Volume 11, Issue 7, July– 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Decentralized Task Offloading for Multi-User Mobile Edge Computing with Machine Learning

Salman Alam[1]; Sami Ud Din[2]; Imtiaz Ali Shah[3]; Saeed Ur Rahman[4]; Aneesa Bibi[5]

[1]School of Electronic Engineering, Xidian University, Xi'an, Shaanxi, China
[2]Information and Communication Engineering, Xidian University, Xi'an, Shaanxi, China
[3]Department of Computer Science, COMSATS University of Technology, Abbottabad, Pakistan
[4]School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China
[5]University of Malakand, Chakdara, Lower Dir, Khyber Pakhtunkhwa, 18800, Pakistan

Publication Date: 2026/07/31

Abstract: The rapid growth of real-time and computation-intensive applications, such as face recognition, virtual reality, 3D gaming, augmented reality, and intelligent transportation systems, has significantly increased the demand for efficient data processing and low-latency services. However, mobile devices are constrained by limited computational capabilities and battery capacity, making them unsuitable for executing heavy workloads locally. Mobile Edge Computing (MEC) has emerged as a promising paradigm to offload computation tasks to nearby edge servers, thereby reducing latency and improving service quality. Despite its advantages, task offloading in MEC environments remains challenging due to the distributed nature of edge resources, energy constraints of end devices, and dynamic net- work conditions. Existing solutions based on heuristic methods, genetic algorithms, NOMA-based techniques, and mobility-aware services often suffer from high latency, excessive energy consumption, and task migration overhead. To address these limitations, this paper investigates a reinforcement learning-based computation offloading strategy using an improved Deep Deterministic Policy Gradient (IDDPG) algorithm. The proposed IDDPG approach enables decentralized decision-making by learning optimal offloading policies from local observations, effectively balancing local execution and task offloading. By minimizing computation costs, power consumption, and latency, the proposed method outperforms greedy offloading strategies and demonstrates improved efficiency in dynamic MEC environments.

Keywords: *Mobile Edge Computing; Deep Reinforcement Learning; Task Offloading; IDDPG; Energy Efficiency; Latency Minimization.*

How to Cite: Salman Alam; Sami Ud Din; Imtiaz Ali Shah; Saeed Ur Rahman; Aneesa Bibi (2026) Decentralized Task Offloading for Multi-User Mobile Edge Computing with Machine Learning. *International Journal of Innovative Science and Research Technology*, 11(7), 2139-2151. https://doi.org/10.38124/ijisrt/26jul612

I. INTRODUCTION

The rapid growth of computation-intensive and latency-sensitive mobile applications such as virtual reality, augmented reality, 3D gaming, face recognition, and intelligent transportation systems has sig- nificantly increased the demand for real-time data processing [1]. However, mobile devices are constrained by limited computational power, storage capacity, and battery life, making local execution of such tasks impractical [2]. The resulting surge in mobile data traffic places heavy pressure on wireless networks, increasing latency and backhaul congestion and degrading Quality of Service (QoS) and Quality of Experience (QoE) [3].

To overcome these limitations, computation offloading has emerged as an effective solution, enabling mobile devices to transfer intensive tasks to external servers for execution. Traditional cloud computing provides scalable computational resources, but its centralized nature and remote location introduce high latency, making it unsuitable for delay-sensitive applications such as autonomous driving, online gaming, and healthcare services [4-11]. Mobile Edge Computing (MEC) addresses this challenge by bringing cloud-like computational and storage resources closer to end users at the network edge [12-19]. By processing tasks near mobile devices, MEC significantly reduces latency, energy consumption, and transmission costs while improving service reliability. Consequently, MEC has become a key enabling technology for next-generation mobile networks and real-time applications.

In MEC environments, task offloading allows mobile devices with limited resources to delegate computation-intensive tasks to nearby edge servers [20-32]. Despite its advantages, efficient task offload- ing remains challenging

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

due to dynamic network conditions, limited edge resources, energy constraints, and task scheduling complexity. Key challenges include task assignment, resource allocation, and exe- cution decision-making under time-varying conditions [33]. Conventional approaches such as heuristic methods, genetic algorithms, NOMA-based schemes, and mobility-aware techniques often suffer from high latency, excessive energy consumption, migration overhead, and suboptimal decision-making in dy- namic environments [34-36]. Machine learning (ML), particularly reinforcement learning (RL), has gained attention for addressing task offloading challenges in MEC systems [37-39]. Unlike supervised and un- supervised learning, reinforcement learning enables an agent to learn optimal decision policies through interaction with the environment, making it well-suited for real-time and dynamic control problems [40]. RL-based approaches can adaptively decide whether tasks should be executed locally or offloaded to edge servers while balancing latency, energy consumption, and computational cost. However, traditional RL methods often suffer from sparse rewards, slow convergence, and high computational complexity [41-43].

Although existing ML-based task offloading techniques focus on minimizing delay and energy con- sumption, many generate sparse rewards and incur excessive training complexity, limiting their practical deployment. Moreover, centralized decision-making strategies fail to scale efficiently in decentralized MEC environments [44]. To address these limitations, this work proposes a reinforcement learning-based

Improved Deep Deterministic Policy Gradient (IDDPG) approach for decentralized computation offload- ing in MEC systems [45]. The proposed method learns intelligent offloading strategies based on local observations, enabling each user to decide between local execution and edge offloading dynamically. By reducing sparse rewards, unnecessary computation, and buffering delays, the proposed approach aims to significantly improve latency, energy efficiency, and overall system performance compared to state-of-the-art methods [46].

II. RELATED WORK

The rapid growth of computation-intensive mobile applications has exposed the inherent limitations of user equipment, including restricted processing capability, limited battery capacity, and stringent latency constraints. Mobile Edge Computing (MEC) has emerged as a key enabling technology to address these challenges by relocating computation tasks from resource-constrained devices to geographically proximate edge servers [47]. By doing so, MEC significantly reduces execution delay, energy expenditure, and network congestion, thereby improving Quality of Service (QoS) [48-49].

Numerous task offloading strategies have been explored in the literature, ranging from classical opti- mization frameworks to advanced machine learning and deep reinforcement learning approaches. Early learning-based solutions include the Energy-Efficient Deep Learning Offloading Scheme (EEDOS) pro- posed in [50], which eliminates the need for complex mathematical optimization by employing a trained deep neural network (DNN) to infer offloading decisions in near real time. The framework achieves no- table reductions in energy usage, transmission latency, and overall system cost. Despite these advantages, EEDOS is limited in scope as it considers only single-user scenarios and assumes sequential execution of application components, restricting its applicability in realistic multi-user MEC environments.

To address real-time offloading in wireless systems, [51] introduced DRLOCO, a deep reinforcement learning-based computation offloading framework designed for multi-carrier non-orthogonal multiple ac- cess (NOMA)-enabled MEC architectures. The proposed solution jointly determines offloading strate- gies and subcarrier assignments without relying on labeled training data. Although the model enhances spectrum utilization and computational efficiency, its decision-making process incurs substantial compu- tational overhead, particularly as the number of available subchannels increases.

In [52], decentralized task offloading strategies were investigated using Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) algorithms. The objective was to minimize long-term system cost by adaptively balancing local execution and edge offloading based on user behavior. While the approach successfully mitigates suboptimal greedy decisions, scalability remains a concern due to increased training time when handling large datasets. The study in [53] explored DQN-based task migration by addressing one-way data offloading through efficient buffer management. However, as the dimensionality of the system state increases, the learning process becomes inefficient, resulting in slow convergence and severe performance degradation due to the curse of dimensionality.

An Energy-Aware Quality Management (EMM) framework was presented in [54] to reduce access delay caused by limited radio and computing resources. A genetic algorithm was integrated to optimize computation cost and improve system profitability. Nevertheless, the framework experiences high time complexity during task migration between adjacent base stations. Mobility-aware service placement was addressed in [55] through the Energy-Efficient Smart Allocator Algorithm (EESA), which enables devices to harvest energy from access points prior to offloading tasks. This mechanism reduces reliance on conventional battery power and improves communication efficiency. However, the inability to support task migration across virtual machines restricts system adaptability.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Table 1 Comparative Analysis of Existing Task Offloading Approaches in MEC

Ref.	Year	Approach	Optimization Strategy
[27]	2019	EEDOS	Deep Neural Network
[28]	2020	DRLOCO	Reinforcement Learning
[29]	2020	DDPG/DQN	Adam-based learning
[30]	2020	DQN	Q-learning
[31]	2020	GAAA	Joint optimization
[32]	2020	EESA	PowMigExpand
[33]	2019	MDRL	DBSCAN-based Clustering
[34]	2019	DDPG	State normalization
[35]	2020	DRL-E2D	Markov decision framework
[36]	2020	Heuristic	Joint resource optimization
[37]	2016	Mobility Recursion	Migration control
[38]	2019	AF	Power allocation
[39]	2021	Context-Aware	CAVTO
[40]	2020	Cloudlets	Cloudlet architecture
[41]	2021	MACS	Adaptive partitioning

The authors in [56] proposed a Model-Based Multi-Agent Deep Reinforcement Learning (MDRL) framework to enhance protocol efficiency and incentivize offloading to edge servers. Although the ap- proach improves task planning accuracy, it depends heavily on a well-defined value function and is in- capable of handling multiple concurrent tasks effectively. A reinforcement learning-based energy control scheme was introduced in [57], leveraging DDPG to manage continuous state spaces and employing state normalization techniques to stabilize training. Despite improved learning performance, the framework encounters difficulties in wireless energy transfer, which negatively impacts overall network throughput.

For deadline-sensitive applications, [58] proposed the DRL-E2D algorithm, which maximizes cumu- lative reward while enforcing strict task deadline constraints in multi-user MEC environments. Simula- tion results demonstrate consistent performance gains over benchmark solutions across latency, energy consumption, and task completion metrics. In [59], a heuristic-based resource allocation strategy was developed to minimize access point energy consumption while supplying power to end devices. Although operational costs are reduced, the priority-based scheduling mechanism leads to excessive waiting times for low-priority tasks.

Mobility recursion and dynamic task partitioning were investigated in [60], where computation is managed directly at the edge. While suitable for lightweight workloads, the approach struggles with large-scale or concurrent task execution due to limited edge resources. An amplify-and-forward-based communication framework was proposed in [61] to minimize transmission delay and control bit error rates. However, its effectiveness is limited to small-sized tasks and does not scale well for computation- intensive applications. Context-aware task execution was explored in [62], allowing computation requests to be processed locally to alleviate network congestion. Nonetheless, insufficient consideration of global resource utilization results in load imbalance across MEC servers.

A cloudlet-assisted collaborative offloading framework was presented in [63], achieving significant la- tency reduction compared to centralized cloud solutions. To extend battery lifetime and reduce process- ing overhead, Mobile Augmentation Cloud Services (MACS) were proposed in [64], utilizing adaptive application partitioning. A greedy task replacement and resource allocation algorithm based on dis-tributed bandit optimization was introduced in [65]. Although the approach reduces delay and energy consumption, it does not account for cooperation among neighboring base stations. In [66], dynamic task scheduling was formulated as a mixed-integer programming (MIP) problem to reduce execution delay. Finally, a collaborative offloading-enabled distributed scheduling framework was presented in [68], although centralized coordination increases the probability of failure as task volume grows.

III. SYSTEM MODEL

As shown in Fig. 1, we consider a multi-user multiple-input multiple-output (MIMO) mobile edge computing (MEC) system composed of a base station (BS) equipped with N antennas, a co-located MEC server, and a set of single-antenna mobile users indexed by $M = \{1, 2, \ldots, M\}$. Due to limited onboard computational resources, each user $m \in M$ generates computation-intensive tasks that cannot be efficiently executed locally. To alleviate this limitation, a MEC server is deployed in close proximity to the BS, enabling partial task offloading through wireless uplink transmission to improve energy efficiency and reduce execution latency. The system operates in discrete time, where the operation horizon is divided into equal-length slots of duration $\tau 0$, indexed by $t \in T = \{0, 1, 2, \ldots\}$. Wireless channel conditions and task arrivals vary across time slots. Consequently, each user dynamically balances local computation and task offloading to achieve a trade-off between long-term energy consumption and task delay.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

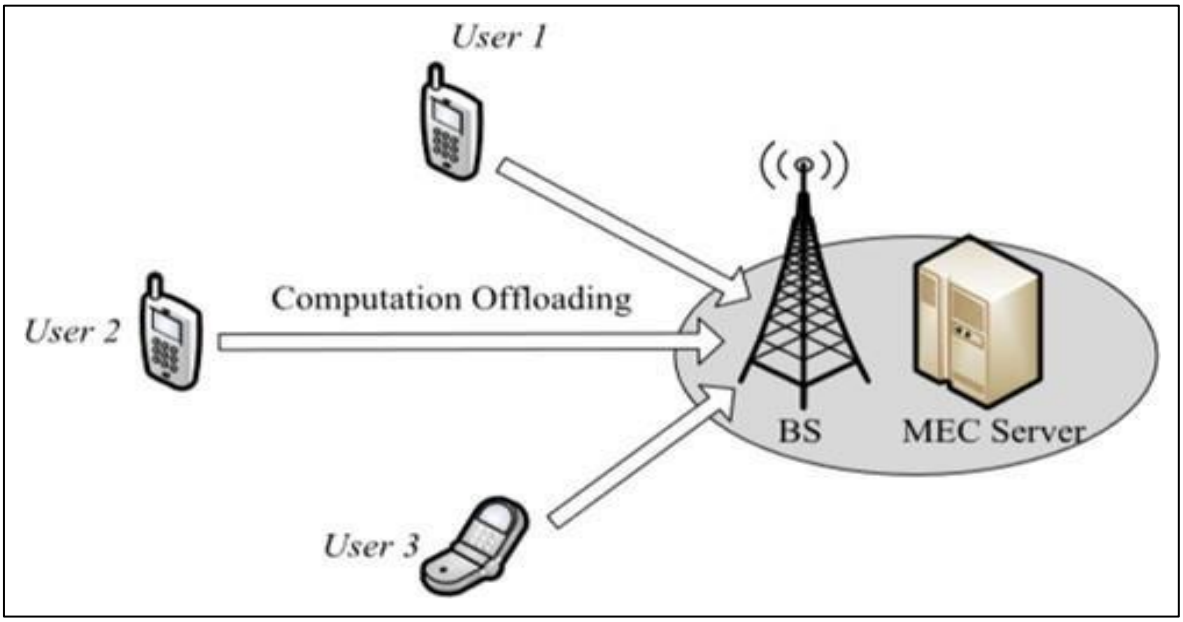

Fig 1 Multi-User MIMO-Enabled MEC System for Computation Offloading.

➢ *Network Model*

Let $h_m(t) \in C^{N\times 1}$ denote the uplink channel vector between user m and the BS during slot t. The received signal at the BS is expressed as:

$$\mathbf{y}(t) = \sum_m \mathbf{h}_m(t) \sqrt{p_{o,m}(t)} s_m(t) + \mathbf{n}(t) \quad (1)$$

Where $po,m(t) \in [0, Po,m]$ denotes the offloading transmit power of user m, $sm(t)$ is the normalized task data symbol with unit variance, and $n(t) \sim CN(0, \sigma^2 I_N)$ represents additive white Gaussian noise.

➢ *Computation Model*

Let $am(t)$ denote the number of task bits generated by user m during slot t. Task arrivals are assumed to be independent and identically distributed over time. Tasks are stored in a local buffer and can be processed starting from slot $t + 1$. During slot t, user m processes $dl,m(t)$ bits locally and offloads $do,m(t)$ bits to the MEC server, subject to buffer availability. Let $Bm(t)$ denote the buffer length at the beginning of slot t. The buffer dynamics are given by:

$$B_m(t+1) = [B_m(t) - d_{l,m}(t) - d_{o,m}(t)]^+ + a_m(t), \quad \forall t \in T \quad (2)$$

Where $Bm(0) = 0$ and $[x]^+ = \max\{x, 0\}$.

➢ *Local Computation Model*

Let $pl,m(t) \in [0, Pl,m]$ denote the local computation power of user m during slot t. The number of locally processed bits is given by:

$$d_{l,m}(t) = \tau_0 f_m(t) L_m^{-1} \quad (3)$$

Where $fm(t)$ is the CPU frequency and Lm is the number of CPU cycles required per task bit. Using dynamic voltage and frequency scaling (DVFS), the CPU frequency is modeled as:

$$f_m(t) = \sqrt[3]{p_{l,m}(t)/\kappa_m} \quad (4)$$

Where κ is the effective switching capacitance. The frequency is constrained by $fm(t) \in [0, Fm]$, where Fm is the maximum allowable CPU frequency.

➢ *Edge Computation Model*

The MEC server is assumed to have sufficient computational capability to process offloaded tasks in parallel with negligible execution delay. Thus, the number of bits offloaded by user m during slot t is determined by the uplink transmission rate as:

$$d_{o,m}(t) = \tau_0 W \log_2(1 + \gamma_m(t)) \quad (5)$$

Where W denotes the system bandwidth and $\gamma m(t)$ is the received signal-to-interference-plus-noise ratio (SINR).

➢ *Computation Cost Model*

The long-term computation cost of user m is defined as a weighted sum of energy consumption and queueing delay. According to Little's Law, the average delay is proportional to the average queue length. The cost function is given by:

$$C_m = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0} [w_{m,1}(p_{l,m}(t) + p_{o,m}(t)) + w_{m,2} B_m(t)] \quad (6)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Where $wm,_1 \geq 0$ and $wm,_2 \geq 0$ are weighting coefficients. Each user independently determines its local computation and offloading power by solving:

$$\min C_m \tag{7}$$

Subject to:

$$p_{l,m}(t) \in [0, P_{l,m}], \quad \forall t \in T \tag{8}$$

$$p_{o,m}(t) \in [0, P_{o,m}], \quad \forall t \in T \tag{9}$$

Decisions are made based solely on local observations, including task arrivals $am(t)$, estimated channel state $h_m(t)$ obtained via channel reciprocity, and previously observed SINR values. This decentralized framework significantly reduces signaling overhead and enhances scalability.

IV. PROPOSED ARCHITECTURE AND PROBLEM FORMULATION

We consider a multi-user MIMO MEC system consisting of a base station (BS) equipped with N antennas, a co-located MEC server, and a set of single-antenna mobile users denoted by $M = \{1, 2, \ldots, M\}$. Owing to the limited computational capability of mobile devices, each user is required to process computation- intensive tasks. To alleviate local processing constraints and enhance quality of service, users are allowed to offload a portion of their computation tasks to the MEC server via wireless uplink transmission [28]. The system operates in a discrete-time manner, where time is divided into slots of equal duration $\tau 0$. In each time slot $t \in T$, both wireless channel conditions and task arrivals are time-varying, which necessi- tates adaptive decisions on local execution and task offloading to balance long-term energy consumption and computation delay. As the number of users increases, decentralized task scheduling becomes in-creasingly attractive, as it mitigates signaling overhead and latency between users and the MEC server, thereby improving scalability and system sustainability.

Table 2 Notation Summary

Notation	Description
M	Set of mobile users
T	Index set of time slots
$\mathbf{h}_m(t)$	Channel vector between user m and BS at slot t
$\mathbf{y}(t)$	Received signal at the BS
ρm	Temporal channel correlation coefficient of user m
$\mathbf{H}(t)$	Channel matrix from all users to the BS
$\mathbf{g}_m(t)$	Zero-forcing (ZF) detection vector of user m
$\gamma_m(t)$	SINR of user m at BS during slot t

➢ *Deep Reinforcement Learning (DRL)*

The proposed IDDPG approach is built upon the DRL framework, which follows the standard RL structure consisting of an agent, an environment E, a set of states S, a set of actions A, and a reward function $r : S \times A \rightarrow R$. At each discrete time step t, the agent observes the current state $st \in S$, then selects and performs an action $at \in A$ based on its policy $\pi : S \rightarrow P(A)$. The agent then receives a scalar reward $rt = r(st, at)$ and transitions to the next state $st+1$ according to the conditional probability distribution $p(st+1 \mid st, at)$. The cumulative discounted return is defined as:

$$R_t = \sum_{i=t} \gamma^{i-t} r(s_i, a_i) \tag{10}$$

Where $\gamma \in [0, 1]$ is the discount factor and T is the total number of iterations. The RL objective is to find the policy π that maximizes the expected discounted reward:

$$J = E_{s\sim E, a\sim \pi}[R_1] \tag{11}$$

Within the policy domain, the action-value function (critic function) is defined as the expected cumulative return starting from state st and taking action at:

$$Q^{\pi}(s_t, a_t) : S \times A \rightarrow R \tag{12}$$

➢ *IDDPG-Based Decentralized Dynamic Control*

To minimize long-term computation cost, an Independent Deep Deterministic Policy Gradient (IDDPG) approach is adopted. Each user independently learns a decentralized control policy to allocate energy for local execution and task offloading using only locally observed information.

- *State Space*

As illustrated in Figure 2, the state observed by user m at the beginning of slot t consists of: (i) the current task buffer length $Bm(t)$, (ii) the normalized SINR feedback from the previous slot $\phi m(t-1)$, and (iii) the estimated uplink channel vector $h_m(t)$, obtained via channel reciprocity. The state vector is defined as:

$$\mathbf{s}_{m,t} = [B_m(t), \phi_m(t-1), \mathbf{h}_m(t)] \tag{13}$$

This formulation captures queue dynamics, interference effects from other users, and channel quality, while remaining fully decentralized.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

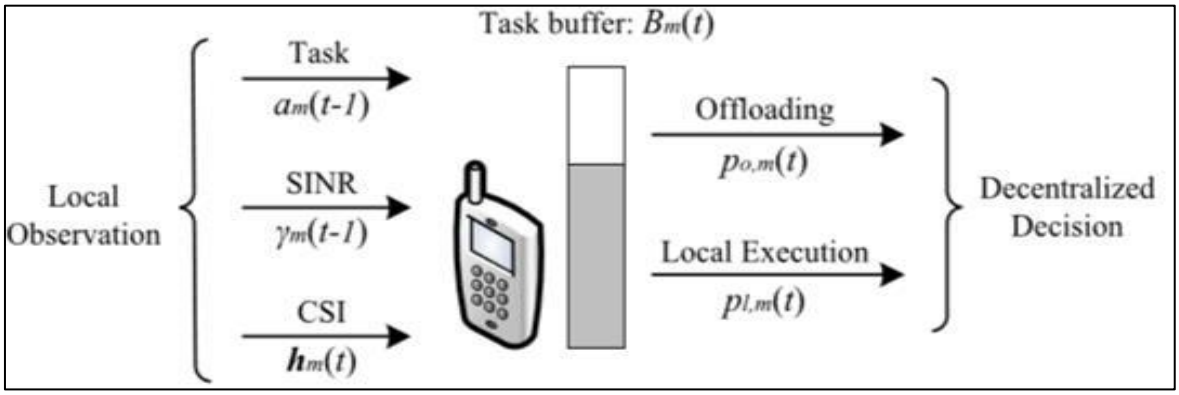

Fig 2 Decentralized Decision with Local Observations Showing Task Buffer $Bm(t)$, Previous Task $am(t-1)$, SINR $\gamma m(t-1)$, and CSI $h_m(t)$ Leading to Offloading $po,m(t)$ and Local Execution $pl,m(t)$ Decisions.

- *Action Space*

Given the observed state $\mathbf{s}_{m,t}$, each user selects a continuous-valued action:

$$\mathbf{a}_{m,t} = [p_{l,m}(t), p_{o,m}(t)] \quad (14)$$

Where $pl,m(t) \in [0, Pl,m]$ and $po,m(t) \in [0, Po,m]$. Unlike discrete power allocation methods, IDDPG directly operates in continuous action spaces, significantly reducing action-space dimensionality and computational burden.

- *Reward Function*

The instantaneous reward for user m at time slot t is defined as:

$$r_{m,t} = -w_{m,1} \cdot (p_{l,m}(t) + p_{o,m}(t)) - w_{m,2} \cdot B_m(t) \quad (15)$$

This reward function penalizes both energy consumption and task queue backlog, thereby encouraging energy-efficient and delay-aware computation offloading decisions.

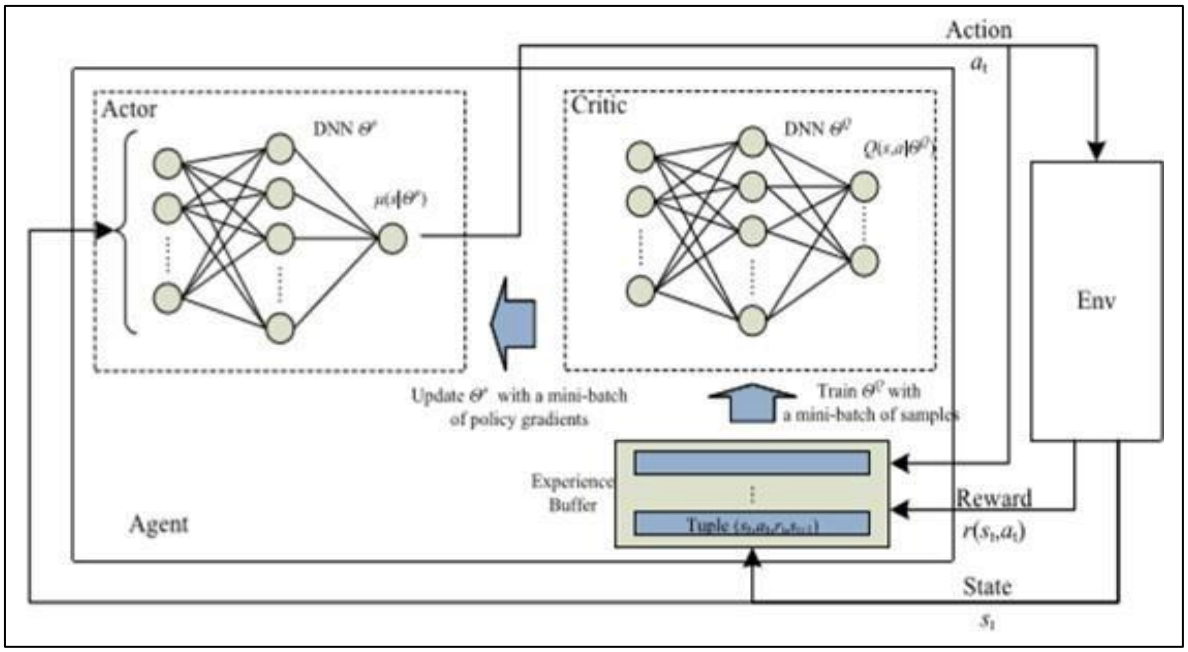

Fig 3 IDDPG Algorithm Architecture Showing Actor Network $\mu(\theta^a)$, Critic Network $Q(s, a; \theta^c)$, Experience Replay Buffer, and the Learning Loop with State st, Action at, and Reward $r(st, at)$.

V. IDDPG-BASED LEARNING ALGORITHM

➢ *Algorithm 1: IDDPG-Based Decentralized Learning Procedure*

for every agent m in M do
Randomly initialize actor network mu(s|theta^u_m) and critic network Q(s,a|theta^Q_m)
Initialize target networks: theta^u'_m <- theta^u_m and theta^Q'_m <- theta^Q_m 4: Initialize empty experience replay buffer B_m
end for

for every episode k = 1 to K_max do
Reset simulation parameters for the multi-user MEC environment
Randomly initialize state s_{m,1} for each agent m in M

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

for each time slot t = 1 to T_max do
for every agent m in M do
Select action a_{m,t} = mu(s_{m,t}|theta^u_m) + Delta_mu
Execute action a_{m,t}; obtain reward r_{m,t} and next state s_{m,t+1}
Store transition (s_{m,t}, a_{m,t}, r_{m,t}, s_{m,t+1}) into buffer B_m
Sample a mini-batch of I tuples from B_m
Update critic network by minimizing loss L
Update actor network using sampled policy gradient
Perform soft target network updates
end for
end for
end for

Algorithm 1 describes the complete IDDPG-based learning procedure, where each user initializes an actor network $\mu(s|\theta^{\mu})$ and a critic network $Q(s, a|\theta^{Q})$ along with their target networks, and maintains an independent replay buffer Bm. At each time slot, actions are generated using the current actor policy with added exploration noise. Transitions are stored in the replay buffer, and mini-batches are sampled to update the critic by minimizing the mean-squared Bellman error and to update the actor via policy gradient descent, while target networks are softly updated using stochastic gradient descent.

➢ *Training and Testing Procedure*

A simulation environment is developed to support decentralized training and evaluation. During training, agents interact with the environment for *Kmax* episodes, each lasting *Tmax* time steps. Experience tuples ($\mathbf{s}_{m,t}$, $\mathbf{a}_{m,t}$, rm,t, $\mathbf{s}_{m,t+1}$) are stored in replay buffers and used for network updates. A key challenge lies in balancing exploration and exploitation in continuous action spaces. Exploration is implemented by injecting carefully designed random noise into the actor outputs, while learning remains deterministic.

During testing, each agent loads the trained actor network and interacts with a randomly initialized environment without exploration noise, making decisions solely based on local observations.

VI. RESULTS AND DISCUSSION

This section presents numerical simulations to validate the effectiveness of the proposed decentralized dynamic computation offloading framework based on the Improved Deep Deterministic Policy Gradient (IDDPG) algorithm. The performance of the IDDPG scheme is evaluated and compared with benchmark approaches under both single-user and multi-user scenarios.

➢ *Simulation Setup*

The system operates in discrete time slots with duration $T0$ = 1 ms. Task arrivals at each mobile user m follow a Poisson distribution, denoted by $am(t) \sim Poisson(\lambda m)$, where $\lambda m = E[am(t)]$ represents the average task arrival rate of user m. At the beginning of each episode, the channel vector for user m is initialized as:

$$\mathbf{h}_m(0) \sim \mathrm{CN}\left(0, h_0(d_0/d_m)^{a}\mathbf{I}_N\right) \quad (16)$$

Where $h0 = -30$ dB is the path-loss constant, $d0$ = 1 m is the reference distance, $\alpha = 3$ is the path-loss exponent, and dm denotes the distance between user m and the base station (BS). The channel evolves over time with a correlation coefficient $\rho m = 0.95$ and error vector:

$$\mathbf{e}(t) \sim \mathrm{CN}\left(0, h_0(d_0/d_m)^{a}\mathbf{I}_N\right) \quad (17)$$

Where the Doppler frequency is set to fd,m = 70 Hz. The system bandwidth is fixed at 1 MHz. The maximum transmission power is Po,m = 2 W, and the interference-plus-noise power is $\sigma^2 = 10^{-9}$ W. For local execution, the required CPU cycles per bit are Lm = 500 cycles/bit, and the maximum CPU frequency is Fm = 1.26 GHz, resulting in a maximum local computation power consumption of Pl,m = 2 W.

The actor and critic networks for each user agent are implemented as fully connected neural networks with four layers, including two hidden layers containing 400 and 300 neurons, respectively. All hidden layers employ the ReLU activation function $f(x) = \max(0, x)$. Network parameters are optimized using the Adam optimizer with learning rates of 0.0001 and 0.001 for the actor and critic, respectively. The target networks are updated using a soft update factor τ = 0.001. To encourage exploration, temporally correlated noise is generated using the Ornstein–Uhlenbeck process with parameters $\theta = 0.15$ and $\sigma = 0.12$. The replay buffer size is $|Bm| = 2.5 \times 10^5$, and the mini-batch size is $M = 16$. The maximum number of training episodes is $Kmax$ = 2000, and each episode consists of at most $Tmax$ = 200 time steps.

Table 3 Simulation Parameters

Parameter	Value
Time slot duration T_0	1 ms
Task arrival model	Poisson(λ_m)
Path-loss constant h_0	-30 dB
Reference distance d_0	1 m
Path-loss exponent α	3
Channel correlation coefficient ρ_m	0.95
System bandwidth W	1 MHz
Max transmission power $P_{o,m}$	2 W
Noise power σ^2	10^{-9} W

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

CPU cycles per bit L_m	500 cycles/bit
Max CPU frequency F_m	1.26 GHz
Max local computation power $P_{l,m}$	2 W
Actor learning rate	0.0001
Critic learning rate	0.001
Soft update factor τ	0.001
OU noise parameters (θ, σ)	0.15, 0.12
Replay buffer size $\|B_m\|$	2.5×10^5
Mini-batch size M	16
Training episodes K_{max}	2000
Time steps per episode T_{max}	200

➢ *Single-User Scenario*

In the single-user (SU) case, the mobile user is randomly located at a distance of $d1 = 100$ m from the BS. The dynamic computation offloading objective is controlled by the weighting factor $w1 \in \{0.5, 0.8\}$. Each result is averaged over 10 independent simulation runs. Two task arrival rates, $\lambda 1 =$ 2.0 Mbps and 3.0 Mbps, are considered.

- *Training Phase*

Figures 4 and 5 illustrate the training performance of IDDPG, DDPG, and DQN for $w1 = 0.5$ and 0.8, respectively. The average reward per episode increases steadily as the agents learn improved of- floading policies through continuous interaction with the environment. Convergence is observed after approximately 1500 episodes.

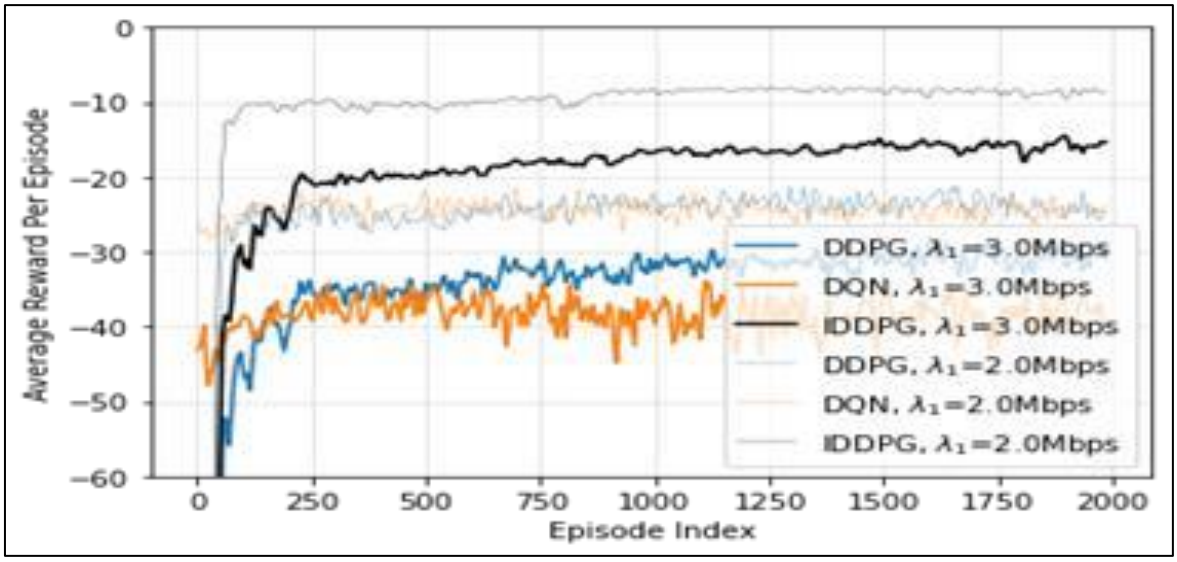

Fig 4 Training Phase, Average Reward Per Episode for Single-User Agent ($w1 = 0.5$).

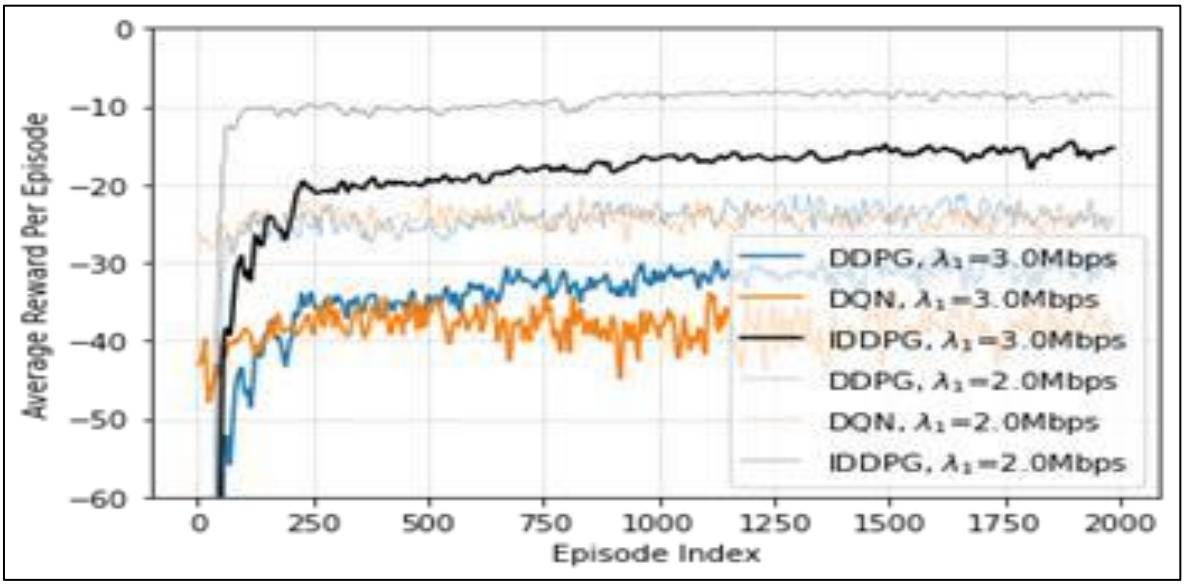

Fig 5 Training Phase, Average Reward Per Episode for Single-User Agent ($w1 = 0.8$).

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Across all configurations, IDDPG consistently achieves a higher steady-state reward than DDPG and DQN, demonstrating superior exploration efficiency in continuous action spaces. This performance gain is attributed to the stochastic gradient-based actor update mechanism, which mitigates reward divergence and stabilizes learning.

- *Testing Phase*

After completing *Kmax* = 2000 training episodes, the learned policies are evaluated over 100 indepen- dent test runs, each consisting of 10,000 time steps. Figures 6 and 7 present the average reward, energy consumption, and buffering delay for $w1 = 0.5$ and 0.8. As the task arrival rate increases, the average reward decreases, reflecting the higher computation cost associated with increased workload. While ID- DPG outperforms greedy strategies (GD-Local and GD-Offload) in terms of reward and energy efficiency, it incurs a marginally higher buffering delay to achieve lower power consumption. For task arrival rates exceeding 3.5 Mbps, the DQN-based method performs worse than greedy approaches due to its limited discrete power levels.

- *Performance Comparison for Different w1 Values*

✓ Average Reward: Figure 6 shows that the average reward decreases with increasing task arrival rate for both $w1 = 0.5$ and 0.8. The IDDPG-based approach consistently outperforms competing methods. A higher weight ($w1 = 0.8$) emphasizes power efficiency, leading to improved reward performance compared to $w1 = 0.5$.

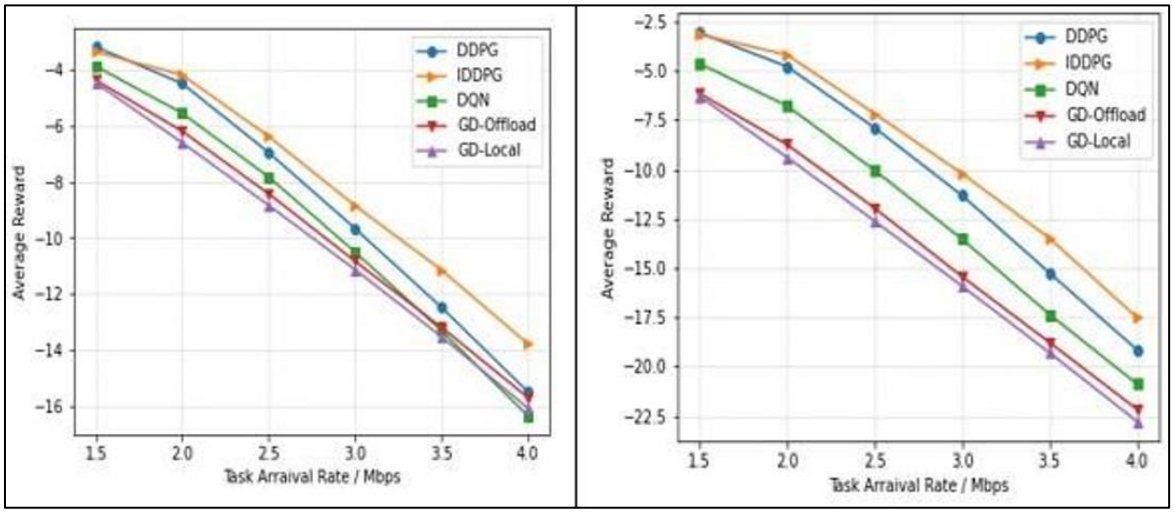

Fig 6 Average Reward Results for w = 0.5 and 0.8.

➢ *Average Power Consumption Results*

Figure 7 illustrates the average power consumption comparison among all evaluated methods across different task arrival rates. The results demonstrate that IDDPG consistently achieves the lowest power consumption compared to DDPG, DQN, GD-Offload, and GD-Local.

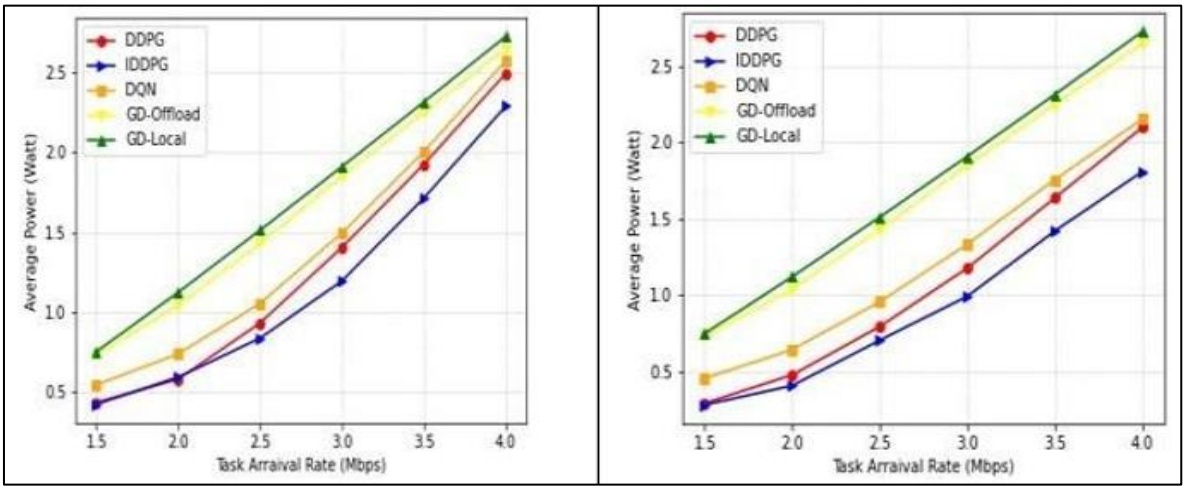

Fig 7 Average Power Consumption (W) Results for Different Task Arrival Rates.

As observed in Figure 7, power consumption increases with higher task arrival rates due to increased computational demand. IDDPG adaptively balances local computation and offloading decisions, result- ing in superior energy efficiency. In contrast, greedy approaches (GD-Local and GD-Offload) consume significantly more power, especially at higher task arrival rates, as they lack adaptive learning capabilities.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

✓ Average Buffering Delay: Figure 8 compares buffering delays across different schemes. For $w1$ = 0.8, IDDPG significantly reduces buffering delay by executing more tasks within each time slot. In contrast, $w1$ = 0.5 results in longer delays due to conservative power usage. The greedy methods (GD- Local and GD-Offload) exhibit similar buffering delays, as their behavior is independent of $w1$.

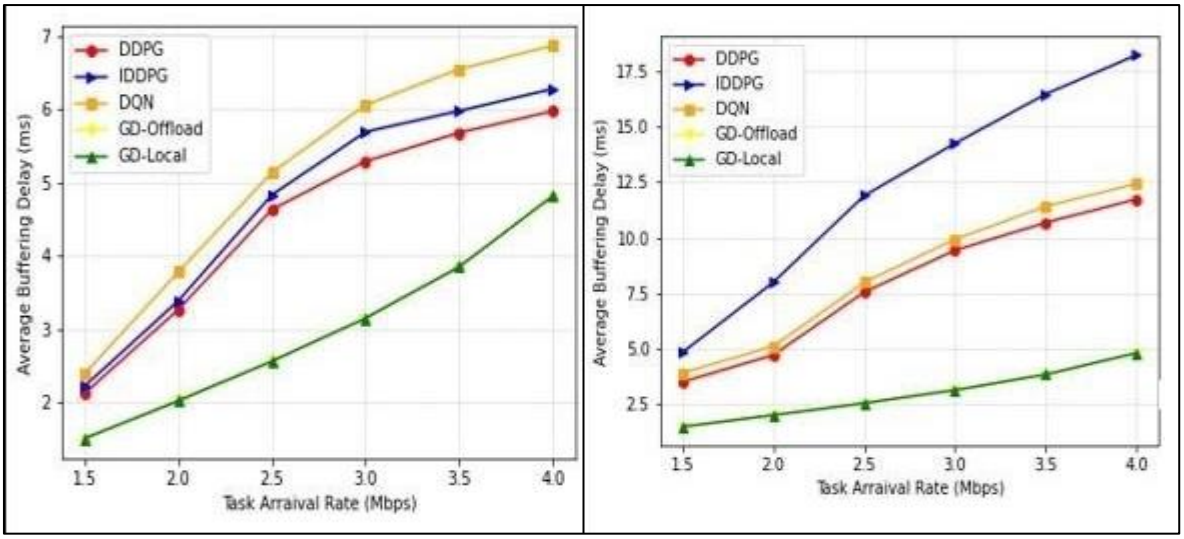

Fig 8 Average Buffering Delay (ms) Results for w = 0.5 and 0.8.

➢ *Multi-User Scenario*

The multi-user system consists of M = 3 mobile users, each located at dm = 100 m from the BS, with identical task arrival rates λm = 1.0 Mbps.

- *Training*

Figure 9 presents the training performance for all users with wm = 0.5. The average reward improves gradually for each user, indicating successful learning of decentralized offloading policies. Users with higher computation demands incur higher computation costs. Compared to the single-user case, the average reward is lower due to reduced transmission efficiency caused by increased user competition.

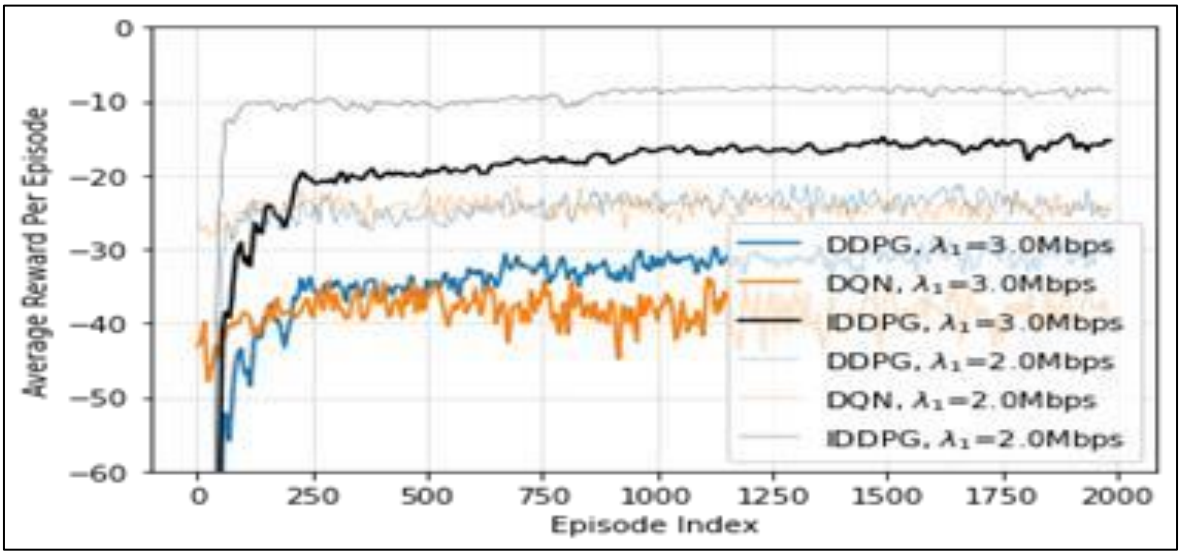

Fig 9 Training Phase, Average Reward Per Episode for Single-User Agent Comparing IDDPG, DDPG, and DQN for $\lambda 1$ = 2.0 Mbps and 3.0 Mbps.

- *Testing*

Testing results after $Kmax$ = 2000 episodes are summarized in Tables 4 and 5. For wm = 0.5, IDDPG achieves superior average reward for users 2 and 3, while slightly underperforming GD-Local for user 1, suggesting room for improved exploration under low power constraints. When wm = 0.8, IDDPG achieves the best average reward for all users, with significantly reduced power consumption and a moderate increase in buffering delay.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Table 4 Testing Results for All Mobile Users (*wm* = 0.5)

Method	Avg Reward			Avg Power (W)			Avg Delay (ms)		
	User1	User2	User3	User1	User2	User3	User1	User2	User3
DDPG	-1.770	-5.670	-12.782	0.205	0.774	1.939	1.489	3.600	6.174
IDDPG	**5.693**	**9.500**	**23.562**	**0.135**	**0.355**	**0.743**	1.374	3.310	5.637
DQN	-2.174	-7.657	-14.688	0.292	1.156	2.320	1.428	3.753	6.176
GD-Offload	-2.514	-7.597	-18.690	0.402	1.309	3.143	1.007	1.103	5.951
GD-Local	-1.633	-9.504	-20.071	0.216	1.678	3.407	1.106	2.228	6.072

Table 5 Testing Results for All Mobile Users (*wm* = 0.8)

Method	Avg Reward			Avg Power (W)			Avg Delay (ms)		
	User1	User2	User3	User1	User2	User3	User1	User2	User3
DDPG	-1.919	-6.366	-16.164	0.162	0.602	1.674	3.114	7.752	13.861
IDDPG	**5.765**	**11.471**	**23.890**	**0.093**	**0.272**	**0.411**	1.890	6.910	11.281
DQN	-2.780	-8.915	-18.675	0.284	0.915	1.954	2.539	7.973	15.216
GD-Offload	-3.417	-10.893	-26.334	0.402	1.309	3.143	1.007	2.103	15.951
GD-Local	-1.949	-13.870	-28.470	0.216	1.678	3.407	1.106	2.228	16.072

Overall, the IDDPG-based approach consistently outperforms DDPG-, DQN-, and greedy-based methods in terms of average reward and energy efficiency. By appropriately selecting the tradeoff pa- rameter *wm*, the system can effectively balance power consumption and buffering delay in multi-user scenarios.

VII. CONCLUSIONS

The rapid expansion of computation-intensive and delay-sensitive mobile applications, including face recognition, virtual reality, three-dimensional gaming, augmented reality, and intelligent transportation, has significantly increased the demand for real-time data processing. However, the limited processing capability and energy availability of mobile devices hinder their ability to efficiently execute such work- loads. Mobile edge computing (MEC) addresses this limitation by enabling computation offloading to nearby edge servers, thereby reducing execution latency.

This paper adopts a decentralized dynamic computation offloading approach based on the independent deep deterministic policy gradient (IDDPG) algorithm with a continuous state space. Relying exclusively on locally observed information, each mobile user learns an effective computation strategy that jointly determines power allocation for local execution and task offloading, achieving efficient resource utilization in a decentralized manner. Simulation results validate that the proposed IDDPG approach consistently outperforms DDPG-, DQN-, and greedy-based benchmark methods in terms of average reward and energy efficiency across both single-user and multi-user scenarios.

FUTURE WORK

Future research will focus on decentralized binary computation offloading for indivisible tasks, where tasks must be processed either entirely locally or entirely at the base station. Addressing this scenario may require a hierarchical deep reinforcement learning framework that integrates discrete offloading decisions with continuous power control.

REFERENCES

[1]. L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520-5530, 2019.

[2]. S. Wang, T. Lei, L. Zhang, C.-H. Hsu, and F. Yang, "Offloading mobile data traffic for QoS-aware service provision in vehicular cyber-physical systems," *Future Generation Computer Systems*, vol. 61, pp. 118-127, 2016.

[3]. J. Dai, Z. Zhang, and D. Liu, "Proactive caching over cloud radio access network with user mobility and video segment popularity," *IEEE Access*, vol. 6, pp. 44396-44405, 2018.

[4]. N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, 2017.

[5]. D. Liu and C. Yang, "A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users," *IEEE Access*, vol. 7, pp. 83120-83136, 2019.

[6]. A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *Journal of Systems Architecture*, p. 102225, 2021.

[7]. G. Lu and W. H. Zeng, "Cloud computing survey," in *Proc. Trans Tech Publications*, 2014, pp. 650-661.

[8]. H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang, and S. Wu, "Cloud types and services," in *Handbook of Cloud Computing*, pp. 335-355, Springer, Boston, MA, 2010.

[9]. A. Poniszewska-Maranda, R. Matusiak, N. Kryvinska, and A. U. H. Yasar, "A real-time service system in the cloud," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 3, pp. 961-977, 2020.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

[10]. A. Asaduzzaman, A. R. Joseph, F. N. Sibai, and N. Mohamed, "Cloud computing: A cloudy future?" in *Proc. IEEE IIT*, 2012, pp. 78-82.
[11]. S. K. Yoo and B. Y. Kim, "A decision-making model for adopting a cloud computing system," *Sustainability*, vol. 10, no. 8, p. 2952, 2018.
[12]. X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 178-183, 2019.
[13]. A. Sunyaev, "Cloud computing," in *Internet Computing*, Springer, 2020, pp. 195-236.
[14]. M. U. Bokhari, Q. Makki, and Y. K. Tamandani, "A survey on cloud computing," in *Big Data Analytics*, pp. 149-164, Springer, Singapore, 2018.
[15]. M. F. Mushtaq, U. Akram, I. Khan, S. N. Khan, A. Shahzad, and A. Ullah, "Cloud computing environment and security challenges: A review," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, pp. 183-195, 2017.
[16]. S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012-1023, 2013.
[17]. C. N. Hoefer and G. Karagiannis, "Taxonomy of cloud computing services," in *Proc. IEEE Globecom Workshops*, 2010, pp. 1345-1350.
[18]. K. Gai and A. Steenkamp, "Feasibility of a Platform-as-a-Service implementation using cloud computing," in *Proc. CONISAR*, vol. 2167, 2013.
[19]. K. K. M. Kumar, "Software as a service for efficient cloud computing," 2014.
[20]. C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge com- puting: A survey," *IEEE Access*, vol. 7, pp. 131543-131558, 2019.
[21]. Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1-16, 2015.
[22]. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv:1701.01090*, 2017.
[23]. J. Lee, J.-W. Kim, and J. Lee, "Mobile personal multi-access edge computing architecture composed of individual user devices," *Applied Sciences*, vol. 10, no. 13, p. 4643, 2020.
[24]. M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473-1499, 2015.
[25]. T. Hou et al., "Proactive content caching by exploiting transfer learning for mobile edge computing," *International Journal of Communication Systems*, vol. 31, no. 11, p. e3706, 2018.
[26]. P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A survey of hierarchical energy optimization for mobile edge computing," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1-44, 2020.
[27]. H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514-4526, 2018.
[28]. H. Yu, B. Ng, and W. Seah, "Multi-gateway polling for nanonetworks under dynamic IoT backhaul bandwidth," in *Proc. IEEE*, 2018, pp. 1-4.
[29]. A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210-229, 1959.
[30]. M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, MIT Press, 2018.
[31]. B. Adam, I. F. C. Smith, and F. Asce, "Reinforcement learning for structural control," *Journal of Computing in Civil Engineering*, vol. 22, no. 2, pp. 133-139, 2008.
[32]. P. C. Sen, M. Hajra, and M. Ghosh, "Supervised classification algorithms in machine learning: A survey and review," in *Emerging Technology in Modelling and Graphics*, Springer, Singapore, 2020, pp. 99-111.
[33]. G. K. Uyanik and N. Güler, "A study on multiple linear regression analysis," *Procedia—Social and Behavioral Sciences*, vol. 106, pp. 234-240, 2013.
[34]. T. O. Ayodele, "Types of machine learning algorithms," *New Advances in Machine Learning*, vol. 3, pp. 19-48, 2010.
[35]. A. I. Karoly, R. Fuller, and P. Galambos, "Unsupervised clustering for deep learning: A tutorial survey," *Acta Polytechnica Hungarica*, vol. 15, no. 8, pp. 29-53, 2018.
[36]. R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, 2005.
[37]. X. Wu et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, 2008.
[38]. M. Espadoto et al., "Toward a quantitative survey of dimension reduction techniques," *IEEE Trans- actions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2153-2173, 2019.
[39]. G. Chao, Y. Luo, and W. Ding, "Recent advances in supervised dimension reduction: A survey," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 341-358, 2019.
[40]. K. Arulkumaran et al., "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, 2017.
[41]. B. Kiumarsi et al., "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042-2062, 2017.
[42]. F. Li et al., "Multi-armed-bandit-based spectrum scheduling algorithms in wireless networks: A survey," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 24-30, 2020.
[43]. G. Miller, "The smartphone psychology manifesto," *Perspectives on Psychological Science*, vol. 7, no. 3, pp. 221-237, 2012.
[44]. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications*

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26jul612

Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, 2017.

[45]. M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587-597, 2018.

[46]. Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623- 149633, 2019.

[47]. H. Ke, H. Wang, W. Sun, and H. Sun, "Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks," *IEEE Transactions on Network and Service Management*, 2021.

[48]. Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1-21, 2020.

[49]. M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, 2020.

[50]. Z. Ali, S. Khaf, Z. H. Abbas, G. Abbas, F. Muhammad, and S. Kim, "A deep learning approach for mobility-aware and energy-efficient resource allocation in MEC," *IEEE Access*, vol. 8, pp. 179530- 179546, 2020.

[51]. S. U. Malik, T. Kanwal, S. U. Khan, H. Malik, and H. Pervaiz, "A user-centric QoS-aware multi-path service provisioning in mobile edge computing," *IEEE Access*, vol. 9, pp. 56020-56030, 2021.

[52]. J. B. Hamrick et al., "On the role of planning in model-based deep reinforcement learning," 2020.

[53]. C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577-8588, 2019.

[54]. Z. Li et al., "Energy-aware task offloading with deadline constraint in mobile edge computing," 2021.

[55]. T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," in *Proc. IEEE WOCC*, 2018, pp. 1-4.

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

[57]. X. Zhang, J. Wu, W. Shi, Y. Wu, and Y. Miu, "Low-latency cooperative computation offloading for mobile edge computing," in *Proc. IEEE ISPA/BDCloud*, 2019, pp. 155-159.

[58]. Z. Jin, C. Zhang, G. Zhao, Y. Jin, and L. Zhang, "A context-aware task offloading scheme in collaborative vehicular edge computing systems," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 2, 2021.

[59]. N. Abdenacer, H. Wu, N. N. Abdelkader, S. Dhelim, and H. Ning, "A novel framework for mobile edge computing by optimizing task offloading," *IEEE Internet of Things Journal*, 2021.

[60]. X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet of Things Journal*, 2021.

[61]. Z. Sun and M. R. Nakhai, "An online learning algorithm for distributed task offloading in multi-access edge computing," *IEEE Transactions on Signal Processing*, vol. 68, pp. 3090-3102, 2020.

[62]. H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668-682, 2019.

[63]. T. Yang, R. Chai, and L. Zhang, "Latency optimization-based joint task offloading and scheduling for multi-user MEC system," in *Proc. IEEE WOCC*, 2020, pp. 1-6.

[64]. M. D. Hossain et al., "Collaborative task offloading for overloaded mobile edge computing in small- cell networks," in *Proc. IEEE ICOIN*, 2020, pp. 717-722.