Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.062989

ARTICLE

Intelligent Management of Resources for Smart Edge Computing in 5G Heterogeneous Networks Using Blockchain and Deep Learning

Mohammad Tabrez Quasim[1,*], Khair Ul Nisa[1], Mohammad Shahid Husain[2], Abakar Ibraheem Abdalla Aadam[1], Mohammed Waseequ Sheraz[1] and Mohammad Zunnun Khan[1]

[1]Department of Computer Science and Artificial Intelligence, College of Computing and Information Technology, University of Bisha, Bisha, P.O. Box 551, Saudi Arabia

[2]College of Computing & Information Sciences, University of Science and Technology, Ibri, 516, Oman

*Corresponding Author: Mohammad Tabrez Quasim. Email: tabrezquasim@gmail.com

Received: 31 December 2024; Accepted: 28 April 2025; Published: 09 June 2025

ABSTRACT: Smart edge computing (SEC) is a novel paradigm for computing that could transfer cloud-based applications to the edge network, supporting computation-intensive services like face detection and natural language processing. A core feature of mobile edge computing, SEC improves user experience and device performance by offloading local activities to edge processors. In this framework, blockchain technology is utilized to ensure secure and trustworthy communication between edge devices and servers, protecting against potential security threats. Additionally, Deep Learning algorithms are employed to analyze resource availability and optimize computation offloading decisions dynamically. IoT applications that require significant resources can benefit from SEC, which has better coverage. Although access is constantly changing and network devices have heterogeneous resources, it is not easy to create consistent, dependable, and instantaneous communication between edge devices and their processors, specifically in 5G Heterogeneous Network (HN) situations. Thus, an Intelligent Management of Resources for Smart Edge Computing (IMRSEC) framework, which combines blockchain, edge computing, and Artificial Intelligence (AI) into 5G HNs, has been proposed in this paper. As a result, a unique dual schedule deep reinforcement learning (DS-DRL) technique has been developed, consisting of a rapid schedule learning process and a slow schedule learning process. The primary objective is to minimize overall unloading latency and system resource usage by optimizing computation offloading, resource allocation, and application caching. Simulation results demonstrate that the DS-DRL approach reduces task execution time by 32%, validating the method's effectiveness within the IMRSEC framework.

KEYWORDS: Smart edge computing; heterogeneous networks; blockchain; 5G network; internet of things; artificial intelligence

1 Introduction to Smart Edge Computing and Blockchain

Tremendous data transfer rates and ultra-low latencies have been predicted from the forthcoming 5G networks [1]. Several 5G technologies in varied areas, such as medical and transportation, are designed to fulfill distinct user quality of service (QoS) needs. Most of the time, these activities are resource-intensive and need heavy computing [2]. Utilities are limited in computational power and battery life because of physical restrictions. According to a recent report, the extreme conflict between computation-intensive programs and reserve-controlled UEs offers a barrier to the good efficiency of 5G apps.

As a solution to this problem, computational offloading has been proposed [3]. The computing power of UEs is considerably enhanced by outsourcing computer activities to a network system with a lot of resources.

Distributed units (DUs) and centralized units (CUs) in 5G connections are often responsible for transferring computational activities from the UE to the clouds to fulfill mass performance management.

Most of the core network activities are handled by the CUs, while the DUs handle real-time jobs. Toutefois, a lengthy transmission length between UE and distant cloud reduces computation performance. Edge computation (EC) is used to offer computer resources near UEs. Edge notes (ENs) are created by converting CUs into edge notes (ENs), which are then used to fulfill the rigorous, non-compromising delay requirements of apps [4]. Vulnerable machines (VM) are installed at edge nodes using virtualized resources, resulting in decreased energy usage. UEs, receive strong computational resources when 5G and EC are combined, which results in vastly improved quality of experience (QoE) for consumers.

As a side effect, the compute offloading procedure violates data integrity. Users are increasingly demanding data security. Therefore, it is important to ensure that data isn't taken or altered throughout the offloading process [5]. These approaches are not always suited for 5G networks due to the possibility of data loss, as most standard integrity-preserving routines rely on a genuine central authority. Whenever the centralized entity that administers the credibility system is hacked, it compromises the safety of the entire network.

A new paradigm for ensuring data integrity has emerged with blockchain, which is the basis of Bitcoin virtual money. A decentralized repository, blockchain maintains information in the form of blocks and, therefore, does not require a third party to be involved. Depending on the blockchain's verifiable and transparent data storage structure, data theft and loss are rare. Proof-of-Work (PoW) is another mechanism used by blockchain technologies to verify data security.

An attack on the blockchain and its verified block data is almost impossible due to its decentralized nature. As a result, blockchain technology can preserve important data integrity even during the offloading procedure. Diverse new apps are presented in the framework of 5G to provide consumers with a variety of possibilities. 5G applications are resource-hungry, and resource-constrained UEs can't provide improved QoE. EC is utilized to benefit the collective computational resources within a nation, ensuring that edge nodes (ENs) are accessible to user equipment (UEs) to overcome computational maximization challenges, there by dramatically enhancing the performance of applications [6,7].

The rapid adoption of IoT applications in 5G environments has led to increased demands for low latency and high-quality service, which traditional cloud computing approaches are often unable to meet due to inherent latency and resource limitations.

This paper addresses the urgent need for innovative solutions by introducing the Intelligent Management of Resources for Smart Edge Computing (IMRSEC) framework, which leverages blockchain technology and deep learning to optimize resource allocation and enhance application performance in edge environments.

The unbalanced distribution of computational resources helps maintain the system reliability of ENs, which results in a decrease in efficiency. Increased delay induced by the offloading method also hurts the QoE (Quality of Experience) of apps. Within the context of heterogeneous edge computing environments, IMRSEC has been presented as a methodology for addressing the challenges of application segmentation, resource allocation, and service cache location. The DS-DRL approach has also been proposed for the IMRSEC framework, which is composed of both a rapid schedule learning and a slow schedule learning procedure.

Through this framework, we aim to do the following:

- Minimize overall unloading latency,
- Improve resource utilization efficiency, and

- Enhance application caching strategies, all while maintaining robust security measures enabled by blockchain.

The remaining sections of the document are organized as follows. Section 2 explores related work on resource management for SEC with Blockchain. An Intelligent Management of Resources for Smart-Edge Computing (IMRSEC) framework has been proposed in Section 3 with the DS-DRL process. Section 4 consists of the analysis and findings obtained from the proposed model. Finally, the conclusion and possible studies have been outlined in Section 5.

2 Related Works on Resource Management Using Edge Computing and Blockchain

The 5G technology demands for low delay and data transmission have led to a substantial increase in the edge network concept. Content storing and computation at the edge of the cellular network serve a significant role in sharing edge capabilities on top of all that, it offers several functions, such as content cache and monitoring procedures, which may be used in wireless (cellular) or Wi-Fi networks. Authors have used MEC technology in this paper at the edges of the mobile network to improve the constraints of the radio interface currently in use. Given the increasing interest in reinforcement learning techniques, particularly Deep Reinforcement Learning (DRL), this section will focus specifically on recent works that have applied DRL strategies for task offloading in edge computing environments. Key studies that have yielded significant advancements in this area will be discussed, highlighting their methodologies, findings, and limitations. A context-aware system is presented for real-time collaboration with an underground communication system using MEC. Mobile edge management, cooperative video storage, and computation, as well as multilayer interfering cancellation, are three examples that demonstrate MEC's usefulness in 5G networks. A two-tier computer Wang and colleagues developed an offloading system in to reduce network delay by utilizing MEC in load balancing dramatically. Lu et al. in [7] suggested activity offloading on the edges of computational network power to minimize task length. MEC efficiency and effectiveness emphasized most of the research, although AI was also employed in certain studies to increase MEC's utility in edge computing.

Dai et al. [8] proposed an optimum offloading strategy based on Edge servers and a deep Q-learning technique to decrease infrastructure costs, assure offloading dependability, and increase system usefulness. Nevertheless, existing AI learning techniques pose a serious data protection risk to smartphones on edge networks. Fuzzy logic (FL) has been actively explored in the academic and business world, coupled with the rising concern for partly overcoming traditional lectures with centralized training programs.

As a forerunner in FL implementations, Google's board is one of the most popular applications on the web. An FL-based language method was created in [9] for a portable virtual keyboard for cell phones. FL application in many sectors is the subject of ongoing study. The FL method in wireless connections has subsequently been the subject of numerous research, which has focused on the benefits of this strategy.

In work by Zhang et al. [10], FL was utilized for safeguarding user privacy in the vehicle of the Internet of Things. Data leakage monitoring and smart data conversion are included in the mitigating option. To improve the UE selection and supply chain method, Hard et al. As part of the implementation process, this architecture also aims to change system efficiency by resolving the FL gradient descent. A protocol for controlling user selections using the FL method is based on their resource's circumstances, described in [11]. The scholars of [12,13] developed a framework of FL over wireless connections that captures two tradeoffs, combining computational power latencies and energy usage to improve FL's operations through a control system.

Research by Weng et al. [14] proposed DeepChain as an incentive-based decentralized architecture to accomplish three objectives during collaborative training: secrecy and equality. Using blockchain digital signatures and encryption barbarians, DeepChain is suggested to respect the safety of local settings and

assure the traceability of training processes. In [15], the authors proposed a decentralized incentive structure based on blockchains and formal verification to establish reliable security settings. They presented a secure P2P data-sharing platform for sensor communications that exploited payment systems and applied a vehicle reputation strategy to increase system performance. The study involves a three-weight objective and logical framework.

Several studies have also investigated the combination of blockchain and artificial intelligence in mobile devices, although further research is needed. A framework for a next-generation cellular network has been suggested by Firdaus and Rhee [15] by using Blockchain and AI to build a safe and smart distribution control system. Their design also presented four use instances: adjusting the frequency, information storage, market making, and compute offloading. A border-sharing method is used in this study to develop a safe and effective edge capacity scheduling method. As a result of utilizing a cross-domain sharing approach, this study proposes a paradigm for edge intelligence on 5G networks and beyond by combining blockchain and AI.

Table 1 provides a detailed comparison between previous research and the proposed work, highlighting key differences, improvements, and contributions made in this study.

Table 1: Comparison between previous research and the proposed work

Study	Methodology	Key findings	Limitations
Wang et al. (2021) [16]	Proposed a two-tier offloading system using MEC	Reduced network delays, improved load balancing	Inefficient resource allocation
Lu et al. (2020) [7]	Activity offloading strategies at the edge	Enhanced computational efficiency by minimizing task lengths	Enhanced computational efficiency by minimizing task lengths
Firdaus and Rhee (2021) [15]	Combined Blockchain and AI for resource management	Developed a secure distribution control system	Limited exploration of real-time decision-making in heterogeneous networks
Proposed IMRSEC Framework	Integrates DS-DRL for optimal resource management	Minimizes unloading latency and optimizes resource allocation	Still requires validation in extensive real-world scenario

Current studies on edge computing resource management focus on techniques like deep Q-learning and fuzzy logic but often overlook their integration with advanced frameworks in heterogeneous networks. Most approaches either optimize offloading efficiency or rely on centralized models, raising privacy concerns. Additionally, research on multi-dimensional QoE factors in real-time applications remains limited. A significant gap exists in developing a holistic framework that integrates application segmentation, resource allocation, and service cache while ensuring strong privacy. The proposed IMRSEC framework, emphasizing how it seeks to overcome the identified deficiencies and contribute to the field.

In addition to DRL approaches, it is vital to consider other AI-based solutions that have been proposed for resource management in edge computing. This includes techniques such as supervised learning, federated learning, and heuristic algorithms that optimize resource allocation and enhance system performance.

Notable frameworks and their impact on minimizing latency and improving Quality of Experience (QoE) will be addressed. This broader perspective will provide a well-rounded understanding of how AI continues to shape the future of edge computing.

3 Intelligent Management of Resources for Smart Edge Computing (IMRSEC) Framework

An intelligent management of resources for a smart edge computing framework has been proposed in this section. The structure of the IMRSEC framework was initially outlined. This will be followed by a presentation of IMRSEC's network model and service cache concept. The proposed IMRSEC's offloading mechanism has finally been developed using DS-DRL.

3.1 Framework of IMRSEC

Fig. 1 displays the architecture of assorted edge computing networks using blockchain and AI. By improving the three main key features of SEC-HN, i.e., computational unloading, distribution of resources, and service cache deployment, the IMRSEC may be implemented and operated. Since edge devices may unload their compute-intensive activities to their adjacent accessible edge servers, load balancing plays a vital role in the SEC-HN architecture. It can also reduce the power consumption of edge devices by transferring decisions in real time. To accomplish real-time decision-making, the computing offloading process is augmented using an artificial intelligence method. The SEC-HN might also be disabled by jamming attacks or sniffer attacks during computation offloading. To provide effective and dependable collaboration between SEC-HN networks, protected blockchain communication (including encoding, addressing, and so on) is pragmatic to the SEC-HN networks.

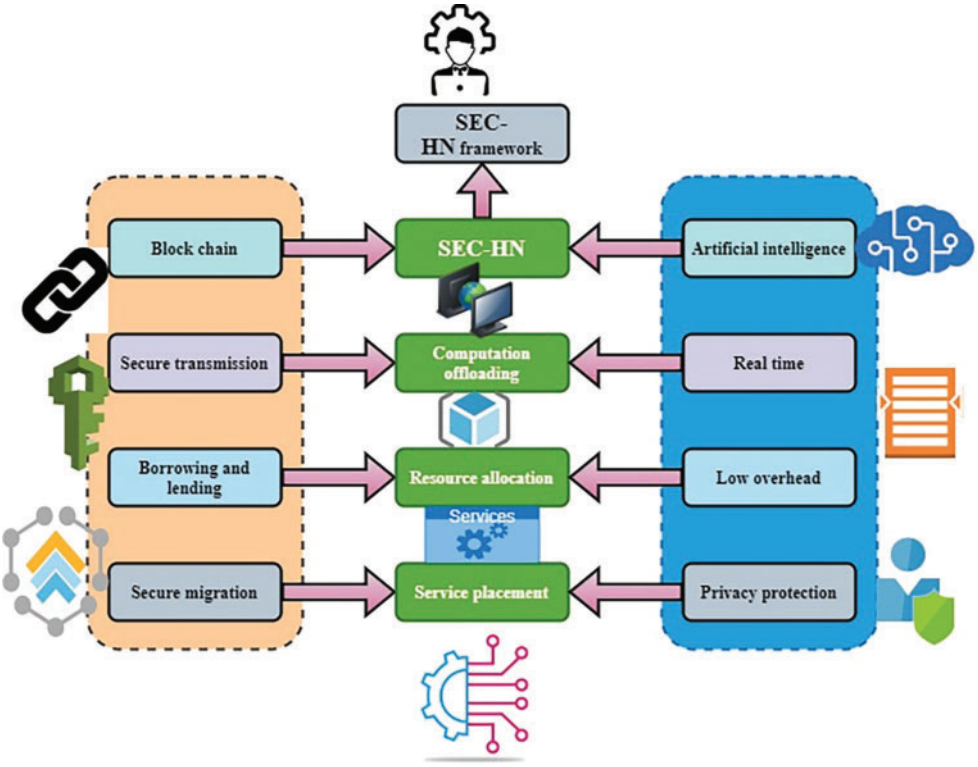

Figure 1: Architecture of heterogeneous edge computing networks using blockchain and AI

Due to the dispersed nature of SEC-HN, it is challenging to handle computing, connectivity, or storage resources. AI-based automated resource provisioning method for the SEC-HN is meant to achieve this goal. The edge servers should also explore a dynamic coordination mechanism based on blockchain (including borrowing and lending of resources). Furthermore, a technique for training models based on federated learning has been developed to preserve the privacy of edge devices. In the transfer of operations across many edge servers, blockchain-based service migration is explored to provide security.

Fig. 2 shows the suggested IMRSEC framework. The user level, the data level, and the control level are all part of the system. In Fig. 2, the user level comprises edge devices (ENs) with computational offloading needs. The data level has I) A distant web server; and II) Edge servers placed close to ENs. It may also tackle overloading assault by storing defective packets using blockchain or coping with DOS attacks by caching packages in the stream. A massive MIMO system controller is placed at the micro base station (MBS). Control capabilities (i.e., surveillance, computational unloading activities, resource distribution techniques, and service cache deployment) have been unified at the control level. At this level, blockchain-based network monitoring is used to avoid harmful activities (e.g., DDoS attacks, packet saturating).

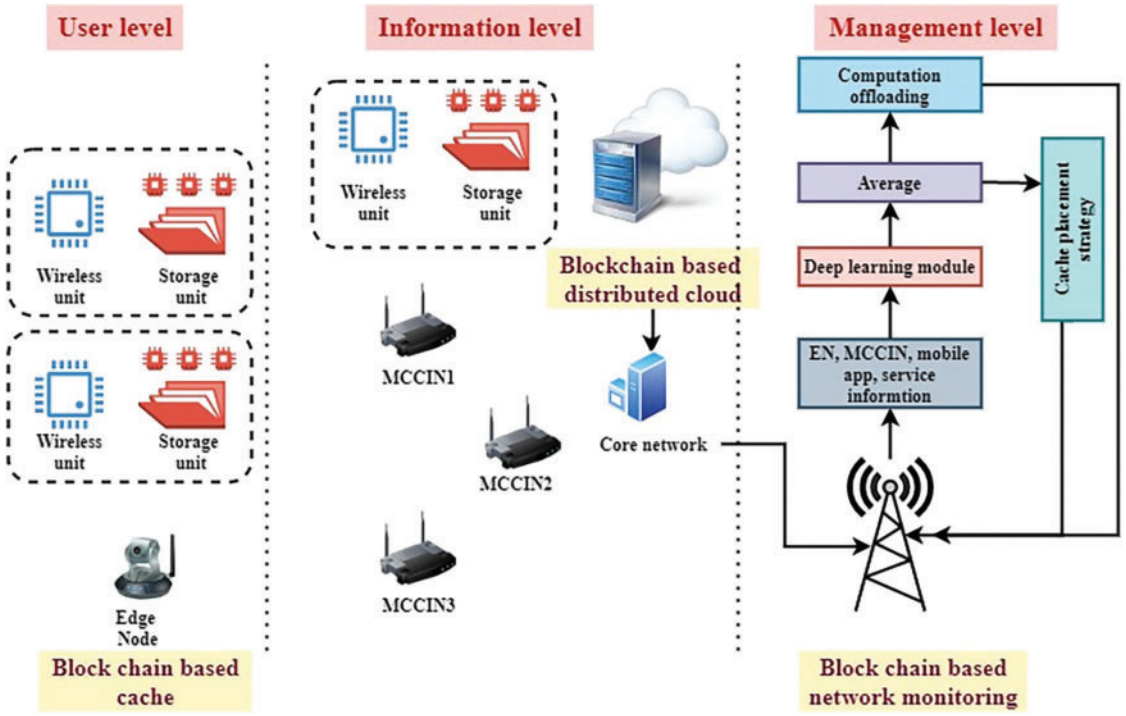

Figure 2: Architecture of the proposed IMRSEC system

Controllers can store records of statistics about Edge Nodes, Edge Servers, Services, and Applications to maximize system settings upon request. Details such as computing and communication capacity are included in the EN information list. The Edge Server knowledge list keeps track of the computing, storage, communication capacity, and fronthaul latency for each edge server. The current service cache placement method for the edge servers and ENs is listed in the service information list. In addition to the program type, the needed service category, the information quantity, and the required processing capacity are all included in the application data.

An EN typically transmits edge data and requests data to the adjacent edge server when it has computational unloading needs. Next, the edge server combines the data from various ENs and the edge server and sends it to the controller. Through deep learning and federated learning components, the organizer accepts and trains the information received. For edge servers and edge nodes (ENs), the controller must make joint computational unloading and reserve distribution choices rapidly. For edge servers, the controller must make service cache placement strategies on a slower schedule.

3.1.1 System Model

The edge processors of the MEC are referred to as microcell cloud-improved eNode (MCCIN). In addition, the ENs and MCCIN are connected through elevated, high-bandwidth lines. MCCIN is provided with higher computing resources relative to edge nodes. In Fig. 2, it has been proposed that the SEC-HN framework functions in an ultra-dense edge computing network. Such a network consists of an edge computing server located in the cloud, an edge computing base station, a set of MCCINs, and a group of ENs within the coverage of MCCIN.

The MCCIN of MEC processors have been denoted as $O = \{1, 2, \ldots, O\}$ and the ENs are referred to as $N = \{1, 2, \ldots, N\}$ found inside the exposure area of MCCIN given as $o(o \in O)$. Also, the ENs are denoted as $(n \in N)$ consist of Y_n Processing units with Z core processors. The storage unit of MCCIN is given by D_t. The processing frequency of MCCIN and EN are given by F given by F_t and F_v, respectively.

In the proposed IMRSEC framework, offloading has been done by creating a wireless connection with an MCCIN or D2D link with some other EN. D2D and mobile phones use orthogonal frequency division multiple-access (OFDMA). There is no interference between the N ENs in the scope of $'o'$ MCCIN since they are segregated in the spectral domain. To keep things simple, a subcarrier-based fixed channel assignment (FCA) method has been used in the N ENs. For an MCCIN, there are L accessible subcarriers to service ENs. Each subcarrier has a bandwidth of BW.

3.1.2 Modeling of the Wireless Link

Let's suppose for the duration of one EN scheduling period that the position of ENs is fixed while the communication networks between ENs and MCCIN remain steady. Due to ENs movement, they may alter throughout various scheduled times. An uplink data rate $s_{n,o}^{UL}$ (in bits per second) between EN $n(n \in N)$ and MCCIN $o(o \in O)$, over an additive white Gaussian noise channel (AWGN), may be formulated:

$$s_{n,o}^{UL} = L_n^o \times BW \times \left(\frac{q_n |H_{n,o}^{UL}|}{\varphi h_{UL} e_{n,o}^{\rho} N o_0} + 1 \right) \tag{1}$$

L_n^o signifies the quantity of OFDM subcarriers allocated by MCCIN to ENs. The bandwidth of each subcarrier is given by BW. q_n is the uplink energy for the transmission of data to EN. The coefficient of Rayleigh fading environment and the mark error rate is given by h_{UL} and $H_{n,o}^{UL}$, respectively, in the uplink between EN $n(n \in N)$ and MCCIN $o(o \in O)$. φ is the signal power needed to achieve the marked error rate with the 16-QAM modulation scheme. The distance between MCCIN and EN in the uplink is given by $e_{n,o}^{\rho}$, with propagation loss of ρ.

It is then put into a matrix of uplink rate weights. $S_o^{UL} = \left[s_{n,o}^{UL} \right]_{1 \times N}$. As a side note, the ENs reserve a portion of their available bandwidth for broadcasting pilot signals in the extremely dense edge computing network. When receiving pilot signals, MCCIN can estimate the uplink channels. As a final step, through increased fronthaul connections, MCCINs transmit estimated channels to the controllers. In reality, the channel's improvement may be predicted quite well (with minor arbitrary noise). Deep reinforcement

learning (DRL) may produce a quite reasonable estimation of the general reality even if the channel state is not perfectly approximated. This is owing to the tremendous learning capabilities of deep learning across vast accumulated learning instances.

3.1.3 Modelling of D2D Link

Encryption units (ENs) in the proposed IMRSEC system can dispatch computation to another EN in the vicinity over a D2D link. Edge nodes can unload processing to one another if their proximity is smaller than a predefined threshold, S_e.

Device to the Device data rate $s_{k,l}^{D2D}$ (in bits per second) between two ENs, over a wireless channel is given by:

$$s_{k,l}^{D2D} = L_n^o \times BW \times \left(\frac{q_k |H_{k,l}^{D2D}|}{\varphi h_{D2D} e_{k,l}^{\rho} No_l} + 1 \right) \tag{2}$$

q_k is the D2D energy for the transmission of data among ENs. The channel coefficient and the bit error rate are given by h_{D2D} and $H_{k,l}^{D2D}$, respectively, in the D2D link among ENs. φ is the signal power needed to achieve the error rate. The distance between two ENs in the D2D link is given by $e_{k,l}^{\rho}$, with propagation loss of ρ. The amount of OFDM subcarriers allocated by MCCIN to ENs is L_n^o. BW is the bandwidth of each subcarrier. D2D link matrix with rate weights given as $S_l^{D2D} = \left[s_{k,l}^{D2D} \right]_{N \times N}$.

The objective of this research has been on ENs' uplink energy usage, with the forward link and power ingesting on the MCCIN side being ignored. However, the magnitude of data yields transmitted back in the downlink for most computation-intensive applications is significantly lower than the information scope of supplies in the uplink (the proportion could be as small as 0.03125). Reverse link communication, on the contrary, consumes around 4.9 times the amount of energy that downlink reception consumes. As a result, the forward reception channel uses far less energy than reverse channel communication links. Furthermore, the MCCIN is often implemented in permanent regions with adequate power generation.

3.1.4 Modeling of Offloading Tasks

As a result, a computing job P is broken into numerous smaller jobs. Let $P = (W, E)$ O, where W indicates the collection of subtasks and E defines the data interdependence among them. Let W denote the number of minor jobs in the task P, where, $W = |w|$. It is possible that each micro-job $W(W \in w)$ be is dispatched to a fog server, unburdened to an end device, or distributed regionally among the ENs. Due to this, interdependence has a significant impact on data delivery and processor unloading and thus cannot be disregarded.

The Fig. 3 illustrates the inputs and outputs of the Generic Deep Neural Network (DNN) in the context of the MEC offloading framework. Specifically, the inputs include:

1. **Inputs from the 5G Network:** The DNN receives various inputs from the 5G network that inform the offloading decision-making process. These can include:
   - **Network Conditions:** Real-time data regarding latency, bandwidth, and jitter within the network.
   - **Device Characteristics:** Information pertaining to the computational and energy capabilities of the User Equipment (UE).
   - **Task Specifications:** Details on the computational requirements, deadlines, and data dependencies of the tasks to be offloaded.

2. **Outputs to the Offloading Framework:** Based on the processed inputs, the DNN produces outputs that guide the decision-making process for optimal offloading, which may include:
   - **Offloading Decisions:** Recommendations on whether to offload specific tasks to an edge server or to execute them locally on the UE.
   - **Resource Allocation Strategies:** Insights on how to allocate available resources efficiently among different tasks and edge servers to maximize quality of experience (QoE).
   - **Task Scheduling Recommendations:** Guidance on the timing and sequencing of task executions to minimize latency and energy consumption.

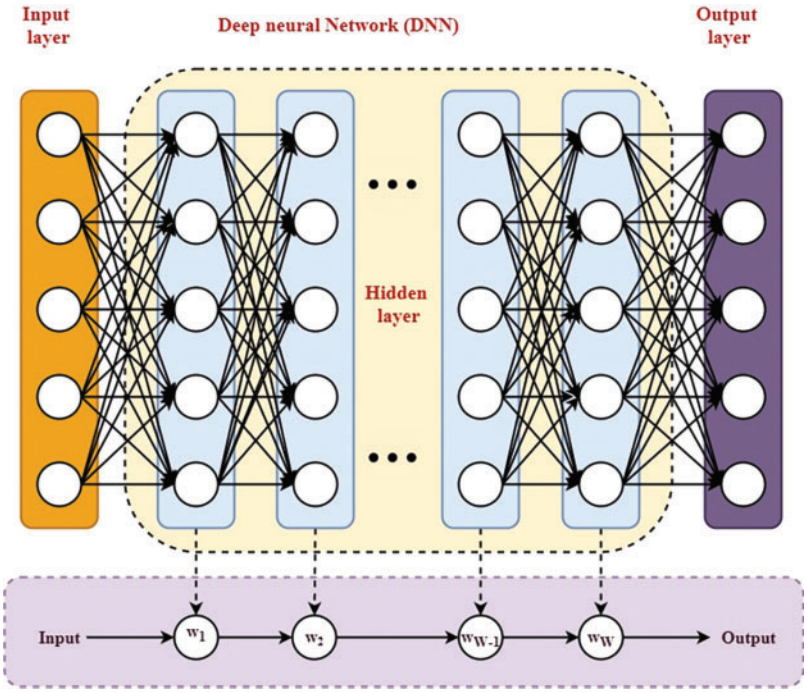

Figure 3: Representation of task modeling using DNN

Fig. 3 depicts the representation of task modeling using DNN. As a basic guideline, there are three main types of dependence models: sequential, concurrent, and generic. As an example, consider the job of partitioning a DNN model with chronological dependence. Fig. 3 shows a way to divide a DNN training phase (the computing job P) into numerous subtasks (the hidden layers of DNN). Inference may be accelerated by dynamically outsourcing resource-intensive subtasks to the edge server via DNN model splitting. The proposed system includes a sequential dependence task model, which involves transferring computational tasks from resource-constrained devices to edge servers or the cloud, and requires intelligent decision-making under dynamic and multi-dimensional constraints. Deep Neural Networks (DNNs) are uniquely suited to addressing these challenges for the following reasons:

(1) First, the task of offloading involves interdependent variables such as fluctuating network latency, heterogeneous device capabilities, server load states, energy budgets, and task-specific requirements (e.g., computation/communication trade-offs). Traditional rule-based or linear models struggle to capture these nonlinear interactions.

(2) Second, offloading decisions must be made in milliseconds to avoid degrading user experience in latency-sensitive applications (e.g., augmented reality, autonomous systems). Once trained, DNN inference is highly efficient, enabling real-time predictions even on edge devices.
(3) Third, DNNs can integrate online learning or transfer learning to adapt to new environments without full retraining. For example, reinforcement learning (RL)-based DNNs can iteratively refine offloading policies by interacting with the environment, outperforming static heuristic algorithms.
(4) Fourth, offloading requires balancing competing objectives: minimizing latency, energy consumption, and cost while maximizing reliability.
(5) Fifth, the new edge systems generate vast multimodal data. DNNs can fuse these heterogeneous data streams using hybrid architectures (e.g., CNNs for spatial data, RNNs for temporal patterns) to extract actionable insights. Classical models often lack this flexibility.

Therefore, we can conclude that DNNs provide a robust framework for task offloading by combining adaptability, scalability, and the ability to model complex, real-world dynamics. Their capacity to process high-dimensional data and optimize multi-objective trade-offs makes them superior to traditional approaches, particularly in large-scale, dynamic edge environments. As edge computing evolves, DNNs will remain critical for enabling efficient, intelligent offloading strategies.

3.2 Offloading Technique in the Proposed IMRSEC

The IMRSEC framework's computation offloading process is broken down into three steps in this paper: (i) service cache deployment, (ii) computational unloading choices, and (iii) allocating resources. Due to the MCCIN's limited storage capacities, an effective service storing deployment policy for the MCCIN is necessary to reduce the provision detachment between ED and the MCCIN that serves them. Nevertheless, unlike a distant cloud with abundant computing and storage resources, a single MCCIN with restricted resources can only cache a portion of the services at a time. The major issues for application storage in IMRSEC are: (i) how to evaluate the fame of amenities and store the prevalent facilities, and (ii) how to poise the tradeoff between many facilities and the limited loading space of MCCIN.

A fine-grained request segmentation technique has been utilized to minimize job delivery time and enhance QoS for ENs. Using fine-grained service segmentation, a mobile application is divided into many subtasks and dynamically offloaded. Also, consider that smartphones can benefit from real-time and dynamic partitioning through computation unloading, which is crucial in SEC-HN systems. MCCIN must now assign computing, connectivity, and storage infrastructure to the offloaded chunks to complete the offloaded activities. MCCIN must allocate system resources to process the offloaded components to handle the offloaded elements of mobile apps. It will also reduce the system's resource utilization and enhance the efficiency of edge nodes.

To this purpose, it seems appropriate to maximize the resource cache placement, application partitioning, and service provision for the IMRSEC in conjunction with the other optimizations. It can be calculated the overall processing time. $U^{pro}(t)$ of the smaller jobs as follows:

$$U^{pro}(t) = \sum_{j=1}^{m} u_j^{pro}(t) \tag{3}$$

u_j^{pro} is the processing time for a particular job j to be carried out. An EN would either perform the sequence of operations internally or unload them to another EN over a D2D connection or an MCCIN through a wireless uplink or to a distant cloud server through the radio uplink and high-speed

front-end processing. It has also been shown that the unloading choices occur in discrete time frames $U = \{1, 2, 3, \ldots, u\}$. MCCINs are assigned to each EN in the IMRSEC architecture.

The proposed pseudocode represents the flow of our algorithm within the Intelligent Management of Resources for Smart Edge Computing (IMRSEC) framework.

Pseudocode for the IMRSEC Algorithm 1:

Algorithm 1: IMRSEC_Resource_Management

Input: Edge Node (EN) requests, Service Cache, Resource Availability
Output: Optimized Resource Allocation, Offloading Decisions

1. *Initialize:*
   a. Load Service Cache information
   b. Load Edge Node processing requirements
   c. Identify resource availability at Edge and Cloud servers
2. *For each Edge Node (EN) in EN_requests do:*
   a. Evaluate processing capacity and latency requirements
   b. Perform Fine-grained Task Segmentation (FTS)
      i. Divide mobile applications into subtasks
      ii. Assign appropriate computational resources to each subtask
3. *For each subtask in FTS do:*
   a. Determine optimal allocation using DS-DRL:
      i. Input current state (resource availability, task status)
      ii. Calculate action (resource allocation decision)
      iii. Update system based on reward feedback
   b. Store offloading decisions in Resource Allocation Table
4. *Deploy Service Cache:*
   a. Based on current requests, optimize service cache placement
   b. Ensure that frequently accessed services are readily available
5. *Monitor and Adjust:*
   a. Continuously monitor Edge Node performance and resource usage
   b. Adjust resource allocations dynamically based on real-time data
6. *End Algorithm*

3.3 Proposed DS-DRL Technique

An optimization issue has been defined in this section for the combined segmentation of applications, reserve apportionment, and application cache appointment in the IMRSEC framework. Long-term job implementation and organization resource utilization are jointly minimized in this objective function (i.e., calculation, connectivity, and loading usage).

Fig. 4 shows the architecture of a unique dual schedule deep reinforcement learning strategy to train the choice manager. In reinforcement learning (RL), an individual may help train operations regularly, monitor which actions result in the most reward, and automatically modify their approach to achieve the best policy. In the literature, Q-learning, for example, is a frequently used model-free RL method. By estimating the Q-function, a decision agent can arrive at the optimum policy in Q-learning (well-defined as the state-action purpose). To evaluate the values of state-action pairings, the Q-function uses samples collected during

interactions with the environment. Nevertheless, because RL must investigate and acquire information about the entire environment, obtaining the optimum policy takes some time. As a result, RL is inappropriate and inadmissible to communication organizations, particularly in the IMRSEC's HN environment.

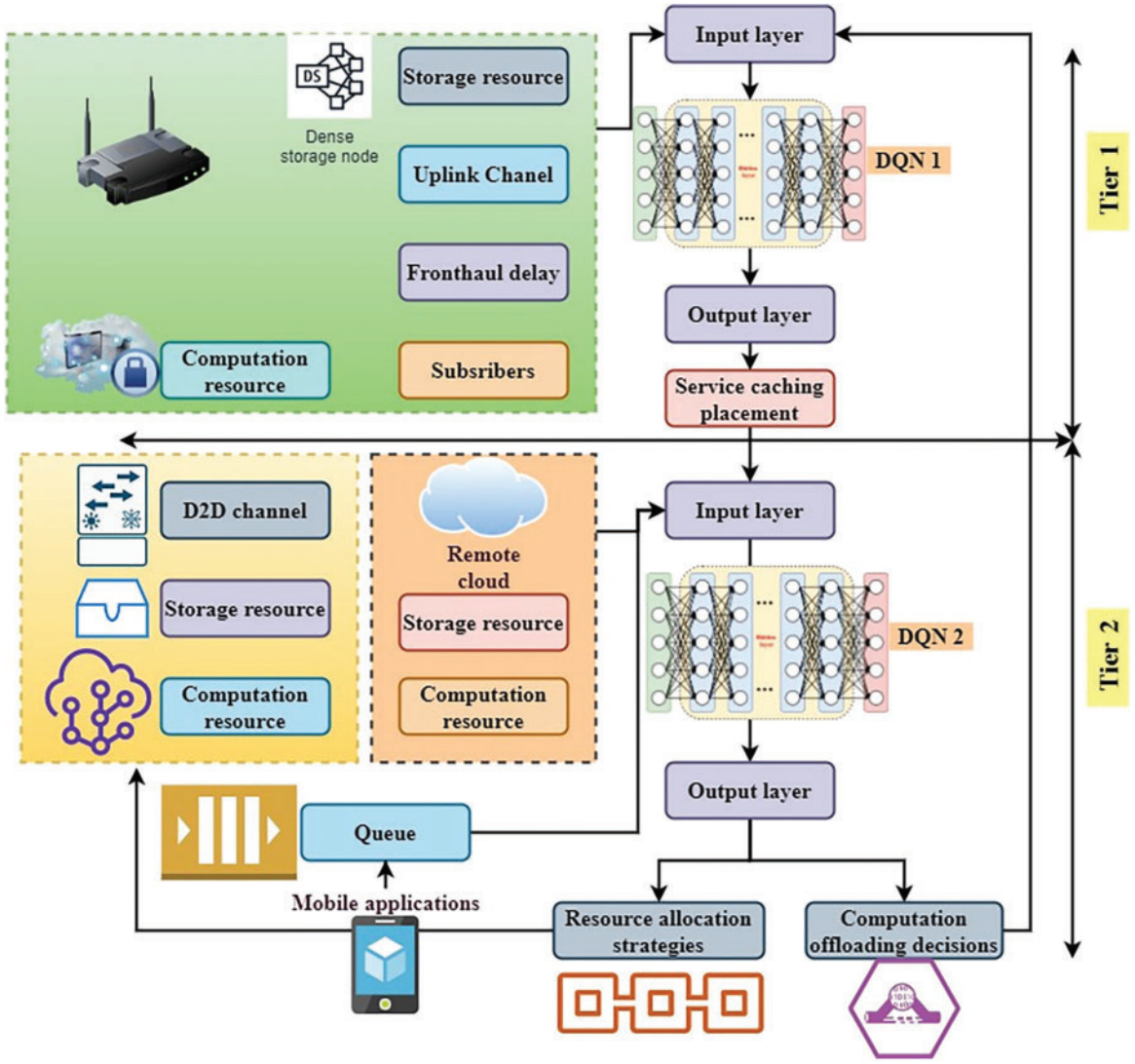

Figure 4: Architecture of dual schedule deep reinforcement learning (DS-DRL) technique

Supervised learning is offered as a cutting-edge approach for overcoming RL's limitations mentioned above, and it generates a new study area called DRL. DRL uses DNNs to train the knowledge model, which enhances learning speed and reinforcement learning performance. To resolve tiny state space and accomplishment space difficulties in Q-learning, the Deep Q-Learning (DQL) method is proposed, which combines deep learning with artificial intelligence. DQL uses a Deep Q-Network (DQN) rather than the Q-table of Q-learning, resulting in a substantial increase in the learning process. The program segmentation, asset, and application caching placement techniques in the IMRSEC architecture have varying delay sensitivity.

To this aim, a unique DS-DRL technique has been developed to optimize the problems described above in two distinct timescales jointly. As illustrated in Fig. 4, a two-tier structure DQN mediator for the DRL has been defined, which acts on two dissimilar timeframes. The informant's lowest tier delays policy debates (such as program segmentation and resource requirements) rapidly. In contrast, the top-level delays insensitive decisions (such as service cache placement technique) over a long timeframe.

The uploaded data is then received by MCCIN, which attempts to simulate the work queue. The work queue status and the appropriate resource state make the system state of MCCIN o at each decision period u, which are provided as follows:

$$T_o(t) = \{L(t), S_o^{UL}, S_o^{D2D}, R_{AC}(t), R_{EN}(t), H_o(t), Y_o(t), Z_o(t), TL_o(t) \tag{4}$$

The amount of OFDM subcarriers allocated is given as $L(t)$ at a particular period t. Uplink data rate (in bits per second) between EN and MCCIN is S_o^{UL}. S_o^{D2D} is the data rate rank matrices of the D2D link. $R_{AC}(t)$, and $R_{EN}(t)$ denote the application caching state of MCCIN and EN at a particular period t, respectively. $H_o(t)$ is the present status of the job queue. The current computation of resources has been provided as $Y_o(t) = \{y_1(t), y_2(t), \ldots, y_N(t)\}$. $Z_o(t)$ is the present state of the processor given as $\{z_1(t), z_2(t), \ldots, z_Z(t)\}$. $TL_o(t) = \{r_w, w \in H_o(t)\}$ reflect the network of resources requested by the activity queue's smaller jobs.

The reward for using the application cache service has been given as:

$$S_d(t) = \omega 1[U^{pro}(t)/U^{reg}(t)] \tag{5}$$

where $U^{pro}(t) <= U^{reg}(t)$. $\omega 1$ is the scaling coefficient. $U^{pro}(t)$ and $U^{reg}(t)$ are the overall processing time and regional processing time of smaller tasks in the list, respectively, at an instant t. Now, the restricted processing time is given as follows:

$$U^{reg}(t) = \cup_{j=1}^{m} \frac{\rho_w \times e_{a,b}}{Y_n \times g_a}, b \in h_j(t) \tag{6}$$

ρ_w is the loss due to the propagation of information in HN. Y_n is the present computational resources used by EN. $h_j(t)$ indicates the channel coefficient. $m$is the quantity of MCCIN nodes in the proposed scheme. g_a is the current resources used by MCCIN in the queue. $e_{a,b}$ is the distance between MCCIN a and the EN b.

The goal of the application cache deployment is to reduce the ENs' protracted processing time and the MCCIN's memory utilization. So, raid reward function is given by:

$$S_d^{rapid} = \max\{S_d(t)\} = max\{\omega 1[U^{pro}(t)/U^{reg}(t)]\}, t = 0, \varphi t, 2\varphi t, \ldots \tag{7}$$

φ denotes the signal power required for uplink transmission. $\omega 1$, $U^{pro}(t)$ and $U^{reg}(t)$ are the scaling coefficient, overall processing time, and regional processing time, respectively. The first component of $S_d(t)$ is a standardized processing time index, and the primary limitation of $S_d(t)$ ensures that the overall execution time due to computational offloading is less than the period spent on regional processing. Furthermore, in $S_d(t)$, the second component provides a standardized memory consumption index, and the additional restriction of the reward function ensures that the overall storage utilization of the present service storing approach $R_{AC}(t)$ is less than the MCCIN's loading capabilities. The reward for slow application cache segmentation has been given as:

$$S_d^{slow} = S_{ac}(t) = min\{\propto_1 [U^{pro}(t)/U^{reg}(t)] + \propto_2 [\sum_{n=1}^{N} L_n^o(t)/L(t)]\} \tag{8}$$

$\propto_1$ and $\propto_2$ are the scaling coefficients. $L_n^o(t)$ is the number of frequency subcarriers allocated to ENs and MCCINs. $L(t)$ is the overall frequency allocation for HN. As illustrated in (8), the first term signifies a normalized implementation period index depicted in Eq. (7). The additional component in Eq. (8) indicates MCCIN's standardized connectivity resource utilization, which ensures that at the decision period (t), the number of spent subcarriers is fewer than the number of accessible subcarriers.

As a result, for the IMRSEC framework, a unique DS-DRL approach has been created, which comprises a fast schedule learning process and a slow scheduled learning experience. The major aim is to decrease total offloading delay and computing resources consumption by improving computation unloading, allocation of resources, and service cache deployment.

4 Results and Discussion

Computer simulations are used in this part to assess the performance of the projected DS-DRL system. The DS-DRL framework has been developed using the MATLAB-supported learning toolkit and deep learning toolbox. Initially, the test scenario was presented, followed by an exposition to the suggested IMRSEC benchmark techniques. The effectiveness of the proposed DS-DRL algorithm was then compared to baseline methods, and the simulation responses were recorded. Table 2 summarizes the values of parameters used in the simulation.

Table 2: Simulation parameters for the analysis of the proposed DS-DRL algorithm in the IMRSEC framework

Parameter	Value
Bandwidth BW	20 kHz
Number of MCCIN N	5
Number of EN O	20
Scaling coefficients for slow schedule $\propto_1$, $\propto_2$	0.33
Scaling coefficient for rapid schedule $\omega 1$	0.5
Rate of learning	0.002
Propagation loss ρ_w	[3500, 11,000] cycles/bit
current resources used by MCCIN in queue g_a	10^8 cycles/se
Distance between MCCIN a and the EN b $e_{a,b}$	[150, 550] KB

Table 2 shows the simulation parameters for the analysis of the proposed DS-DRL algorithm in the IMRSEC framework. The suggested DS-DRL dependent job execution strategy was then deployed and compared against the succeeding job implementation schemes, a reserve apportionment method, and a provision caching deployment strategy, namely:

- Regional Processing Approach (RPA): Regional implementation of all subtasks executed locally on the ENs.
- Edge Processing Approach (EPA): The ENs delegate all their duties to the MCCINs in their immediate vicinity for execution. It is worth noting that the needed service for EN $n's$ subtask w is not stored in a neighboring MCCIN. The specific task w is processed locally by EN n.
- Arbitrary Processing Approach (APA): This is an arbitrary computation divesting technique in which each subtask is offloaded to MCCIN, a cloud server, a neighboring EN, or stored regionally.
- Cloud Processing Approach (CPA): It offloads all sub-problems to a distant central server.

- Impartial Resource Provisioning Approach (IRPA): The EDs evenly distribute communication resources (available subcarriers).
- Publicity-based Service Cache Assignment Procedure (PSCAP): MCCINs cache the most important programs until memory space is reached.

Fig. 5 shows the mean processing time for various schemes in the proposed IMRSEC framework. This is because resource allocation and service cache placement policies do not influence RPA and CPA, respectively. The performance of our suggested DS-DRL has been evaluated with reference methods, as shown in the figure. It relates to an edge processing system with an impartial resource provisioning method and a publicity-based application cache deployment strategy and is part of the EPA-IRPA-PSCAP standard. An arbitrary processing approach with an unbiased allocation of resources method and publicity-based service caching deployment strategy is called APA-IRPA-PSCAP. ENs upload operations to neighboring MCCINs using CPA-IRPA, a cloud execution method with a fair resource distribution mechanism. DS-DRL beats other processing methods in terms of computation time reductions. Comparing the RPA, EPA-IRPA-PSCAP, APA-IRPA-PSCAP, and CPA-IRPA schemes to the proposed DS-DRL scheme, task execution time is reduced by 32, 15, 26, and 23 percent, respectively.

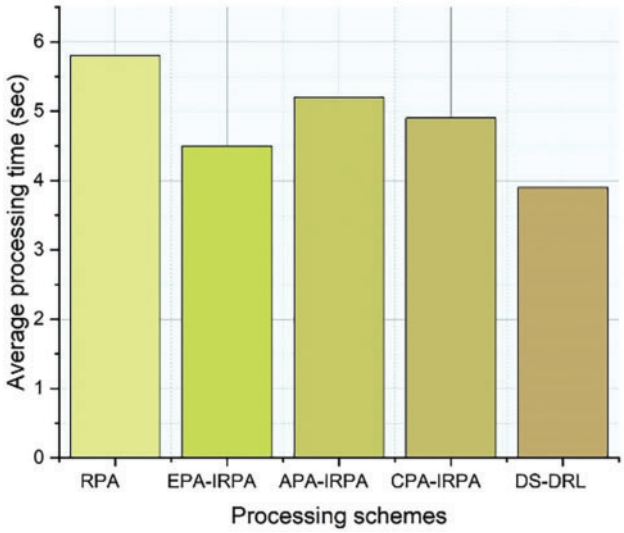

Figure 5: Mean processing time for several structures in the proposed IMRSEC framework

The improvement in the speed is due to the cache-hit effectiveness of the proposed framework. Also, Knowledge-based model training approach provides computational efficiency in the reinforced learning, thus improving overall speed as compared to existing approached (EPA, IRPA, and PSCAP approaches).

Average power consumption for several approaches in the proposed IMRSEC framework has been depicted in Fig. 6. It has been observed in Fig. 6 how much energy is consumed by the sub-problem throughout its processing. It is important to note that it has been considered only the energy usage on the edge devices. All sub-tasks are offloaded to a distant cloud platform and encoding and decoding devices only use power for information transfer. Despite the large end-to-end latency, the DS-DRL implementation period is extensive, as illustrated in Fig. 6. A comparison of EPA-IRPA-PSCAP shows that DS-DRL beats the existing schemes and delivers energy savings of 26%, 18%, and 11% relative to the execution as mentioned in baseline strategies.

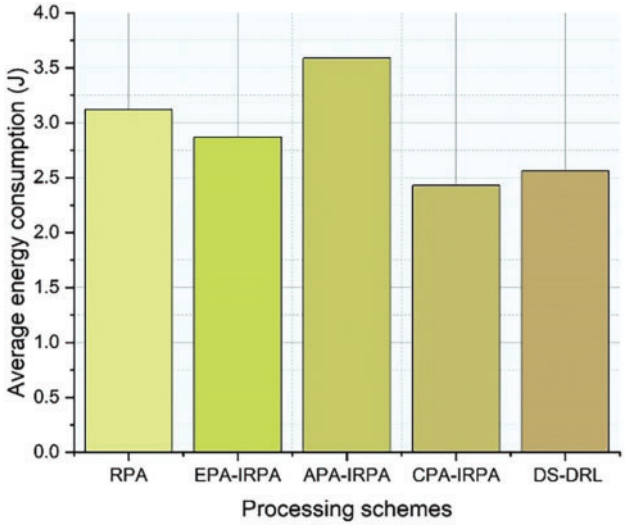

Figure 6: Average power consumption for several approaches in the proposed IMRSEC framework

As shown in Fig. 7, the performance of the suggested method in the SEC-HN scenario is evaluated by comparing task execution times for different EN densities. In terms of task execution time, the suggested DS-DRL beats the benchmark policies. When the task density increases from 1 to 10, the RPA has constant task execution time since it conducts regional execution and does not have any added communication overhead. The suggested DS-DRL and APA-IRPA-PSCAP processing times rise as EN density is increased. All ENs share a limited number of computing and communication resources within a single MCCIN coverage area. As the quantity of EN rises, the performance of the task execution time decreases. Installing extra MCCINs at the edge networks, particularly in specific hot spots, can resolve the error. A huge rise in operation processing time occurs when $N > 7$ for DS-DRL. Due to restricted computing capability, MCCINs can only support up to 7 ENs in the simulation. When $N > 7$, certain ENs are required to execute RPA.

Fig. 8 shows the cache hit probability for various cache deployment schemes using the proposed IMRSEC framework. As an example in Fig. 8a, FL points out that it offers a collaborative learning-based development tool. An administrator's queries from the last decision period have been used to train them. It is possible to make several data points. Because all services have predefined popularity indexes and PSCAP caches the most popular services, PSCAP has a set likelihood of cache hits. When the decision agent receives facility requirements from the ENs, the Unified scheme has a static cache hit probability from decision period 0 to 1500. So, as the range of episodes increases, the store hit likelihood. The stock hit efficiency of the Unified scheme is comparable to that of the proposed framework, which is a positive outcome. It is possible to preserve ENs' privacy by using the suggested federated learning-based model training technique.

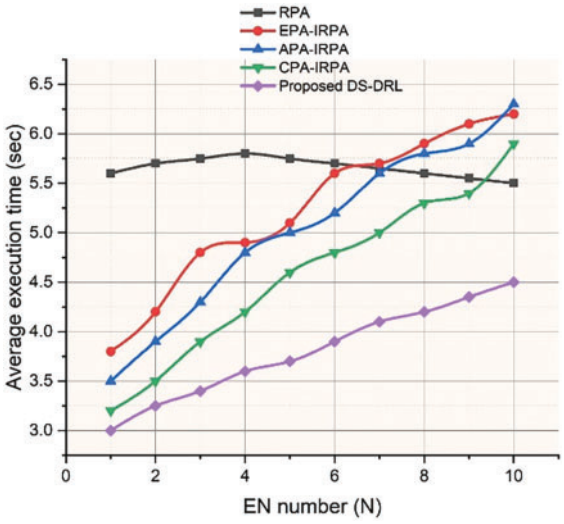

Figure 7: Average execution time for various edge nodes in the proposed IMRSEC framework

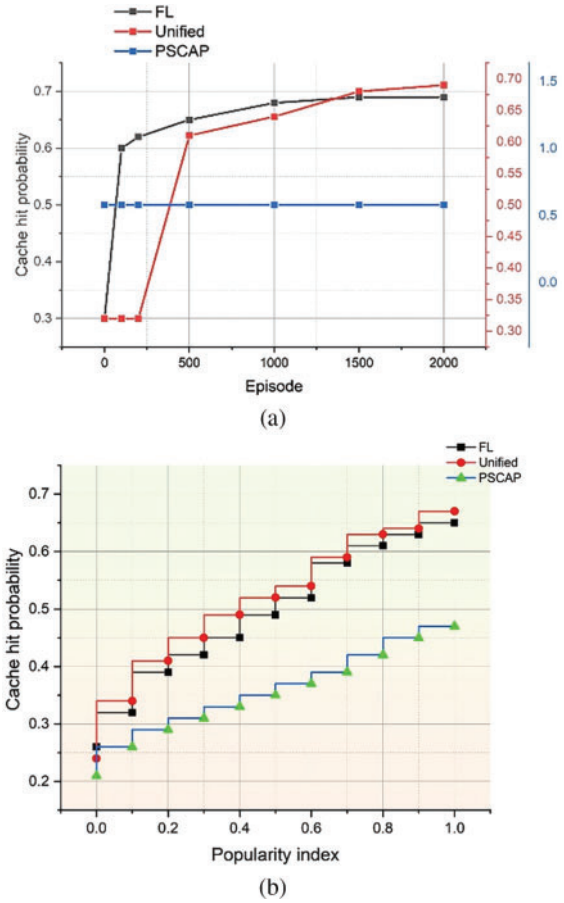

Figure 8: Cache hit probability for various cache deployment schemes using the proposed IMRSEC framework. (**a**) Likelihood of Cache hit vs. episodes; (**b**) Likelihood of Cache hit vs. popularity level (Δ)

Under different popularity indices, Fig. 8b presents the cache hit probability of various service cache deployment policies. If the popularity level = 0, all policies have an identical cache hit likelihood. DRL-based training methods (unified and decentralized) consistently beat PSCAP in terms of the cache hit probability.

5 Conclusion and Key Findings

IMRSEC is a platform that combines blockchain, edge computing, and artificial intelligence (AI) with 5G HNs. In addition, a new DS-DRL method has been created, which consists of a rapid schedule learning process and a slow schedule learning procedure. The main aim is to reduce overall unburdening latency by improving computational unloading, reserve distribution, and application cache deployment. The proposed DS-DRL has been compared against four other task execution methods, one resource allocation approach and one service caching methodology. Energy savings achieved are 26%, 18%, and 11% higher with the DS-DRL than with the EPA, IRPA, and PSCAP approaches. The cache-hit effectiveness of the Unified system is equivalent to that of the proposed framework, which is a favorable conclusion for both. Using the recommended united knowledge-based model training approach, it is feasible to maintain the confidentiality of ENs.

Acknowledgement: The authors are thankful to the Deanship of Graduate Studies and Scientific Research at University of Bisha for supporting this work through the Fast-Track Research Support Program.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Mohammad Tabrez Quasim, Khair Ul Nisa, and Mohammad Shahid Husain contributed to the conceptualization, design of the model, and data analysis. Abakar Ibraheem Abdalla Aadam, Mohammed Waseequ Sheraz, and Mohammad Zunnun Khan handled implementation, simulations, literature review, and manuscript preparation. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Yang S, Yin D, Song X, Dong X, Manogaran G, Mastorakis G, et al. Security situation assessment for massive MIMO systems for 5G communications. Future Gener Comput Syst. 2019;98:25–34. doi:10.1016/j.future.2019.03.036.
2. Garg S, Aujla GS, Erbad A, Rodrigues JJPC, Chen M, Wang X. Guest editorial: blockchain envisioned drones: realizing 5G-enabled flying automation. IEEE Netw. 2021;35(1):16–9. doi:10.1109/MNET.2021.9355047.
3. Sedik A, Tawalbeh L, Hammad M, El-Latif AAA, El-Banby GM, Khalaf AAM, et al. Deep learning modalities for biometric alteration detection in 5G networks-based secure smart cities. IEEE Access. 2021;9:94780–8. doi:10.1109/ACCESS.2021.3088341.
4. Li J, Lei G, Manogaran G, Mastorakis G, Mavromoustakis CX. D2D communication mode selection and resource optimization algorithm with optimal throughput in 5G network. IEEE Access. 2019;7:25263–73. doi:10.1109/ACCESS.2019.2900422.
5. Tong Z, Liu B, Mei J, Wang J, Peng X, Li K. Data security aware and effective task offloading strategy in mobile edge computing. J Grid Comput. 2023;21(3):41. doi:10.1007/s10723-023-09673-y.
6. Haghighi MS, Ebrahimi M, Garg S, Jolfaei A. Intelligent trust-based public-key management for IoT by linking edge devices in a fog architecture. IEEE Internet Things J. 2021;8(16):12716–23. doi:10.1109/JIOT.2020.3027536.
7. Lu Y, Huang X, Zhang K, Maharjan S, Zhang Y. Blockchain empowered asynchronous federated learning for secure data sharing in Internet of vehicles. IEEE Trans Veh Technol. 2020;69(4):4298–311. doi:10.1109/TVT.2020.2973651.

8. Dai Y, Xu D, Maharjan S, Zhang Y. Joint computation offloading and user association in multi-task mobile edge computing. IEEE Trans Veh Technol. 2018;67(12):12313–25. doi:10.1109/TVT.2018.2876804.
9. Chen M, Hao Y. Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J Select Areas Commun. 2018;36(3):587–97. doi:10.1109/jsac.2018.2815360.
10. Zhang K, Zhu Y, Leng S, He Y, Maharjan S, Zhang Y. Deep learning empowered task offloading for mobile edge computing in urban informatics. IEEE Internet Things J. 2019;6(5):7635–47. doi:10.1109/JIOT.2019.2903191.
11. Du Z, Wu C, Yoshinaga T, Yau KA, Ji Y, Li J. Federated learning for vehicular Internet of Things: recent advances and open issues. IEEE Comput Graph Appl. 2020;1:45–61. doi:10.1109/OJCS.2020.2992630.
12. Chen M, Yang Z, Saad W, Yin C, Poor HV, Cui S. A joint learning and communications framework for federated learning over wireless networks. IEEE Trans Wirel Commun. 2021;20(1):269–83. doi:10.1109/TWC.2020.3024629.
13. Yao H, Mai T, Wang J, Ji Z, Jiang C, Qian Y. Resource trading in blockchain-based industrial Internet of Things. IEEE Trans Ind Inform. 2019;15(6):3602–9. doi:10.1109/TII.2019.2902563.
14. Weng J, Weng J, Zhang J, Li M, Zhang Y, Luo W. DeepChain: auditable and privacy-preserving deep learning with blockchain-based incentive. IEEE Trans Dependable Secur Comput. 2021;18(5):2438–55. doi:10.1109/TDSC.2019.2952332.
15. Firdaus M, Rhee KH. On blockchain-enhanced secure data storage and sharing in vehicular edge computing networks. Appl Sci. 2021;11(1):414. doi:10.3390/app11010414.
16. Wang J, Hu J, Min G, Zomaya AY, Georgalas N. Fast adaptive task offloading in edge computing based on meta reinforcement learning. IEEE Trans Parallel Distrib Syst. 2021;32(1):242–53. doi:10.1109/TPDS.2020.3014896.