set of adjacent edge nodes of m_k
$\mathbb{Z}(m_k)$	The feasible collaborative set of m_k
$\boldsymbol{D}(u_i^k)$	The total delay of user u_i^k
$\boldsymbol{E}(u_i^k)$	The total energy consumption of user u_i^k
$\Psi(u_i^k)$	The total cost of user u_i^k
$\phi(s_t)$	The updating function of state s_t

3.2 Transmission model for image processing tasks

In this segment, a specialized transmission model has been formulated, concentrating predominantly on the intricacies of managing image processing tasks. Emphasis is placed on the offloading of tasks by users to edge servers. The pronounced need for extensive bandwidth and dependable transmission arises from the nature of high-resolution image data. Under this framework, the collaborative essence of edge nodes is highlighted, ascertaining proficient offloading to either a directly connected edge node or its collaborative counterpart.

Based on the Rayleigh fading channel model [20], the transmission rate $r_{u_i^k,m_k}$ between user u_i^k and the associated edge node m_k is articulated as:

$$r_{u_i^k,m_k} = b_{i,k} \cdot \log_2\left(1 + \frac{\gamma_{i,k} h_{i,k}}{p(u_i^k, m_k)^{\omega_i} N_i}\right) \tag{1}$$

where, $b_{i,k}$ and $h_{i,k}$ are established to signify the channel gain and transmission bandwidth between user u_i^k and m_k, respectively. Given the data-heavy constitution of images, these parameters become central to the process. Concurrently, $p(u_i^k, m_k)$ defines the spatial relation between u_i^k to m_k, while $\gamma_{i,k}$ represents the transmission power. Parameters like Gaussian noise and path loss exponent, represented by N_i and ω_i respectively, are identified as cardinal for maintaining transmitted image data's authenticity.

In parallel, the transmission rate r_{m_k,m_h} between edge nodes m_k and m_h is framed as:

$$r_{m_k,m_h} = B_{k,h} \cdot \log_2\left(1 + \frac{\Upsilon_{k,h} H_{k,h}}{p(m_k, m_h)^{\omega_i} N_i}\right) \tag{2}$$

The above Eq. (2) elucidates the channel gain $B_{k,h}$ and transmission bandwidth $H_{k,h}$ between the said edge nodes. By facilitating collaborations in image processing, an uptick in efficiency and a dip in latency are witnessed. The spatial dynamics between m_k and m_h are mapped by $p(m_k, m_h)$, and $\Upsilon_{k,h}$ documents the transmission power. The Gaussian noise and path loss exponent for the edge node m_k are symbolized by N_k and ω_k, respectively.

In the presented model, meticulous considerations are made to cater to the unique transmission requirements of images. Factors such as bandwidth, channel gain, and inter-node collaboration are seamlessly integrated to form a comprehensive framework. This model, thus, offers insights tailored to fortify reliable image processing tasks, cognizant of the nuanced challenges that high-fidelity image data transmission incurs.

3.3 Execution model for image processing

The concept of task offloading, particularly for image processing tasks, is examined within a three-tier edge-cloud framework. Image processing tasks possess unique attributes like high computational requirements and significant data volume, necessitating a distinct approach when considering offloading. Potential offloading locations for these tasks could encompass the local device, the directly connected edge node, collaboratively connected edge nodes, or the cloud. Various scenarios have been identified and the execution models are outlined as follows:

(1) Execution on Local Device

The offloading issue is initially envisioned for multiple users choosing to conduct image processing tasks locally. These tasks could encompass image enhancement, filtering, or object detection, all of which demand considerable computational resources. The execution delay for a user, denoted by $D_l(u_i^k)$, is formulated as follows:

$$D_l(u_i^k) = \frac{w_i^k}{l_i^k} \tag{3}$$

In this context, w_i^k represents the workload, and l_i^k signifies the CPU frequency of the user u_i^k. The energy consumption for task χ_i^k of u_i^k is then defined:

$$E_l(u_i^k) = \kappa_i^k \cdot w_i^k \cdot (l_i^k)^2 \tag{4}$$

where, κ_i^k is the coefficient factor of chip architecture for user u_i^k, reflecting the specific requirements of image-related computations.

(2) Execution on Connected Edge Node

Subsequently, the scenario is studied wherein the image processing tasks are executed on the directly connected edge node. This strategy proves to be advantageous for real-time image analytics. The transmission delay and the execution delay between u_i^k and m_k are respectively defined as follows:

$$D_m^t(u_i^k) = \frac{d_i^k}{r_{u_i^k,m_k}} \tag{5}$$

where, d_i^k is the data size of u_i^k, reflecting significant volume of image data. The execution delay on edge node m_k is

$$D_m^e(u_i^k) = \frac{w_i^k}{f_k} \tag{6}$$

Thus, the total delay for image processing on the connected edge node is expressed as:

$$D_m(u_i^k) = D_m^t(u_i^k) + D_m^e(u_i^k) \tag{7}$$

And the energy consumption is given by:

$$E_m(u_i^k) = \gamma_{i,k} \cdot D_m^t(u_i^k) \tag{8}$$

where, $\gamma_{i,k}$ is the transmission power from user u_i^k to m_k.

(3) Execution on Collaborative Edge Node

For more intricate image processing tasks, an analysis is conducted where execution on the collaborative edge node is contemplated. The transmission delay in this case is composed of two parts:

$$D_z^t(u_i^k) = \frac{d_i^k}{r_{u_i^k,m_k}} + \frac{d_i^k}{r_{m_k,m_h}} \tag{9}$$

where, $\frac{d_i^k}{r_{u_i^k,m_k}}$ is the transmission delay from user u_i^k to edge node m_k. $\frac{d_i^k}{r_{m_k,m_h}}$ is the transmission delay from edge nodes m_k to m_h.

Since the task produced by user u_i^k processes on edge node m_h, the execution delay $D_z^e(u_i^k)$ is defined as:

$$D_z^e(u_i^k) = \frac{w_i^k}{f_h} \tag{10}$$

where, f_h is the computing capacity of m_h. So the total delay for executing image tasks on the collaborative edge node is:

$$D_z(u_i^k) = D_z^t(u_i^k) + D_z^e(u_i^k) \tag{11}$$

which is the sum of the transmission delay and the execution delay. Similar to the scenario of execution on the connected edge node, the energy consumption of execution on the collaborative edge node is defined as:

$$E_z(u_i^k) = \gamma_{i,k} \cdot \frac{d_i^k}{r_{u_i^k,m_k}} + \Upsilon_{k,h} \cdot \frac{d_i^k}{r_{m_k,m_h}} \tag{12}$$

where, $\Upsilon_{k,h}$ is the transmission power from m_k to m_h.

(4) Execution on Cloud

Lastly, the prospect of executing image processing tasks on the cloud is scrutinized, particularly for tasks that necessitate significant computational resources. The transmission delay and energy consumption in this case are defined as follows:

$$D_c^t(u_i^k) = \frac{d_i^k}{r_{u_i^k,\mathbf{c}}} \tag{13}$$

where, $r_{u_i^k,C}$ is the transmission rate between user u_i^k and the cloud $\boldsymbol{C}$. Based on that, the energy consumption of execution on the cloud is defined as:

$$E_c(u_i^k) = \gamma_{i,k} \cdot \frac{d_i^k}{r_{u_i^k,\mathbf{c}}} \tag{14}$$

This section has conducted a thorough exploration of diverse execution models within an edge-cloud framework, with a specific focus on image processing tasks. Explicit expressions for delays and energy consumption across different scenarios have been provided, incorporating the unique attributes of image processing into a cohesive framework. This investigation aids in understanding the specific requirements and potential advantages of different offloading strategies, ensuring the efficient use of computational resources and effective management of the unique aspects of image data. The models proposed here can further guide the design and optimization of edge-cloud architectures for real-time image analytics.

4. PROBLEM FORMULATION

4.1 Representation of image processing tasks

In the rapidly evolving landscape of image processing tasks, which includes activities such as object detection, image classification, filtering, enhancement, and other complex computational tasks, a critical need has arisen to optimize both energy consumption and delay in multi-user scenarios. Numerous users are engaged in a variety of image processing tasks, and decisions related to offloading need to be carefully crafted to meet the unique requirements of each task.

4.2 User delay and energy consumption

The unique characteristics of tasks generated by users, particularly those related to image processing, demand a simultaneous optimization of the overall energy consumption and delay experienced by multiple users. The total delay for a specific user, denoted by u_i^k, is articulated as follows:

$$\begin{aligned}\boldsymbol{D}(u_i^k) = \alpha^l \cdot D_l(u_i^k) \\ +\alpha^m \cdot D_m(u_i^k) \\ +\alpha^z \cdot D_z(u_i^k) + \alpha^c \cdot D_c(u_i^k)\end{aligned} \tag{15}$$

The Boolean variables α^l, α^m, α^z, α^c are defined, with the constraints:

$$\alpha^l + \alpha^m + \alpha^z + \alpha^c = \{0,1\} \tag{16}$$

The overall energy consumption of use u_i^k is denoted as $\boldsymbol{E}(u_i^k)$, and is formulated as:

$$\begin{aligned}\boldsymbol{E}(u_i^k) = \beta^l \cdot E_l(u_i^k) \\ +\beta^m \cdot E_m(u_i^k) \\ +\beta^z \cdot E_z(u_i^k) + \beta^c \cdot E_c(u_i^k)\end{aligned} \tag{17}$$

The constraints on the Boolean variables β^l, β^m, β^z and β^c are defined:

$$\beta^l + \beta^m + \beta^z + \beta^c = \{0,1\} \tag{18}$$

Subsequently, the total cost for user u_i^k is defined as:

$$\Psi(u_i^k) = \alpha \cdot \boldsymbol{D}(u_i^k) + \beta \cdot \boldsymbol{E}(u_i^k) \tag{19}$$

4.3 Objective function

The main goal of this formulation is to identify an effective framework that minimizes the aggregate cost of image processing and associated tasks. This includes various challenges inherent to image processing, such as handling high-resolution images, complex image analysis computations, and requirements for real-time responses. The formulation is outlined as follows:

$$\text{minimize} \sum_{k=0}^{|\boldsymbol{M}|} \sum_{i=1}^{|\boldsymbol{U}^{m_k}|} \Psi(u_i^k) \tag{20}$$

subject to:

$$\boldsymbol{D}(u_i^k) \leq \tau_i^k \tag{21}$$

$$\alpha, \beta \in [0,1], 0 \leq \alpha + \beta \leq 1 \tag{22}$$

$$\forall u_i^k \in \boldsymbol{U}^{m_k}, \forall m_k \in \boldsymbol{M} \tag{23}$$

4.4 Complexity of image processing offloading

The challenge of task offloading, given capacity constraints while optimizing overall energy consumption and delay for multiple users, is identified as NP-hard. The inherent complexity of image processing tasks introduces an additional

layer of intricacy to this optimization puzzle.

4.5 Special considerations for image processing

Within the sphere of image processing, certain considerations are crucial, including the need for powerful computational capabilities, sensitivity to delay, and a focus on energy efficiency. Unique algorithms may require specific offloading strategies. The above formulation thoroughly encompasses these aspects, thereby providing a solid foundation for the analysis and application of image processing tasks.

5. DRL-BASED COLLABORATIVE IMAGE PROCESSING AND TASK OFFLOADING FRAMEWORK (CTOS-DRL-IP)

This section introduces a DRL-based Collaborative Task Offloading and Image Processing Framework (CTOS-DRL-IP). This framework extends the CTOS-DRL, with a focused objective of optimizing image processing in edge network environments. The key goal of CTOS-DRL-IP is the efficient management of image processing tasks by crafting a practical collaborative set during the decision-making process. This is accomplished using deep Q-learning. The subsequent sections provide an in-depth discussion of the components and functions of the CTOS-DRL-IP.

5.1 DRL formulation for image processing

In this subsection, the CTOS-DRL-IP framework, inspired by deep Q-learning and specifically tailored for image processing applications, is introduced. This includes the representation of the total cost and the agent's knowledge of image processing applications within the state space to accurately model the edge network environment.

Definition 1 (State):

The state, denoted as s_t, is defined as a vector consisting of $s_t = \left[T_i^k, U_i/\widehat{U}_i, I_i^p\right]_t$. Here $\boldsymbol{U}_i/|\widehat{\boldsymbol{U}}_i|$ represents the task generated by users that requires processing. $\boldsymbol{T}_i^k = \sum_{k=1}^{|\hat{U}_i|} T_i^k$ signifies the cumulative cost of the scheduled completed tasks $\widehat{\boldsymbol{U}}_i$, while the last part of the vector symbolizes the attributes of the image processing task, such as resolution, type, and processing requirements.

Definition 2 (Action):

The action space a_t, embodying the adjusting action, is represented by the vector $\left[\zeta_i^l, \zeta_i^c, \zeta_i\right]_t$. Within this, ζ_i^l or ζ_i^c indicates whether the destination location of the adjustment resides on a local device or in the cloud. The vector ζ_i, denoting the set of feasible edge nodes $\mathbb{M}_i$ is characterized by $\zeta_i = \left(\zeta_i^m(k)|_{m_k \in \mathbb{M}_i}\right)$. Additionally, depicts the processing action specific to the image task, such as filtering, sharpening, or scaling.

Definition 3 (Reward):

The immediate reward is expressed by $R(s_t, a_t) = (\bar{E}_i - E)/\bar{E} - \alpha \cdot Q_i^p$, where $\overline{\boldsymbol{E}}_i$ is the aggregate cost of multiple tasks for users, $\overline{\boldsymbol{E}}$ is a fundamental cost that is being selectively offloaded, and $\alpha \cdot Q_i^p$ is a quality factor pertinent to image processing. This factor ensures a balance between cost efficiency and quality of image processing.

In this context, the agent is prepared by selecting a destination for various users and defining particular image processing tasks. Training the agent, as a result, is anticipated to facilitate the simultaneous accomplishment of offloading and image processing tasks. The action a_t is designed for each time slot *t*, and the agent is rewarded $R(s_t, a_t)$ in a given state s_t. Efforts are made to minimize the total cost, comprised of energy consumption and delay, without undermining the quality of image processing. This aligns with the reinforcement learning objective of maximizing long-term reward.

In conclusion, the CTOS-DRL-IP framework presents an innovative method for collaborative image processing and task offloading in edge networks, utilizing DRL. Through the provision of tailored state, action, and reward definitions, the framework provides a robust solution for managing complex image processing tasks. This results in the achievement of optimal energy efficiency and a reduction in delay without a compromise in image quality. The insights and findings from this study extend the current understanding of the field, opening new avenues for exploration and application in real-world scenarios.

5.2 Feasible collaborative set construction with image processing consideration (FCSC-IP)

The goal of the agent operating within the CTOS-DRL-IP framework is to achieve effective task offloading for multiple users, aiming to minimize the overall cost, with special attention given to image processing requirements. The nature of these decisions depends on the observation of the environment and the overall architecture, which, given the constraint of edge server capacities, influences the number of tasks that can be offloaded to individual regions. When an edge node reaches its capacity, collaboration with neighboring nodes becomes necessary, particularly when managing complex image processing tasks that require stringent adherence to processing, quality, and time parameters. The complexity and high dimensionality of this collaborative set construction process are addressed through the introduction of the FCSC-IP method.

5.2.1 Image processing integration into collaborative decisions

In an edge network comprising seven interconnected edge nodes, where each node has the ability to choose collaboration partners for task completion, decisions are not made solely based on proximity and availability. A novel method, FCSC-IP, is proposed, taking into account specific image processing capabilities such as resolution handling, filtering, and transformation. Users serving at edge nodes m_1 that connect with m_2 to m_5 will have collaboration options, with consideration given to the complexity introduced by the adjacency matrix.

Definition 4 (Collaborative Influence Factor with Image Processing Consideration):

The collaborative influence factor with image processing consideration, denoted as $\varphi(m_k)$, is formulated to assess the collaborative capacity of edge node $m_k \in \mathbf{A}(m_k)$ for user m_k, reflecting the image processing requirements:

$$\phi(m_k) = \frac{f_h}{(p(m_k, m_h) \cdot |U^{(m_k)}|)} \times \beta\left(I_i^p\right)$$

where, $\beta\left(I_i^p\right)$ serves as a weight parameter, factoring in image processing attributes such as type, resolution, and complexity.

Definition 5 (Feasible Collaborative Set with Image Processing Consideration):

The feasible collaborative set with image processing consideration, denoted as $\mathbb{Z}(m_k) = \{m_h\}$, is derived from the edge nodes in set $\boldsymbol{A}(m_k)$, with the collaborative influence factor under the boundary parameter Γ_{m_k}, such that $\varphi(m_h) \leq \Gamma_{m_k}$.

5.2.2 Algorithm for FCSC-IP

Algorithm 1. Feasible Collaborative Set Construction, FCSC

Input: The topology G of the edge-cloud architecture;
Output: The feasible collaborative set $\mathbb{Z}$ of each edge node in **M**;
1: **for** each edge node m_k in **M do**
2: Initialize the set $\mathbf{U}^j$ in the area of edge node m_k;
3: Calculate the boundary parameter Γ_{m_k} according to the maximum tolerant delay of users in set $\mathbf{U}^j$, where $\Gamma_{m_k} = \arg\min_{u_i^k \in \mathbf{U}^{m_k}}\{\tau_i^k\}$;
4: Initialize the set $\mathbf{A}(m_k)$ of adjacent edge nodes of m_k;
5: **for** each adjacent edge node m_h in $\mathbf{A}(m_k)$ **do**
6: Calculate the collaborative influence factor $\varphi(m_h) = \frac{f_h}{p(m_h,m_k)\cdot|\mathbf{U}^h|}$;
7: **if** $\varphi(m_h) \leq \Gamma_{m_k}$ **then**
8: Adding edge node m_h into set $\mathbb{Z}$;
9: Update $\mathbf{A}(m_k) = \mathbf{A}(m_k)/m_h$;
10: **end if**
11: **end for**
12: **end for**
13: **return** the feasible collaborative set $\mathbb{Z}$;

The procedure for feasible collaborative set construction is outlined in Algorithm 1, with explicit attention to the requirements of image processing. The FCSC-IP algorithm focuses on image processing needs, aiming to reduce complexity while improving efficiency in the formation of a feasible collaborative set. The time complexity of FCSC-IP is $O(|\boldsymbol{M}| \cdot |\boldsymbol{U}^h|)$, taking into account the attributes of image processing.

In conclusion, the FCSC-IP presents an innovative method that enables edge nodes to participate in collaborative decisions in task offloading, specifically reflecting the complexity inherent in image processing tasks. By integrating image processing prerequisites into both collaborative influence factors and feasible set construction, FCSC-IP offers a balanced and efficient solution, harmonizing cost, efficiency, and quality in managing complex image processing tasks within edge networks. This method not only enhances existing offloading strategies but also provides a nuanced understanding of the interaction between task distribution and image processing, making a significant contribution to the field.

5.3 CTOS-DRL for image processing tasks

In this subsection, a task offloading algorithm specifically designed to meet the demands of image processing tasks is introduced, grounded in DRL. The fundamental principle of CTOS-DRL, summarized in Algorithm 2, involves using a DRL agent to facilitate dynamic offloading of image processing tasks from multiple users, with the goal of minimizing the total cost.

The set of image processing tasks, denoted by χ, comes from various users in set U and forms the input. The output, denoted as the offloading strategy X, might include numerous image manipulations such as filtering, object detection, segmentation, and enhancements that require substantial computational resources.

The approach is summarized as follows:

Algorithm 2 CTOS-DRL

Input: The tasks χ generated by users in set **U**;
Output: Offloading strategy $\mathbb{X}$;
1: Initialize Θ to N, Q with random weights θ, and $\hat{Q}$ with weights $\bar{\theta} := \theta$;
2: **for** episode from 1 to κ **do**
3: Initialize sequence s for multiple users;
4: Construct the feasible set $\mathbb{Z}$ based on Algorithm 1;
5: **for** t from 1 to T **do**
6: Select a random action a_t with probability ε under the feasible set $\mathbb{Z}$;
7: Otherwise select $a_t = \text{argmax}_a\ Q(\phi(S_t), a; \theta)$;
8: Set $s_{t+1} = s_t, a_t, x_{t+1}$,and preprocess $\phi_{t+1} = \phi(s_{t+1})$;
9: Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in Θ;
10: Sample random minibatch of transitions$(\phi_j, a_j, r_j, \phi_{j+1})$ from Θ;
11: **if** episode terminates at step $j + 1$ **then**
12: Set $y_j = r_j$;
13: **else**
14: Set $y_j = r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)$;
15: Perform a gradient descent step on $\left(y - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ;
16: Reset $\hat{Q} = Q$ every C steps;
17: **end if**
18: **end for**
19: **end for**
20: **return** Offloading strategy $\mathbb{X}$ for multiple users

(1) Initialization: Basic settings are initialized (Line 1), including the capacity N of replay memory Θ, the target action-value function $\hat{Q}$ with weights $\bar{\theta} := \theta$, and the random weights θ of action-value function Q.

(2) Environment Familiarization: Over a series of kappa episodes, the agent becomes familiar with the environment, tailored to the complex nature of image processing tasks (Lines 2 to 19).

(3) Sequence Initialization: The sequence for multiple users is initialized (Line 3), considering the unique characteristics inherent to image processing tasks.

(4) Collaborative Set Construction: A feasible collaborative set Z is constructed (Line 4) based on Algorithm 1, with specific attention to the constraints and requirements of image processing.

(5) Training Process: Spanning lines 5 to 18, the agent's decision-making process encompasses action selection (lines 6-7), state and action setting (line 8), storing transitions in replay memory (line 9), and applying gradient descent to minimize total cost (lines 14 to 16).

(6) Final Offloading Strategy: Representing the optimized solution for handling image processing tasks, the offloading strategy X is returned (Line 20).

(7) Time Complexity: The time complexity of CTOS-DRL is denoted by $O\left(\kappa \cdot |\boldsymbol{M}| \cdot |\boldsymbol{U}^h| \cdot \left|\tilde{\boldsymbol{U}}_i\right|\right)$, where κ represents the number of episodes, and U ˜_i refers to the scale of sub-tasks pending scheduling on edge server mi, expressed as $\left|\tilde{\boldsymbol{U}}_i\right| = \left|\boldsymbol{U}_i/\hat{\boldsymbol{U}}_i\right|$.

The CTOS-DRL algorithm provides a robust and efficient method for managing the offloading of image processing tasks. By recognizing and accommodating the specific requirements and complexities associated with image processing, it offers a flexible and optimized solution that can adapt to the dynamic

nature of contemporary applications involving image manipulation.

6. EXPERIMENTAL EVALUATION FOR IMAGE PROCESSING TASKS

This section provides a comprehensive investigation of the task offloading problem through both simulations and experimental methodologies. A particular emphasis is placed on image processing tasks within the realm of edge-cloud collaborative computing, echoing the growing demand for real-time image processing in current applications such as facial recognition, augmented reality, medical imaging, and surveillance.

A prototype framework was utilized for these experiments, developed using Python. This included the structure of the edge network and the generation of image processing tasks by multiple users, covering operations such as image classification, segmentation, and enhancement. These tasks are representative of the common requirements in contemporary applications, requiring quick and proficient image processing.

The experimental results were meticulously examined from multiple perspectives, incorporating both computational and communicational aspects of the offloading strategies, with the quality of image processing acknowledged as a critical factor in performance assessment.

6.1 Basic setting for image processing task evaluation

This subsection explores the effectiveness of the CTOS-DRL approach to the task offloading problem, focusing on image processing tasks, using a synthetic dataset generated by 6 to 10 edge nodes. The tasks included areas such as filtering, object detection, enhancement, and segmentation.

A 500 square meter area, interacting with 6 to 14 mobile users for each edge server, was included in the analysis. This setup mirrors real-world situations where image processing tasks might come from various sources, including mobile applications, surveillance cameras, or IoT devices.

Table 2 outlines the essential parameter settings employed in this study, tailored to meet the specific demands of image processing, as referenced in literatures [20-22].

Four baseline algorithms were considered for comparative evaluation in these experiments, each evaluated for its capacity to manage image processing tasks:

- Offloading Tasks on Local Devices (OTLD): Image processing tasks generated by users were iteratively offloaded on local devices, considering the computational capabilities and power limitations of these devices.
- Offloading Tasks on Edge Nodes (OTEN): Image processing tasks generated by users were iteratively offloaded on directly connected edge nodes, assessing the implications on processing time and quality.
- Offloading Tasks on the Cloud (OTC): Image processing tasks generated by users were iteratively offloaded on the cloud, focusing on scalability and potential latency.
- Offloading Tasks through Greedy Strategy (OTGS): The offloading location was chosen based on total cost consideration for each iteration of image processing tasks created by users, taking into account both computational and communication expenses.

CTOS-DRL was compared with these four baseline algorithms, and the effectiveness of CTOS-DRL in managing image processing tasks was confirmed.

By incorporating the unique prerequisites and constraints of image processing into the experimental design, a robust foundation was established for evaluating the performance of the proposed algorithm in a relevant context. These results further validate the practicality and efficiency of the CTOS-DRL approach in offloading image processing tasks within edge-cloud collaborative computing environments.

Table 2. Setting of parameters for image processing tasks

Parameters	Values
The path loss exponent ω_i	3
The data volume of tasks (specific to image processing)	0.5Mb~1Mb
The local device's transmission power $\gamma_{i,k}$	3W
The local device's computing capacities (suitable for image processing)	0. 5-1GHz
The edge nodes' computing capacities (optimized for image processing)	5GHz
The channel's fading coefficient	10^{-6}
The chip architecture coefficient factor	10^{-20}

6.2 Performance evaluations for image processing tasks

In this subsection, the performance of the strategy was evaluated, focusing on image processing tasks such as object detection, segmentation, and recognition. The goal was to collaboratively optimize the overall energy consumption and delay for these tasks. In this regard, the results were examined in three distinct groups, characterized by α=0.3, β=0.7, α=0.5, β=0.5, and α=0.4, β=0.6. These groups, with varying parameters, indicate the proportion of delay and energy consumption, crucial factors for real-time image processing.

A comparative analysis was performed to understand the impact of the number of servers within identical strategies. Here, the average total cost was calculated under the same number of servers but varying numbers of users. Focus was given to the relationship between costs and key variables such as image quality, processing speed, and energy efficiency. These are particularly relevant for edge devices constrained by limited computational resources.

The findings, illustrated in Figures 2-4, led to several key conclusions:

(1) It was observed that the total costs under OTLD and OTC were significantly higher than other strategies. This pattern became more pronounced when dealing with complex image processing tasks, which can place increased computational demands on local devices. An analysis was conducted on the scenario with increasing edge servers in OTLD and OTC, revealing the inherent shortcomings of these strategies in managing image processing tasks.

(2) Conversely, the total costs under OTEN, OTGS, and CTOS-DRL were found to decrease with the expansion of edge servers. Specifically for image processing tasks, CTOS-DRL showed the lowest total cost, enhancing efficiency by an average of 48.93% compared to OTEN, and 22.29% relative to OTGS.

Further insights were derived from the graphs in Figures 5-10, providing comprehensive comparisons of costs under various edge server conditions.

In the final analytical segment, Figures 11-13 depicted the convergence of the strategies, showcasing how CTOS-DRL operated across different scenarios of image processing. This included tasks such as real-time object tracking, feature

extraction, and image enhancement. The examination underscored the substantial improvements brought about by CTOS-DRL, particularly within groups with smaller scale edge servers. This presents a powerful solution for mobile edge computing within the realm of image processing applications.

The explicit integration of specific image processing tasks added a layer of practical relevance to the analysis, affirming the applicability of the findings to real-world scenarios where efficient image processing at the edge remains of utmost importance.

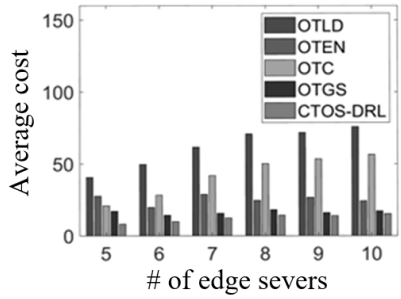

Figure 2. The average cost of the first group (α=0.3, β=0.7)

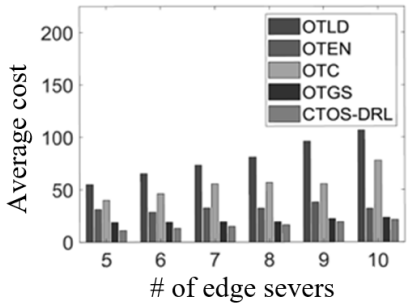

Figure 3. The average cost of the second group (α=0.4, β=0.6)

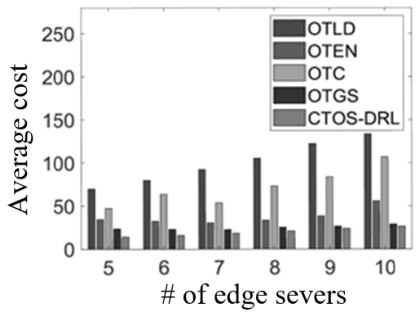

Figure 4. The average cost of the s third group (α=0.5, β=0.5)

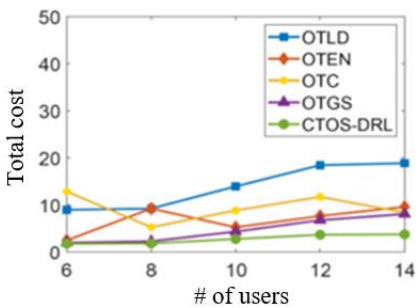

Figure 5. Total cost under 5 edge servers

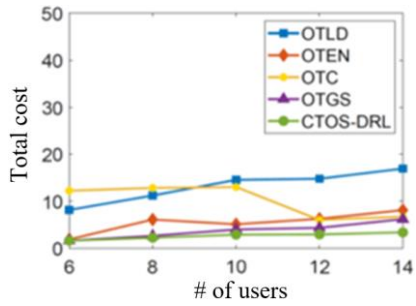

Figure 6. Total cost under 6 edge servers

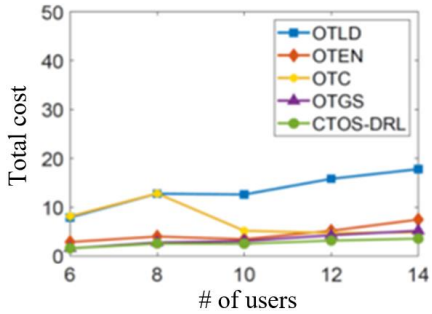

Figure 7. Total cost under 7 edge servers

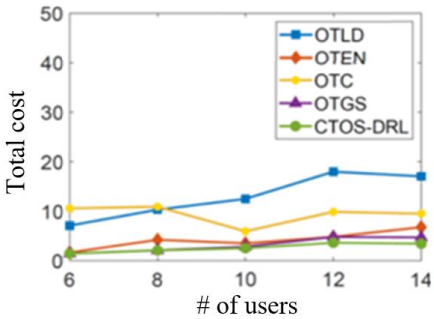

Figure 8. Total cost under 8 edge servers

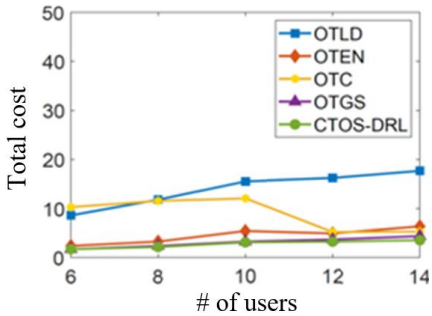

Figure 9. Total cost under 9 edge servers

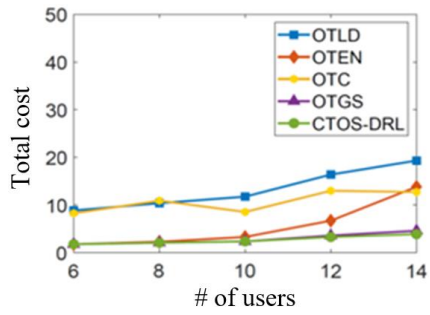

Figure 10. Total cost under 10 edge servers

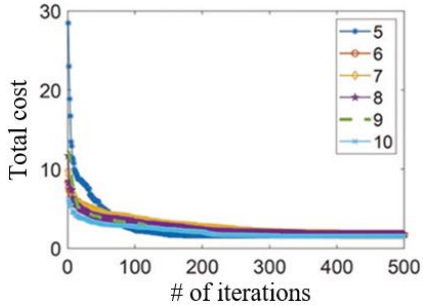

Figure 11. The convergence of the s first group (α=0.3, β=0.7)

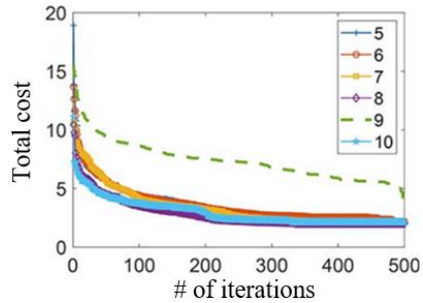

Figure 12. The convergence of the s second group (α=0.4, β=0.6)

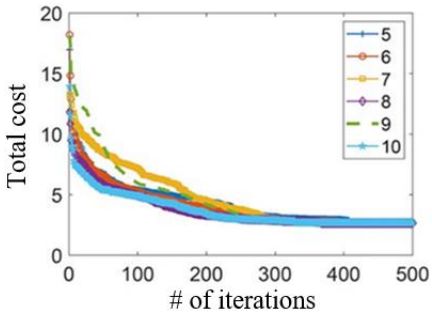

Figure 13. The convergence of the s third group (α=0.5, β=0.5)

6.3 Convergence evaluations for image processing tasks

In this subsection, the convergence of CTOS-DRL across various user groups was examined, with a special focus on image processing tasks that include classification, feature extraction, and texture analysis. For each group of users, the average total cost was calculated over a span of 500 iterations, capturing performance metrics relevant to image processing. After fitting, the results were outlined in Figures 11-13, leading to the following conclusions:

(1) As illustrated in Figures 11-13, the cumulative costs under the five strategies were found to nearly converge to a specific value after 500 iterations. This trend attests to a consistent performance across different parameters, even when dealing with computationally demanding image processing tasks. However, it should be noted that the rate of convergence might be slow in some cases, especially when the number of users is relatively high. Evidence of this phenomenon can be seen in the group with α=0.4 and β=0.6 consisting of 10 users, as explained in Figure 12.

(2) The overall speed of convergence was found to be closely related to the proportion of latency and energy consumption to the total cost, both critical aspects for image processing tasks. As shown in Figure 11, when the factor of energy consumption is high, convergence was observed around the 300th iteration. In stark contrast, when the proportion dropped to β=0.5, convergence was noticed around the 400th iteration. This sensitivity of convergence to these variables underscores the importance of balancing computational resources within edge computing environments, especially when faced with image processing tasks requiring immediate responses.

(3) Within the groups where α=0.3, β=0.7, α=0.5, β=0.5, and α=0.4, β=0.6, a significant decrease in total costs was recorded after approximately 50 iterations. This rapid convergence highlights the effectiveness of the CTOS-DRL method in adapting to diverse computational demands, validating its suitability for a wide range of image processing applications.

In summary, the findings demonstrate that CTOS-DRL effectively achieves and maintains quick convergence, an essential characteristic for real-time or near-real-time image processing tasks. The analysis included here not only confirms the robustness of the proposed model but also provides insights into its successful deployment across various scenarios requiring advanced image processing at the network's edge.

7. CONCLUSION

In this study, the crucial issue of task offloading for multiple users was explored, with a focus on image processing tasks such as object detection, segmentation, and feature extraction within an edge-cloud collaborative environment. The intricate nature of image processing requires careful examination of computational and communicative resource limitations, especially in situations requiring prompt responses.

The main findings and contributions of this study are outlined as follows:

(1) Optimization of Total Cost: An optimized offloading strategy was developed that simultaneously considers delays and energy consumption. The integration of these aspects enables efficient processing of image-related tasks, avoiding performance degradation or unnecessary costs.

(2) Innovative Offloading Strategy (CTOS-DRL): A novel CTOS, based on DRL, was proposed that incorporates a crowd updating mechanism within the collaborative edge cloud computing framework. This approach reduces the high complexity traditionally associated with nearby edge servers, offering a heuristic algorithm to identify feasible ones. Through the implementation of a fully connected neural network, the task offloading process was further refined, making it particularly suitable for image processing applications.

(3) Extensive Simulations and Experiments: The efficiency and effectiveness of the algorithms were confirmed through detailed simulations and practical experiments, specifically focusing on the context of image processing. Such experimental results provided empirical evidence for the superiority of the proposed approach in reducing the total cost, while maintaining high image processing performance at the edge.

(4) Implications for Image Processing at the Edge: The proposed CTOS-DRL was found not only to solve the task offloading problem but also to open new possibilities for the execution of complex image processing applications at the

network's edge. By leveraging advanced DRL techniques, a pathway for future innovations in the field was established, including potential extensions into video processing, augmented reality, and other visual computing domains.

In summary, the findings of this investigation represent a significant step forward in realizing the potential of edge computing for image processing tasks, offering a promising solution that expertly balances efficiency, cost-effectiveness, and robust performance. The methodologies and insights gleaned from this research provide a valuable foundation for further exploration and enhancement in this rapidly evolving field.

REFERENCES

[1] Luo, Q., Hu, S., Li, C., Li, G., Shi, W. (2021). Resource scheduling in edge computing: A survey. IEEE Communications Surveys & Tutorials, 23(4): 2131-2165. https://doi.org/10.1109/COMST.2021.3106401

[2] Mach, P., Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 19(3): 1628-1656. https://doi.org/10.1109/COMST.2017.2682318

[3] Wang, B., Wang, C., Huang, W., Song, Y., Qin, X. (2020). A survey and taxonomy on task offloading for edge-cloud computing. IEEE Access, 8: 186080-186101. https://doi.org/10.1109/ACCESS.2020.3029649

[4] Chen, M., Hao, Y. (2018). Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications, 36(3): 587-597. https://doi.org/10.1109/JSAC.2018.2815360

[5] Xia, Q., Lou, Z., Xu, W., Xu, Z. (2020). Near-optimal and learning-driven task offloading in a 5G multi-cell mobile edge cloud. Computer Networks, 176: 107276. https://doi.org/10.1016/j.comnet.2020.107276

[6] Guo, H., Liu, J., Lv, J. (2019). Toward intelligent task offloading at the edge. IEEE Network, 34(2): 128-134. https://doi.org/10.1109/MNET.001.1900200

[7] Upadhyay, S.K., Kumar, A. (2021). Early-stage brown spot disease recognition in paddy using image processing and deep learning techniques. Traitement du Signal, 38(6): 1755-1766. https://doi.org/10.18280/ts.380619

[8] Thayumanasamy, I., Ramamurthy, K. (2022). Performance analysis of machine learning and deep learning models for classification of Alzheimer's disease from brain MRI. Traitement du Signal, 39(6): 1961-1970. https://doi.org/10.18280/ts.390608

[9] El Rejal, A.A., Nagaty, K., Pester, A. (2023). An end-to-end CNN approach for enhancing underwater images using spatial and frequency domain techniques. Acadlore Transactions on AI and Machine Learning, 2(1): 1-12. https://doi.org/10.56578/ataiml020101

[10] Keshavarznejad, M., Rezvani, M.H., Adabi, S. (2021). Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Computing, 24: 1825-1853. https://doi.org/10.1007/s10586-020-03230-y

[11] Zhang, Y. (2023). Information acquisition method of tomato plug seedlings based on cycle-consistent adversarial network. Acadlore Transactions on AI and Machine Learning, 2(1): 46-54. https://doi.org/10.56578/ataiml020105

[12] Wu, H., Wolter, K., Jiao, P., Deng, Y., Zhao, Y., Xu, M. (2020). EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. IEEE Internet of Things Journal, 8(4): 2163-2176. https://doi.org/10.1109/JIOT.2020.3033521

[13] Yudhana, A., Riadi, I., Suharti, S. (2022). Network forensics against volumetric-based distributed denial of service attacks on cloud and the edge computing. International Journal of Safety and Security Engineering, 12(5): 577-588. https://doi.org/10.18280/ijsse.120505

[14] Moussa, M., Douik, A. (2021). Synthesis and comparison of improved edge detection technique based on metaheuristic and intelligent algorithm optimization. Traitement du Signal, 38(6): 1613-1622. https://doi.org/10.18280/ts.380605

[15] Wang, H.Y. (2020). Three-dimensional image recognition of athletes' wrong motions based on edge detection. Journal Européen des Systèmes Automatisés, 53(5): 733-738. https://doi.org/10.18280/jesa.530516

[16] Tunga, H., Kar, S., Giri, D. (2022). Intrinsic profit maximization of the offloading tasks for mobile edge computing with fixed memory capacities and low latency constraints using ant colony optimization. Mathematical Modelling of Engineering Problems, 9(3): 668-674. https://doi.org/10.18280/mmep.090313

[17] Qu, G., Wu, H., Li, R., Jiao, P. (2021). DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Transactions on Network and Service Management, 18(3): 3448-3459. https://doi.org/10.1109/TNSM.2021.3087258

[18] Hou, W., Wen, H., Song, H., Lei, W., Zhang, W. (2021). Multiagent deep reinforcement learning for task offloading and resource allocation in cybertwin-based networks. IEEE Internet of Things Journal, 8(22): 16256-16268. https://doi.org/10.1109/JIOT.2021.3095677

[19] Chen, X., Liu, G. (2021). Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks. IEEE Internet of Things Journal, 8(13): 10843-10856. https://doi.org/10.1109/JIOT.2021.3050804

[20] Sun, M., Bao, T., Xie, D., Lv, H., Si, G. (2021). Towards application-driven task offloading in edge computing based on deep reinforcement learning. Micromachines, 12(9): 1011. https://doi.org/10.3390/mi12091011

[21] Li, Q.P., Zhao, J.H., Gong, Y. (2019). Computation offloading and resource management scheme in mobile edge computing. Telecommunications Science, 35(3): 36-46. https://doi.org/10.11959/j.issn.1000-0801.2019060

[22] Guo, F., Zhang, H., Ji, H., Li, X., Leung, V.C. (2018). An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Transactions on Networking, 26(6): 2651-2664. https://doi.org/10.1109/TNET.2018.2873002