Quantized Graph-Based Personalized DRL for Dependency-Aware Task Offloading in Heterogeneous Edge Networks

Manjinder Kaur, Hui Cheng, and Pandelis Kourtessis

***Abstract*—Task offloading in mobile edge computing (MEC) becomes particularly challenging when dealing with complex, interdependent subtasks, especially under dynamic network conditions and resource constraints. This challenge is central to enabling intelligent and adaptive behavior in holographic counterparts, real-time virtual replicas of physical consumer Internet of Things (IoT) devices such as smart wearables, home automation units, or augmented reality systems. In this paper, we propose HiDeR- GQ, a novel Hierarchical Dependency Reduction framework that integrates quantized graph-based deep reinforcement learning (DRL) which acts as the decision-making core of holographic counterparts deployed in heterogeneous edge networks. The core of HiDeR-GQ is a customized deep reinforcement learning agent that integrates a quantized Graph Attention Network (GATv2) for structure-aware state encoding of task dependency graphs, and a personalized federated Double Deep Q-Network (DDQN) to enable collaborative learning across distributed edge nodes while accounting for heterogeneity. To further optimize performance, we introduce a latency-aware dependency pruning mechanism that hierarchically simplifies task graphs by removing non-critical dependencies to reduce end-to-end latency without compromising task accuracy. We evaluate HiDeR-GQ in a simulated MEC environment involving multiple edge servers and users executing directed acyclic graph (DAG) structured application tasks. The results demonstrate that HiDeR-GQ achieves up to 25% lower average task completion delay, higher cumulative reward, and reduced communication cost compared to state-of-the-art offloading strategies. These benefits come with minimal trade-offs in decision optimality, highlighting HiDeR-GQ's efficiency and scalability for next-generation IoT and edge intelligence applications.**

***Index Terms*—Task Offloading, Mobile Edge Computing (MEC), Deep Reinforcement Learning (DRL), Graph Attention Networks (GAT), Quantization, Federated Learning.**

I. Introduction

THE rapid proliferation of Internet of Things (IoT) devices and the rise of latency-sensitive consumer electronics, such as wearables in augmented reality (AR), smart home appliances, and health monitors, has increased the need for real-time intelligence at the network edge [1]. A key enabler of this intelligence is the Holographic Counterpart (HC), a continuously synchronized virtual representation of a physical consumer device that empowers predictive control, adaptive offloading, and autonomous optimization in dynamic environments [2] [3]. These HCs must efficiently analyze task dependencies, make decisions under resource constraints, and adapt to device heterogeneity all in real time. Such as intelligent task offloading mechanisms are crucial to ensuring low latency, responsive execution in modern consumer IoT ecosystems [4] [5]. In this context, MEC plays a key role by decentralizing computational resources to the network edge to reduce latency and bandwidth congestion while enabling intelligent decision-making at the HC level [6]. For example, a smart wearable digital twin can assess its energy state, workload, and network conditions to decide whether to execute a task locally or offload it to a nearby edge server [7]. However, such twin-based systems often rely on complex task graphs with interdependencies that must be executed in a predefined order. Efficiently offloading such dependency-laden tasks in MEC environments is essential but challenging due to resource constraints and dynamic workload variations. Digital twins must continuously evaluate whether to process tasks locally or offload them, based on task priority, device state, and network dynamics [8], [9].

Many of these advanced applications require substantial computation, which strains resource-constrained mobile devices [10]–[12]. To mitigate this issue, a traditional technique involves transferring all data and computations from the mobile device to a cloud server for processing [13]. However, MEC represents a fundamental shift by bringing computation closer to the data source, enhancing efficiency and minimizing delay. Offloading computation-intensive tasks to edge servers reduces processing delay in the edge computing system [14]. These tasks typically follow task graphs with predefined dependencies (e.g., sensor data must be pre-processed before ML inference).

Task offloading in MEC can be categorized into fine-grained and coarse-grained methods. The fine-grained method divides the task into smaller interdependent subtasks, allowing concurrent processing on multiple edge servers and thereby enhancing resource utilization and minimizing latency in delay-sensitive applications [15]. Many practical applications involve tasks structured as Directed Acyclic Graphs (DAGs), which exhibit dependencies that require tasks to be executed sequentially (e.g., sensor data preprocessing preceding machine learning inference). Figure 1 illustrates a task decomposition process [16], [17]. However, the inherent complexity in these dependencies introduces computational challenges, which classify DAG- based offloading as an NP-hard problem [18]. Several critical challenges arise in dependency-aware task offloading:

- **Dynamic Task Dependencies:** Modern applications such as real-time video analytics pipelines involve DAGs of tasks with intricate dependency relationships [19]. Conventional offloading methods often ignore these relationships, which can lead to invalid schedules where predecessor tasks must be completed before successor

Manjinder Kaur, Hui Cheng, and Pandelis Kourtessis are with the School of Physics, Engineering & Computer Science, University of Hertfordshire, Hatfield, United Kingdom (e-mail: manjindermatharu48@gmail.com; h.cheng2@herts.ac.uk; p.kourtessis@herts.ac.uk).

tasks can begin execution, potentially causing system failures or redundant processing [20].

- **Resource-Precision Trade-offs:** Deploying high-precision models on edge devices is infeasible due to memory and latency bottlenecks [21]. Although uniform quantization reduces computational load, it can disrupt task dependencies, leading to suboptimal or unstable decisions.

Previous work addresses these issues in isolation. Graph attention networks (GATs) prune edges statically by overlooking evolving task priorities [22]. Post-training quantization applies uniform bit widths across layers, which ignores sensitivity variations [23]. Our framework bridges this gap through a hierarchical quantization strategy coupled with an adaptive graph attention mechanism to ensure accurate decision-making. We propose HiDeR-GQ, a novel framework that enables resource-constrained devices to make intelligent task-offloading decisions through adaptive sparse GATs and hierarchical DRL. Key aspects of our approach include:

- **Quantized GATv2 with Dependency Pruning:** We introduce a dynamic graph pruning mechanism that refines task dependencies in real time. Unlike static GATs, our method uses adaptive thresholds to prune low-priority edges. For example, in a drone swarm scenario, tasks like obstacle avoidance receive higher priority during navigation, while the low-impact sensor calibration edges are pruned.
- **Hierarchical quantization across task and model levels:** Our method applies adaptive scalar quantization of input features to preserve the topology of the graph while reducing the dimensionality. Simultaneously, model weights are compressed using dynamic mixed-precision quantization, where layers with low-rank and high-sensitivity matrices are prioritized.
- **Double Deep Q-Networks (DDQN):** We design a shared feature extractor trained on quantized task embeddings and combine it with personalized decision heads tailored to individual agents. This architecture supports both global learning and device-specific adaptation accommodating local constraints, such as varying battery levels or processor speeds. Our framework enables the deployment of demanding applications in resource-limited environments.

Although DDQN enables personalization, it does not inherently address decentralized coordination among multiple edge devices with heterogeneous capabilities. To overcome this, we extend our approach to a personalized federated DDQN with hierarchical quantization enabling scalable learning in semi-decentralized environments where each digital twin maintains a local Q-head while sharing a global representation network across the federation.

The remainder of this paper is structured as follows: Section II reviews related work. Section III presents the system model and problem formulation. Section IV details the proposed HiDeR-GQ framework. Section V evaluates performance using simulation results, and Section VI concludes the paper.

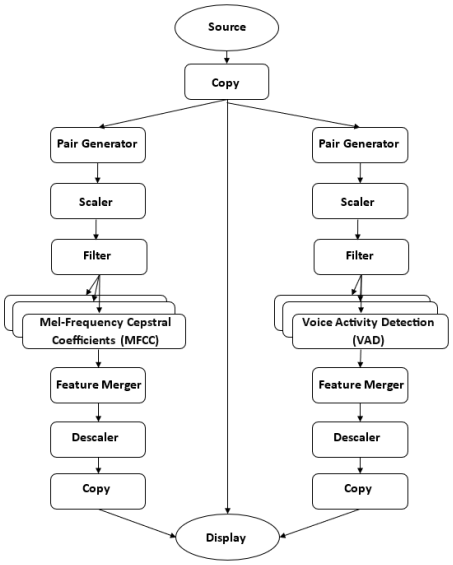


Fig. 1. Task Decomposition into Subtasks for Voice Recognition

II. Related Works

Task offloading in MEC has evolved rapidly, driven by the growing demand for high- performance computing in latency-sensitive applications. The emergence of digital twins and, more recently, HCs has redefined intelligent decision-making in real-time IoT systems. These virtual agents serve as continuously updated surrogates for physical devices, allowing predictive analytics, adaptive control, and autonomous task delegation. In the context of consumer electronics, such advancements support applications including smart wearables, augmented reality systems, and home automation, which must dynamically adapt to evolving conditions.

A wide range of approaches has been explored ranging from heuristic algorithms to DRL and graph neural networks (GNNs). Early studies emphasized rule-based heuristics and optimization frameworks to manage task offloading decisions. For example, Mach and Becvar [24] provided a comprehensive review of offloading architectures, emphasizing the trade-offs between local computation delay and offloading overhead. These foundational approaches positioned offloading as a key latency-mitigation strategy.

Subsequent works extended resource scheduling models under simplified assumptions. Luo et al. [25] proposed a task scheduling model that treated jobs as independent, ignoring inter-subtask dependencies. This simplification proved inadequate for complex real-world tasks such as video analytics and industrial monitoring, which are more accurately modeled as DAGs. Game-theoretic approaches [10], [26] and Lyapunov-based models [17], [27] enhanced resource efficiency under stable network conditions, often using static heuristics like critical path ranking. However, these solutions demonstrated limited adaptability under dynamic network conditions due to reliance on static priority rules and homogeneity assumptions [28]. A notable example is the DAG scheduling strategy by Topcuoglu et al. [29], which used fixed priority lists based on critical path analysis. While effective in controlled

environments, this method struggles in practical MEC contexts characterized by dynamic workloads, server variability, and fluctuating network latency. Thus, heuristic-driven solutions, while foundational, lack robustness in complex, real-time MEC environments. As heterogeneity increased across MEC infrastructures, DRL gained traction for its ability to adapt to evolving resource conditions. DRL agents learn offloading policies through interaction, improving responsiveness to changing workloads and infrastructure availability. An early demonstration of DRL's potential to reduce task latency through real-time offloading was presented by Shi et al. [15]. Building on this, Chen et al. [16] validated the advantages of DRL in scenarios with heterogeneous user requirements, showing that DRL-learned policies can continuously adapt to dynamic conditions. However, many of these studies overlooked task dependency modeling, which limits their applicability.

Recently, dependent task offloading has attracted attention as an emerging focus in the field of task offloading research [24]–[27]. Yet, many of these early DRL studies failed to model subtask dependencies, often treating them as isolated entities. Recent research has increasingly focused on incorporating DAG structures into DRL pipelines to capture task dependencies. Yan et al. [30] proposed a S2S model to linearize DAGs, improving dependency awareness but at the cost of high dimensionality and slow convergence. Zhou et al. [31] introduced GCNs to encode DAG structures more directly by producing node embeddings that informed a DRL agent's offloading decisions. Nonetheless, these models still relied on partially static scheduling mechanisms, which reduce adaptability and increase policy bias. GATs have since emerged as promising tools for capturing task dependencies without manually engineered features. Huang et al. [30] employed GCNs in an Actor-Critic architecture which showed that embedding the DAG structure yields better performance than purely sequential representations. Liu et al. [32], [33] extended this by applying GATs to dynamically reweight subtasks during decision-making, reducing reliance on rigid heuristics [34].

Despite these advances, two key challenges persist. Firstly, the existing GAT-based frameworks assume static task graphs, neglecting real-time structural updates. Secondly, reliance on high-precision floating-point arithmetic introduces overhead unsuited for memory and energy-constrained edge devices. While Zhang et al. [28] addressed multi-edge offloading with communication constraints, they did not incorporate quantization, a crucial step for improving efficiency in edge settings. Furthermore, the role of model compression and personalization remains underexplored in distributed learning, where data heterogeneity and privacy concerns restrict centralized training.

The proposed HiDeR-GQ framework addresses these gaps by integrating quantized GATv2 with dynamic dependency pruning (QGAT-DP), hierarchical quantization at both the task and model levels, and a Personalized Federated Double Deep Q-Network (PF-DDQN) for decentralized policy learning. The QGAT-DP module supports real-time graph reduction by removing contextually non-critical dependencies. This preserves core structural properties of the DAG while reducing computational complexity which makes graphs more tractable for real-time inference and learning. Another main component of HiDeR-GQ is hierarchical quantization, which operates at both task and model levels. Task-level features are compressed using scalar quantization, while layer-wise quantization of model parameters adjusts bit-depth based on sensitivity. This significantly reduces memory and bandwidth usage, enabling edge devices to run large models with limited resources.

To address the challenge of decentralized learning in heterogeneous and bandwidth-constrained edge environments [35], [36], we integrate a personalized federated DDQN (PF-DDQN) framework [37]. This extension allows each device to maintain a customized Q-head while sharing a globally synchronized feature extractor via periodic updates to an edge server. By combining federated learning with hierarchical quantization, we enable local adaptability and global generalization without centralized data sharing, to overcome practical limitations of existing edge learning approaches.

In addition to graph reduction and multi-level quantization, the federated DDQN structure supports local fine-tuning, making the framework resilient to varying compute capabilities, battery states, and network bandwidth. This hybrid approach accommodates diverse network conditions and varying computational capabilities across heterogeneous edge devices. An unified approach is poised to significantly reduce latency and energy consumption in data-intensive applications like drone swarm navigation, healthcare IoT monitoring, or smart grid management.

The synergy between quantized graph encoding, federated learning personalization, and real-time dependency pruning positions HiDeR-GQ as a scalable and resource-efficient solution for task offloading in dynamic MEC environments. Table I presents a comparative analysis of recent methodologies, outlining their architectural frameworks, identifying existing limitations, and summarizing their key technical contributions to the task offloading domain.

III. System Model

We consider a task offloading system enabled by holographic counterparts within a heterogeneous MEC environment. Each consumer IoT device such as a wearable health monitor, AR headset, or smart home controller is paired with a synchronized holographic agent located at the network edge. These agents act as intelligent virtual managers responsible for handling the computational workload of their respective devices while adhering to latency and resource constraints. The system employs the HiDeR-GQ framework, where "HiDeR" denotes hierarchical dependency reduction, and "GQ" represents quantized graph-based decision-making. This reflects the dual goals of reducing task dependency complexity and optimizing computation via quantization. Each application task is decomposed into a set of interdependent subtasks, structured as a DAG $G_i = (V_i, E_i)$, where V_i is the set of subtasks and E_i defines the precedence relationships. Such DAGs model real-world task pipelines, e.g., multi-stage activity recognition in AR. To efficiently manage these tasks key Innovations in HiDeR-GQ are:

TABLE I
COMPARATIVE ANALYSIS OF STATE-OF-THE-ART METHODS

Reference	Contribution	Model Used	Limitations
[24]	Presented MEC architectures and offloading strategies focused on reducing latency and energy consumption.	Rule-based methods, Binary Particle Swarm Optimization (BPSO), Constrained Markov Decision Process (CMDP), Lyapunov optimization.	Lacking adaptive, learning-based policies for dynamic environments; lacking integration of DRL.
[25]	Analyzed resource scheduling in edge computing, including offloading, allocation, and provisioning across layers.	Unified task model (delay, energy, cost); offloading ratio modeling; Amdahl's law; partial/binary offloading with resource optimization using mixed-integer linear programming (MILP) and metaheuristics.	No integration of DRL for learning optimal policies in dynamic, high-dimensional state spaces.
[15]	Proposed a vision for edge computing with key use cases (e.g., smart home, smart city), addressing challenges in data locality, programmability, and privacy.	Conceptual and architectural models; scenario-driven discussion without mathematical modeling.	Formal optimization or learning-based scheduling not included; adaptive decision-making using DRL in dynamic IoT settings also not addressed.
[26]	Developed a multi-hop cooperative offloading framework for industrial IoT (IIoT) edge–cloud using game theory to minimize energy and latency.	Multi-hop computation offloading modeled as a potential game; Nash equilibrium achieved using distributed algorithms with quality-of-service (QoS) guarantees.	Game-theoretic models are static and predefined; lack DRL-based scalability and adaptability for real-time, multi-agent coordination.
[28]	Dependent task offloading mechanism for cloud–edge–device collaboration, incorporating service caching awareness.	Binary optimization model considering service constraints, task dependencies; greedy-based scheduling algorithm.	Relies on static heuristics; lacks adaptability through DRL for varying workloads and dynamic task structures.
[32]	Introduced GA-DRL, a GNN-augmented deep reinforcement learning framework for DAG scheduling in dynamic vehicular networks.	GGAT for DAG feature extraction; DDQN for subtask allocation.	Designed for vehicular scenarios; limited generalizability to heterogeneous edge-cloud networks; lacks federated learning support.
[34]	DAG-DQN: A dependency-aware online task offloading approach for the Internet of Vehicles (IoV) leveraging DRL.	DAG-modeled task graphs; DRL with MDP formulation using Q-learning enhanced with task priority and dependencies.	No support for cross-device learning or federated extensions; scalability in dense multi-agent IoV scenarios remains unverified.
[35]	Analysis of federated learning paradigms focused on system-level constraints and open research issues.	Federated Averaging (FedAvg), client–server architecture, secure aggregation, differential privacy.	No incorporation of DRL for adaptive scheduling or dependency-aware task offloading in federated environments.

- Adaptive Graph Attention (GATv2) for dependency-aware learning and edge pruning.
- Hierarchical quantization applied to both task features and model parameters.
- A Personalized Federated Double Deep Q-Network for device-specific offloading decisions.

A. Overview Design

The system architecture comprises a set of mobile devices and multiple edge servers through stable wireless links. Each mobile device $m_i \in \{MD\} = \{m_1, m_2, \ldots, m_n\}$ is mirrored by an HC agent, which autonomously makes real-time offloading decisions by continuously monitoring device workload, subtask dependencies, and resource availability. The edge server set is defined as $\mathcal{E} = \{ES_1, ES_2, \ldots, ES_f\}$. Subtasks are offloaded to different ES_ω based on predicted execution benefit, while others are retained locally. Each device m_i is paired with an HC agent hosted at the edge. The agent monitors real-time device conditions and generates offloading decisions. Each task t_i generated by m_i is partitioned into subtasks $t_{i,j} = \{t_{i,j,1}, t_{i,j,2}, \ldots, t_{i,j,k_{max,i,j}}\}$, governed by a DAG G_i. The scheduling process respects the DAG's partial order. For example, if subtask $t_{i,j,k}$ depends on $t_{i,j,k'}$ the latter must complete and transmit its result before the former can begin. Figure 2 illustrates a sample DAG, where node color denotes subtask priority and size represents computational demand. In the example, $t_{i,j,1}$ is executed locally, while $t_{i,j,2}$ and $t_{i,j,3}$ are offloaded to different ES_ω.

To reduce computational overhead, adaptive graph attention (GATv2) is applied to refine G_i by pruning non-critical edges. Each edge attention score $\alpha_{i,j}$ is compared to a dynamic threshold τ and edges with $\alpha_{i,j} < \tau$ are removed. Subtask priorities $P_{i,j,k}$ are updated incrementally only when their change $\delta P_{i,j,k}$ exceeds a significance threshold ϵ. Additionally, each subtask feature vector $g_{i,j}$ are quantized to k^{bit} precision to minimize transmission and computation overhead.

$$\hat{\mathbf{g}}_{i,j} = \Delta \mathcal{N}\left(\frac{\mathbf{g}_{i,j} - \mu}{\Delta}\right) + \mu \tag{1}$$

where μ is the feature mean and Δ is the step size. This operation compresses the input data while preserving topological integrity of the DAG structure. A shared GATv2 encoder processes these quantized DAGs and generates global representations, while personalized decision heads adapt offloading policies to specific devices based on their local constraints. These device-specific heads ensure personalization while preserving global coordination across the federation. Finally, adaptive attention enables pruning of low-impact dependencies across users (i.e., when $(\alpha < \tau)$, allowing efficient aggregation of tasks for global scheduling. This multi-user pruning strategy

improves overall system throughput and supports scalable, context-aware offloading in federated MEC deployments.

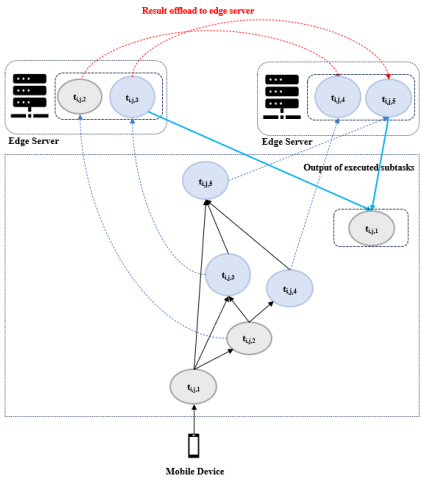


Fig. 2. Scheduling of a single task with subtasks executed locally and offloaded to edge servers

B. Communication Model

In the proposed MEC environment, each subtask$t_{i,j,k}$ may be executed locally on the mobile device m_i, or offloaded to one of the edge servers $ES_\omega \in \mathcal{E}$. The computation and communication costs vary accordingly.

1) Local Computing: The execution time for a subtask performed locally is determined by the quantized workload $\hat{w}_{i,j,k}$ derived from the subtask's quantized feature vector $\hat{g}_{i,j}$ and the local CPU frequency f_{cpu}:

$$T^{\text{local}}_{i,j,k} = \frac{\hat{w}_{i,j,k}}{f_{\text{cpu}} \times c} \tag{2}$$

where c represents the number of floating-point operations per cycle. The corresponding local cost is modeled as:

$$C^{\text{local}}_{i,j,k} = \lambda_{\text{local}} \cdot T^{\text{local}}_{i,j,k} \tag{3}$$

where λ_{local} is a tunable weight to balance computation cost versus latency.

2) Edge Offloading: Offloading incurs additional overhead. Specifically, the upload time depends on the size of the quantized subtask input $\hat{T}^{\text{data}}_{i,j,k}$ and the available uplink transmission rate r^{off}:

$$T^{\text{up}}_{i,j,k} = \frac{\hat{T}^{\text{data}}_{i,j,k}}{r^{\text{off}}(t_{i,j,k})} \tag{4}$$

Inter-subtask transfer time is determined by the dependencies among subtasks and the use of attention-based filtering from the predecessor set $\mathcal{P}(t_{i,j,k})$:

$$T^{\text{tr}}_{i,j,k} = \sum_{t_{i,j,k'} \in \mathcal{P}(t_{i,j,k})} \frac{\hat{T}^{data}_{i,j,k'}}{r^{tr}} \cdot \mathbb{I}(\alpha_{k,k'} \geq \tau) \tag{5}$$

where $\mathbb{I}$ is the indicator function, r^{tr} is the intra-edge transmission rate, and $\alpha_{k,k'}$ is the attention score between subtasks. The term τ is the attention threshold used in dependency pruning.

TABLE II
TABLE OF NOTATIONS

Symbol	Description
MD	Set of mobile devices $m_i \in \{MD\} = \{m_1, m_2, \ldots, m_n\}$
$\mathcal{E}$	Set of edge servers $ES_\omega \in \mathcal{E}$
t_i	Tasks generated by mobile device m_i.
$t_{i,j}$	j^{th} task from mobile device m_i
$t_{i,j,k}$	k^{th} subtask of $t_{i,j}$
G_i	DAG representing subtask dependencies m_i
$z_{i,j,k}$	Binary offloading decision: 1 if offloaded to edge, 0 if executed locally
$s_{i,j,k'}$	Binary indicator: 1 if $t_{i,j,k'}$ is a predecessor of $t_{i,j,k}$
$ST_{i,j,k}$	Start time of subtask $t_{i,j,k}$
$FT_{i,j,k}$	Finish time of subtask $t_{i,j,k}$
$\mathcal{Z}_i$	Embedding of task t_i generated via quantized GATv2
$\alpha_{i,j}$	Attention score between subtasks i and j
τ	Adaptive pruning threshold used in GAT-based dependency graph

C. Optimization Goal

The objective is to minimize the total system cost incurred by executing all subtasks across local and edge environments. The cost includes both computation time and communication overhead, and the policy must respect task dependency constraints defined by DAGs. The decision variable $z_{i,j,k} \in \{0,1\}$ denotes whether subtask $t_{i,j,k}$ is offloaded to edge server ES_ω:

$$\min \sum_{i,j,k} \left[z_{i,j,k} \cdot \hat{C}^{\text{edge}}_{i,j,k} + (1 - z_{i,j,k}) \cdot \hat{C}^{\text{local}}_{i,j,k} \right] \tag{6}$$

Subtasks must obey DAG-based dependencies. For a subtask to start, all its predecessors must have completed:

$$ST_{i,j,k} \geq \max_{t_{i,j,k'} \in \mathcal{P}(t_{i,j,k})} \left(FT_{i,j,k'} + \tau^{\text{delay}}_{k',k} \right) \tag{7}$$

where $ST_{i,j,k}$ and $FT_{i,j,k'}$ denote the start and finish times of subtask $t_{i,j,k}$ and $\tau^{\text{delay}}_{k',k}$ is the propagation delay incurred in transmitting results from predecessor $t_{i,j,k'}$ to successor $t_{i,j,k}$. The convergence of the quantized personalized DDQN is guaranteed if:

$$\|Q^* - \hat{Q}^*\|_\infty \leq \frac{\epsilon_Q}{1 - \gamma} \tag{8}$$

This condition guarantees convergence of the personalized federated DDQN's Q-function despite quantization, with error bounded by ϵ_Q and controlled discounting via γ.

subject to:

$$z_{i,j,k} \in \{0,1\} \quad \forall i,j,k \tag{9}$$

$$s_{i,j,k'} \in \{0,1\} \quad \forall i,j,k' \tag{10}$$

$$ST_{i,j,k} \geq \sum_{k'} s_{i,j,k'} \cdot \left(CT_{i,j,k'} + \tau^{\text{delay}}_{k',k} \right) \quad \forall i,j,k, \tag{11}$$

where $s_{i,j,k'} \in \{0,1\}$ indicates whether $t_{i,j,k'}$ is a predecessor of $t_{i,j,k}$ and $\tau^{\text{delay}}_{k',k}$ is the propagation delay between subtasks.

IV. THE HIDER-GQ FRAMEWORK

In this section, we present HiDeR-GQ (Hierarchical Dependency Reduction with quantized graph-based Q-learning), a novel framework that addresses the critical challenge of task offloading in heterogeneous MEC environments. Our approach stems from the insight that task performance is often influenced by task dependencies, which impact execution latency, resource utilization, and overall system efficiency. While modeling these dependencies as fully connected graphs can reveal meaningful execution patterns, it also leads to substantial computational overhead. To manage this trade-off, HiDeR-GQ balances the retention of meaningful task relationships with computational feasibility by selectively pruning less critical edges that minimally contribute to execution order or performance. This pruning reduces graph size, making it more tractable for both DRL algorithms and real-time offloading decisions.

A core component of HiDeR-GQ framework is hierarchical quantization, which is applied at both the task representation and model parameter levels. Input DAG features are quantized using scalar quantization, while the model weights are compressed via a layer-aware quantization strategy that adjusts bit precision based on sensitivity. This dual-level compression at both the feature and model level is referred to as hierarchical quantization in our framework. This approach reduces memory usage and transmission overhead, facilitating efficient execution on resource-constrained edge devices. The quantization strategy is formulated to preserve the contraction property of the Bellman operator, thereby ensuring that the approximate Q-value representations converge reliably under reinforcement learning. In addition to graph pruning and quantization, HiDeR-GQ integrates a personalized Federated Double Deep Q-Network (DDQN) to learn task offloading strategies tailored to each device's local context.

The architecture includes a shared global feature extractor for capturing general patterns across devices and device-specific output heads for personalization. This enables efficient policy learning without sacrificing adaptation to individual device constraints. Training is performed in a federated manner, where each device updates its local model using on-device data, preserving privacy by avoiding the transmission of raw data. Instead, only quantized model updates are shared with an edge server for periodic aggregation, ensuring global coordination while maintaining data privacy. These updates are synchronized intermittently to reduce communication cost and maintain model consistency. Overall, HiDeR-GQ provides a comprehensive, privacy-preserving, and resource-efficient solution to dependency-aware task offloading in MEC systems by combining federated learning, latency-aware dependency pruning, and adaptive hierarchical quantization.

A. Background

In DRL, the decision-making process is modeled as an MDP, represented as (S, A, P_a, R_a), where S and A represent the state and action spaces, respectively, and P_a is a transition probability matrix. The reward function is represented by R_a. The policy $p\,(a|s)$ denotes the probability of taking action a in state s. Trajectories sampled from the policy $\mathbb{T}p$ and comprise sequences $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, where s_0 is drawn from the initial state distribution P_0, and a_t and r_t denote the action and reward at time t.

Formally, the state-action value function $q_p(s_t, a_t)$ evaluates the return starting from state s_t and action a_t, defined as $[q_p(s_t, a_t) = \mathbb{E}p\,[\sum u = 0^{\infty}\gamma, r_t + u, |, s_t, a_t]]$. The goal of DRL is to discover the policy that maximizes the total return $[g_0 = \sum_{s_0} P_0(s_0)q_p(s_0, a)]$. DRL techniques are categorized into value-based and policy-based methods. Value-based methods seek the optimal value function for each state, expressed as $(v^*(s) = \max_a q^*(s, a)$ [38]. DQN uses neural networks to approximate $q(s, a)$, but suffers from overestimation bias and instability, especially in large, continuous or high-dimensional state-action spaces [39], [40].

The HiDeR-GQ framework is designed as a hybrid approach that integrates DDQN with GAT to effectively tackle these challenges. The DDQN reduces overestimation by decoupling action selection and evaluation, while GAT captures structured dependencies among subtasks modeled as DAGs. Additionally, hierarchical quantization is applied to reduce both computational complexity and memory usage in multi-device edge computing environments. This integration enables HiDeR-GQ to learn efficient, personalized offloading policies in highly dynamic settings without compromising performance or scalability.

B. HiDeR-GQ Design

The methodology is designed to address efficient decision making in resource-constrained environments by integrating two main algorithms:

1) Quantized GATv2 with Dependency Pruning: We extract compact task embeddings by combining feature quantization with latency-aware dependency pruning. This process proceeds in three stages: quantizing task features, applying dynamic latency-based pruning to filter non-essential edges, and computing graph attention over the pruned graph using quantized GATv2 weights. Each task t_i is represented by a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, which encodes data size, computational urgency, and resource requirements. To minimize memory and transmission costs, each feature vector is quantized to k^{bit} precision using dynamic range adaptation:

$$\mu_i = \frac{1}{d}\sum_{j=1}^{d}\mathbf{x}_{i,j}, \quad \Delta_i = \frac{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}{2^k - 1} \tag{12}$$

$$\hat{\mathbf{x}}_i = \Delta_i \cdot \mathcal{N}\left[\frac{\mathbf{x}_i - \mu_i}{\Delta_i}\right] + \mu_i \tag{13}$$

This approach scales features on a per-task basis, mitigating significant variations in magnitude across different tasks and eliminating spurious connections by pruning edges using the latency τ. The pairwise latency matrix is then computed using quantized features as follows:

$$L_{i,j} = f_{\text{latency}}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) \tag{14}$$

where latency function f_{latency} estimates the execution delay when tasks i and j are sequentially dependent. Leveraging quantized feature vectors $\hat{\mathbf{x}}$ ensures that these computations remain lightweight and scalable. To accommodate varying network conditions and dynamically evolving task dependencies, we employ a time-adaptive threshold $\tau_{\underline{t}}$. This threshold is updated at each time step $\underline{t}$ using an exponential moving average of observed inter-task latencies:

$$\tau_{\underline{t}} = \alpha \cdot \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} (L_{i,j})_{\underline{t}} + (1-\alpha) \cdot \tau_{\underline{t}-1} \tag{15}$$

where α is a smoothing parameter with $(0 < \alpha < 1)$, $(\mathcal{L}_{i,j})_{\underline{t}}$ is the latency estimate between task pair (i,j) at time $\underline{t}$, and $\tau_{\underline{t}-1}$ denotes the previous threshold. This formulation ensures that the pruning mechanism remains responsive to temporal changes while avoiding abrupt threshold shifts, effectively preserving only the most critical task dependencies. We implement a binary pruning strategy that removes low-impact dependencies by comparing the estimated pairwise latencies $(\mathcal{L}_{i,j})_{\underline{t}}$ to the dynamic threshold $\tau_{\underline{t}}$. The pruned adjacency matrix $A_{i,j}^{pruned} \epsilon \{0,1\}^{m\times n}$ defined element-wise as:

$$A_{i,j}^{\text{pruned}} = \begin{cases} A_{i,j}, & \text{if } \mathcal{L}_{i,j,t} \geq \tau_t \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

This operation removes relationships unlikely to provide latency benefits while preserving critical edges. The pruned DAG and quantized feature vectors are processed by a modified GATv2 with k^{bit} quantized parameters. To capture diverse aspects of task dependencies, GATv2 employs multiple attention heads parallel attention heads, each learning from a different subspace. For each node pair (i,j), the attention scores are computed as:

$$e_{i,j} = \text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}_q \hat{\mathbf{x}}_i \,\|\, \mathbf{W}_q \hat{\mathbf{x}}_j\right]\right) \tag{17}$$

where $W_q \in \mathbb{R}^{m\times d}$ is the linear transformation matrix with weights quantized to k^{bit} precision, $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \in \mathbb{R}^d, k^{bit}$ quantized task feature vector and $a \in \mathbb{R}^{2m}$. The activation function LeakyReLU introduces controlled non-linearity while allowing small gradients for negative values:

$$\text{LeakyReLU}(\xi) = \begin{cases} \xi, & \text{if } \xi \geq 0 \\ \epsilon \cdot \xi, & \text{otherwise} \end{cases} \quad \text{with } \epsilon \in (0,1) \tag{18}$$

In this formulation, $\xi = (a^{\intercal}\left[W_q\widehat{x_i} || W_q\widehat{x_j}\right])$, where a is a parameter vector, and $||$ denotes vector concatenation. The activation improves training stability by preventing neuron inactivity, especially when applied in federated settings with skewed or sparse input features. Next, attention scores are masked using the pruned adjacency matrix $A_{i,j}^{pruned}$:

$$\alpha_{i,j} = \frac{\exp(e_{i,j}) \cdot A_{i,j}^{\text{pruned}}}{\sum_{k\in\mathcal{N}_i} \exp(e_{i,k}) \cdot A_{i,k}^{\text{pruned}}} \tag{19}$$

This ensures that attention score is concentrated only on high-latency dependencies. The final embedding for node i is computed using an attention-based aggregation weighted of its neighbors' features:

$$\mathcal{Z}_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{i,j}\, \mathbf{W}_q \hat{\mathbf{x}}_j\right) \tag{20}$$

In this context, σ is a parametric activation of $ReLU\,(\theta) = max\,(0,\theta)$,where $\theta = \sum_{j\in\mathcal{N}_i} \alpha_{i,j} W_q \widehat{x_j}$ is the aggregated input to the node i. The ReLU activation promotes sparsity and ensures non-negativity in the learned representations, which is especially beneficial for preserving semantically significant features in quantized graph embeddings.

Algorithm 1 Quantized Graph Attention with Dependency Pruning (QGAT-DP)

Input: Task features $\mathbf{x} \in \mathbb{R}^{n\times d}$, adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, bit-width k, latency threshold $\tau_{\underline{t}}$
Output: Task embeddings $\mathbb{Z} \in \mathbb{R}^{n\times m}$
for $i = 1$ **to** n **do**
  Compute μ_i, Δ_i *(Eq. 12)*
  Quantize $\mathbf{x}_i \to \hat{\mathbf{x}}_i$ *(Eq. 13)*
end for
for all (i,j) such that $A_{i,j} = 1$ **do**
  Compute $L_{i,j} = f_{\text{latency}}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$ *(Eq. 14)*
  if $L_{i,j} \geq \tau_{\underline{t}}$ **then** *(Eq. 16)*
    $A_{i,j}^{\text{pruned}} \leftarrow 1$
  else
    $A_{i,j}^{\text{pruned}} \leftarrow 0$
  end if
end for
for all (i,j) such that $A_{ij}^{\text{pruned}} = 1$ **do**
  Calculate $e_{i,j}$ *(Eq. 17)*
end for
for $i = 1$ **to** n **do**
  Normalize $\alpha_{i,j}$ *(Eq. 19)*
end for
for $i = 1$ **to** n **do**
  Aggregate $\mathcal{Z}_i$ *(Eq. 20)*
end for
return $\mathbb{Z} \leftarrow [\mathcal{Z}_1, \mathcal{Z}_2 \ldots, \mathcal{Z}_n]$

2) Personalized Federated DDQN with Hierarchical Quantization (PF-DDQN): To support decentralized decision-making in mobile edge environments, the method outlined in Algorithm 2 combines personalized reinforcement learning with hierarchical quantization. Each device maintains a personalized head, while sharing a global feature extractor referred to as the shared head. The shared extractor learns general representations across devices, while device-specific heads refine them for local decisions. It processes input states into shared embeddings that are later refined by device-specific heads to support tailored decision-making. Periodic synchronization via an edge server ensures that the global model remains updated.

Hierarchical quantization is applied at both the task level (e.g., feature vectors) and model level (e.g., neural network weights) to reduce computational and memory overhead. This is essential in heterogeneous environments where devices

may differ significantly in storage capacity, compute power, and bandwidth. The framework integrates personalized model heads and hierarchical quantization to strike a balance between accurate Q-learning performance and efficient communication. Each device i maintains a local Q-function defined as:

$$Q_i(s,a) = f^i_{\text{personal}}\Big(f_{\text{shared}}(s;\theta_s);\theta^i_p\Big) \tag{21}$$

where f_{shared} is a shared feature extractor with parameters θ_s. It usually consists of early layers of a neural network that learns to represent features on all devices. f^i_{personal} is the device-specific head with parameters θ^i_p. The device-specific head, also referred to as the personalized head with parameters θ^i_p. This personalized head enables each device to fine-tune the shared representation $f_{\text{shared}}\,(s;\theta_s)$ to match its local context, which may vary in terms of state distribution and reward structure. It maps the shared feature representation to Q-values specific to each device's context, guiding task offloading decisions.

This design promotes better generalization across the federation while preserving performance for heterogeneous environments. The shared parameters θ_s capture common knowledge, while the personalized head θ^i_p allows local policy adaptation. This architecture improves sample efficiency by avoiding a one-size-fits-all policy and enabling each device to independently optimize its decision-making function based on its specific conditions. For local training and loss functions, each device collects local transitions and updates its Q-function parameters θ^i_p by minimizing a hybrid loss that combines temporal-difference (TD) error and a federated regularization term. Each device trains locally using a hybrid loss function combining the temporal-difference (TD) loss and a federated consistency regularization term (FedProx):

$$\mathcal{L}^i_{\text{TD}} = (Q_i(s,a) - y_i)^2 \tag{22}$$

$$\mathcal{L}^i_{\text{FedProx}} = \mu \cdot \|\theta^i_p - \theta_{\text{global}}\|^2 \tag{23}$$

$$\mathcal{L}^i_{\text{local}} = \mathcal{L}^i_{\text{TD}} + \mathcal{L}^i_{\text{FedProx}} \tag{24}$$

The DDQN mitigates the overestimation bias by using double network: one network to select the best action and another target network, denoted by θ^-_s to evaluate it. This helps stabilize training and makes Q targets more accurate. Local models can drift too far from the global model if local objectives diverge in federated learning. The FedProx term penalizes a larger deviation to maintain the update stable and aligned with global parameters. The DDQN target is defined as:

$$a^* = \arg\max_{a'} Q_i(s',a';\theta_s,\theta^i_p) \tag{25}$$

$$y_i = r + \gamma\, Q_i(s',a^*;\theta^-_s,\theta^i_p) \tag{26}$$

While quantization significantly improves system performance, communication efficiency and reduces model size, it also introduces approximation errors in Q-value estimation. Our quantization strategy is designed to preserve the contraction property of the Bellman operator to ensure the learning process remains stable despite these errors. Specifically, the Bellman operator $\mathcal{T}$ used in the Q-learning update (Eq. 8) remains a γ contraction under bounded quantization noise, satisfying the inequality:

$$\left\|\mathcal{T}\hat{Q} - \mathcal{T}Q\right\|_\infty \le \gamma \left\|\hat{Q} - Q\right\|_\infty \tag{27}$$

This inequality ensures that the quantized Q-values $\hat{Q}$ converge to a neighbourhood of the optimal Q-function Q^* by maintaining learning stability under quantized conditions. Thus, the Bellman update formulation in Equation (8) continues to drive convergence during personalized federated DDQN learning even when using low-precision model parameters [41]–[43] After local training, devices send their updates to an edge server for asynchronous aggregation. The updated global model is computed as:

$$\theta^{(\underline{t}+1)}_{\text{global}} = \frac{1}{|MD|}\sum_{m_i \in MD} \theta^{(\underline{t})}_{p,i} \tag{28}$$

The above equation represents a simple averaging approach, where devices $m_i \in MD$ updates at iteration $\underline{t}$. The aggregated parameters $\theta^{(\underline{t}+1)}_{\text{global}}$ act as a reference model for future local updates to ensure consistency throughout the federation. Transmitting full-precision parameters from many devices can saturate the bandwidth. Quantization compresses updates and enables more efficient communication without drastically sacrificing accuracy. In model-level quantization, the mean μ_θ and scale Δ_θ are selected such that most parameter values fall within a representable range with fewer bits. The continuous weights are discretized to the nearest quantized level by rounding.

$$\hat{\theta}^i_p = \Big[\frac{\theta^i_p - \mu_\theta}{\Delta_\theta} + \frac{1}{2}\Big].\Delta_\theta + \mu_\theta \tag{29}$$

This methodology outlines a comprehensive framework for dependency-aware decision making in semi-decentralized, multi-device environments. The integration of a quantized GATv2 for efficient task embedding extraction with a personalized federated DDQN for adaptive decision making ensures a robust and scalable solution. The application of hierarchical quantization and the convergence analysis demonstrate that the system can maintain optimal performance even under quantization noise.

Algorithm 2 Personalized Federated DDQN with Adaptive Quantization (PF-DDQN)

Input:
- Experience buffers $\{\mathcal{D}_i\}$ with tuples (s, a, r, s'), where $\mathbb{S}_{\underline{t}} = [S_{MD}, S_{ES}, \mathbb{Z}] \in \mathbb{S}'$ includes mobile device states, edge server states, and task graph embeddings
- Shared parameters θ_s, target network θ_s^-
- Initialized personalized heads $\{\theta_p^i\}$, global head θ_{global}^0
- Quantization threshold $R_{\text{threshold}}$,
- Hyperparameters $(\mu, \mathbf{k}, \mathfrak{C}, \gamma, \mathfrak{R})$: FedProx regularize, aggregation interval, sync interval, discount factor, total rounds

Output: Optimized θ_s^*, $\{\theta_p^{i*}\}$, and quantized weights $\mathbf{W}_{\text{quant}}^l$

for iteration $\mathfrak{t} = 1$ to T **do**
  for each device $m_i \in \{MD\}$ **in parallel do**
    Sample mini-batch $(s, a, r, s') \sim \mathcal{D}_i$
    Compute target action using DDQN selection (Eq.25)
    Calculate target value y_i (Eq.26)
    Compute personalized TD + FedProx loss:

$$\mathcal{L}_i \leftarrow \|Q_i(s,a) - y_i\|^2 + \mu\|\theta_p^i - \theta_{\text{global}}^{(\mathfrak{t})}\|^2$$

    Update shared and personalized parameters via gradient descent:

$$(\theta_s, \theta_p^i) \leftarrow \theta_s, \theta_p^i - \eta\nabla\mathcal{L}_i$$

    for each layer l in local Q-network **do**
      if rank$(\mathbf{W}^l) < R_{\text{threshold}}$ **then**
        Quantize weights:

$$(\mathbf{W}_{\text{quant}}^l \leftarrow \Delta\left\lfloor\frac{\mathbf{W}^l}{\Delta}\right\rfloor,$$

$$\Delta = \frac{\max(|\mathbf{W}^l|)}{2^k - 1}$$

      else
        $\mathbf{W}_{\text{quant}}^l \leftarrow \mathbf{W}^l$
      end if
    end for
    Backpropagate through $\mathbf{W}_{\text{quant}}^l$ using straight through estimator (STE)
  end for
  if $t \bmod \mathbf{k} = 0$ **then**
    Aggregate personalized heads:

$$\theta_{\text{global}}^{(t+1)} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\theta_p^i$$

  end if
  if $t \bmod \mathfrak{C} = 0$ **then**
    Sync target network:

$$\theta_s^- \leftarrow \theta_s$$

  end if
end for

C. The Task Offloading Model

To solve the task offloading problem using DRL, we model the environment as an MDPdefined by a tuple comprising the space, action space, and reward function.

1) State: In the proposed MEC system, effective task offloading requires the reinforcement learning agent to observe a comprehensive and structured system state that reflects the current conditions across mobile devices, edge servers, and task dependencies. Each mobile device $m_i \in MD = \{m_1, m_2, \ldots, m_n\}$ is described by a set of dynamic attributes including its local computation capability $\mathfrak{f}_{m_i}$, remaining battery level $\mathfrak{b}_{\mathfrak{m}_i}$, and connectivity strength $\mathfrak{c}_{\mathfrak{m}_i}$. These components form the per-device state vector $S_{m_i} = \{\mathfrak{f}_{m_i}, \mathfrak{b}_{\mathfrak{m}_i}, \mathfrak{c}_{\mathfrak{m}_i}\}$. The collective state of all mobile devices is denoted as $S_{MD} = \{S_{m_1}, S_{m_2}, \ldots.., S_{m_n}\}$. Similarly, each edge server $ES_\omega \in \mathcal{E} = \{ES_1, ES_2, \ldots, ES_f\}$ is characterized by its available computational resources $\mathfrak{f}_{ES_\omega}$, operational or energy cost $\mathfrak{p}_{ES_\omega}$, transmission power $\mathfrak{tr}_{ES_\omega}$, and bandwidth $\mathfrak{f}_{ES_\omega}$. These values comprise the server state $S_{ES_\omega} = \{\mathfrak{f}_{ES_\omega}, \mathfrak{p}_{ES_\omega}, \mathfrak{tr}_{ES_\omega}, \mathfrak{bw}_{ES_\omega}\}$, and the joint edge server state is denoted by $S_{\mathcal{E}} = \{S_{ES_1}, S_{ES_2}, \ldots., S_{ES_f}\}$. In addition to device and server states, the system includes a task dependency representation encoded through a graph-based quantized embedding. The task graph is modeled as a directed acyclic graph (DAG) where each node represents a subtask, and each edge denotes a dependency. Using a quantized GATv2 mechanism, the system computes embeddings $\mathbb{Z} = \{\mathcal{Z}_1, \mathcal{Z}_2 \ldots, \mathcal{Z}_n\}$, where each embedding , $\mathcal{Z}_i$ represents subtask t_i and encapsulates its position, dependencies, and workload context in the graph. These embeddings serve as the task-level input features for decision-making. The full system state observed at time step $\underline{t}$ is therefore a combination of mobile device states, edge server states, and task graph embeddings. This is formally defined as:

$$\mathbb{S}_{\underline{t}} = [S_{MD}, S_{ES}, \mathbb{Z}] \in \mathbb{S}' \tag{30}$$

2) Action Space: The agent observes the current system state and selects an action that determines whether to execute a subtask locally or offload it to one of the available edge servers. For each subtask $t_{i,j,k}$, the action space is defined as:

$$A_{\underline{t}} = \{a_0, a_1, a_2, ..., a_f, a_{local}\} \tag{31}$$

3) Reward function: The reward function directs the reinforcement learning agent toward minimizing the overall system cost by discouraging decisions that incur high communication or computation overhead. At each time step $\underline{t}$, the agent receives a reward $r_{\underline{t}}$ that reflects the negative cost of its offloading decision. This cost incorporates both local and edge execution costs, defined by:

$$r_{\underline{t}} = -\min\sum_{i,j,k}\left[z_{i,j,k}\cdot\hat{C}_{i,j,k}^{\text{edge}} + (1 - z_{i,j,k})\cdot\hat{C}_{i,j,k}^{\text{local}}\right] \tag{32}$$

This reward structure aligns with the optimization goal (Eq. 6), which encourage the agent to make offloading decisions that minimize the sum of edge and local processing costs.

V. Results

A. Simulation Setup and model training metrics

To validate the performance of intelligent real-time decision-making in holographic counterparts for consumer IoT environments, we conduct extensive simulations with various device and task configurations. Task workflows were synthetically generated and modeled as Barabási–Albert (BA) graphs, representing dependencies among subtasks $t_{i,j,k}$ derived from tasks $t_{i,j}$ generated by mobile devices $m_i \in MD$.

Each task $t_{i,j}$ is characterized by a six-dimensional feature vector $\mathbf{x}_i$ including attributes such as data size (ranging between $10KB$ and $500KB$) and CPU cycles (between 1 and 10×10^6 cycles). These attributes were sampled from uniform distributions to model the heterogeneous computational demands typical of edge computing scenarios, where subtasks vary significantly in both resource requirements and urgency.

To adaptively prune non-critical under dynamic network conditions, we introduced a latency-driven threshold τ updated at each time step $\underline{t}$ using exponential moving average with $\alpha = 0.1$, and an initial value of $20ms$. A pairwise latency matrix L is computed using quantized subtask embeddings $\mathcal{Z}_{i,j,k}$ generated via a quantized GATv2 with $8 - bit$ precision. This matrix determines which edges are retained in the dependency graph, based on the threshold τ. HiDeR-GQ operates within a heterogeneous MEC environment comprising mobile devices $m_i \in MD$ (CPU: 1–4GHz, RAM: 2–8GB, Battery: 2000–5000mAh) and edge servers $ES_\omega \in \mathcal{E}$(CPU: 8–16$GHz$, RAM: 32–64$GB$). These devices interact under dynamic network conditions with bandwidth ranging from 10 to $50Mbps$ and latency varying between 5 and $50ms$, realistically representing hardware and wireless variability in practical IoT deployments.

The quantized GATv2 employs 8 attention heads and a hidden dimension of 128, where the multi-head mechanism captures diverse dependency patterns among subtasks $t_{i,j,k}$. Using quantized weights W_q at $4 - bit$ precision reduces memory consumption, while the increased hidden dimension enhances the representational capacity relative to smaller configurations. Features are quantized to $8 - bit$ precision, and weights to $4 - bit$, leading to a 4 reduction in memory usage.

In training the federated DDQN, we selected a learning rate of 10^{-4} to ensure stable parameter updates across heterogeneous devices m_i. Empirical tests confirmed that this rate prevents the divergence observed at higher learning rates 10^{-3} and avoids slow convergence at lower rates 10^{-5}. A batch size of 64 was chosen to balance gradient stability with device memory constraints, outperforming alternative batch sizes tested (32 and 128).

The DDQN's target network synchronization occurs every 200 iterations to balance exploration and exploitation. To manage local model divergence while preserving personalization, the FedProx penalty term was set to $\mu = 0.01$. Additionally, an epsilon-greedy policy with ϵ decaying linearly from 1.0 to 0.1 over 10^{-4} steps was employed, promoting robust initial exploration and later exploitation for policy refinement.

These hyperparameters significantly impact the training efficacy and convergence of HiDeR-GQ. A detailed summary of the experimental parameters used in our evaluations is presented in Table III.

B. Performance Metrics Analysis

We simulate an MEC environment involving 5–10 mobile devices $m_i \in MD$ and 2–3 edge servers $ES_\omega \in \mathcal{E}$, with each application represented as DAG comprising of 20–50 interdependent tasks $t_{i,j}$. To evaluate HiDeR-GQ, we compare it against the following baseline strategies:

1) Standard Double DQN (DDQN): A conventional DRL offloading agent that employs Double DQN but lacks explicit modeling of task dependency structures. This baseline operates in a flat state space, learning whether to offload or execute tasks locally without considering the DAG. In our evaluation, each device either independently trains its own DDQN or relies on a centralized DDQN that makes offloading decisions for all devices, without personalization [44].

2) Graph RL without Quantization: This variant retains the DAG-aware GNN state representation and hierarchical DRL policy but omits hierarchical quantization. It reflects a decentralized graph-based RL approach that communicates full-precision parameters across devices. The purpose is to isolate and evaluate the impact of quantization on system efficiency and learning performance.

3) Federated DDQN (No Personalization): In this setup, all devices collaboratively train a shared DDQN model. Devices periodically upload local model parameters to a central server for aggregation (e.g., via FedAvg), and the aggregated global model is redistributed to all participants. Although this approach enables cross-device knowledge sharing and improves learning convergence and it introduces considerable communication overhead due to full-precision model synchronization.

4) Local-Only Heuristic: A rule-based baseline in which all tasks are executed locally on each mobile device, without any offloading. This strategy eliminates network communication cost; however, it may result in high execution latency due to resource-constrained devices.

5) Edge-Only Heuristic: In contrast, this fixed policy offloads all tasks to an edge server whenever feasible. This can reduce execution time per task due to powerful edge resources but incurs network delay for every task and may overload the edge.

All learning-based approaches use comparable neural network sizes for fairness. We train the DRL models over many episodes; in each episode a random set of task DAGs is generated for each device and the agent's decisions yield a cumulative reward. We define reward as inversely proportional to latency and deadline violations i.e., lower task completion times and meeting deadlines yield higher reward. We evaluate performance using several key metrics: average task completion time, training convergence, communication overhead and the deadline violation rate (percentage of tasks/DAGs that fail to meet their deadlines).

C. Training Convergence (Cumulative Reward)

The higher reward indicates better performance (lower latency and fewer violations). HiDeR-GQ converges to a higher

TABLE III
SYSTEM CONFIGURATION AND HYPERPARAMETERS USED IN THE HIDER-GQ FRAMEWORK

Category	Parameter	Value/Range	Description
Mobile Edge Devices	CPU	$1-4GHz$	Computational capability of mobile devices
	RAM	$2-8GB$	Memory constraints for local task execution
	Battery	$2000-5000mAh$	Energy capacity influencing offloading decisions
Edge Servers	CPU	$8-16GHz$	High-performance computation for offloaded tasks
	RAM	$32-64GB$	Parallel processing capacity for resource-intensive workflows
Network	Bandwidth	$10-50Mbps$	Wireless network throughput
	Latency	$5-50ms$	Communication delay between devices and edge servers
	Connectivity Strength	$\mathcal{U}(0.5, 1.0)$	Normalized link reliability (0: weak, 1: strong)
Task Generation	Number of Tasks (n)	$(20-50)$	Total tasks in the Barabási–Albert (BA) graph
	Preferential Attachment (e)	2	Edges per new task during BA graph growth
	Data Size	$[10, 500]KB$	Task input size ($\mathcal{U}(10, 500)$)
	CPU Cycles	$[1, 10]MB$	Computational load per task ($\mathcal{U}(1\times10^6, 10\times10^6)$)
	Deadline	$[1, 5]sec$	Task completion urgency ($\mathcal{U}(1, 5)$)
Quantized GATv2	Attention Heads	8	Multi-head attention for dependency modeling
	Hidden Dimension	128	Embedding size for task features
	Feature Quantization	$8-bit$	Compressed task features for memory efficiency
	Weight Quantization	$4-bit$	Compressed model weights
	Initial Pruning Threshold (τ_0)	$20ms$	Starting latency cutoff for edge retention
Federated DDQN	Learning Rate (α)	1×10^{-4}	Step size for Q-network optimization
	Discount Factor (γ)	0.95	Future reward discounting
	Batch Size	64	Transitions sampled from replay buffer
	Target Update Interval (C)	200 iterations	Frequency for syncing target and online networks
	FedProx Penalty (μ)	0.01	Regularization to balance local-global model divergence
	Epsilon Decay (ϵ)	1.0	Exploration-exploitation schedule
Dynamic Pruning	Smoothing Factor (α)	0.1	Responsiveness to latency fluctuations

reward more quickly. In the convergence plot figure 3, we see that the HiDeR-GQ agent's reward grows rapidly and stabilizes at the highest level, indicating that it learns an effective offloading policy within relatively few episodes. This rapid convergence is attributed to the framework's graph-based state representation and task-specific action hierarchy. By leveraging a GNN to encode the DAG dependencies, the agent effectively identifies and prioritizes critical subtasks for execution or offloading. Results show that HiDeR-GQ achieves near-optimal rewards by episode 50, demonstrating high sample efficiency. Overall, the convergence analysis validates that HiDeR-GQ not only outperforms baselines in final performance but also requires significantly fewer training iterations.

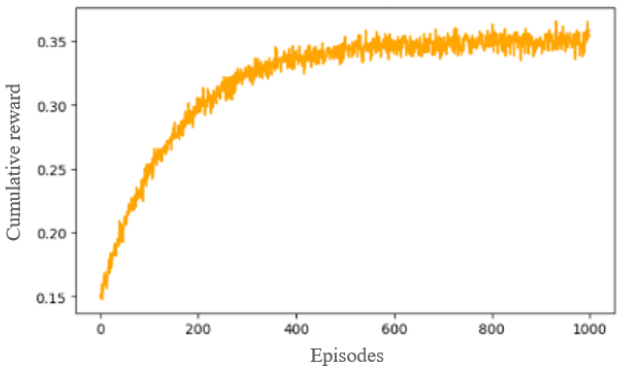


Fig. 3. Progression of Cumulative Reward Over Training Episodes

D. Loss analysis

The training loss curve initially shows a high error, which rapidly decreases within the first 50–100 episodes (Figure 4. This behavior reflects a learning phase where the agent corrects large Q-value predictions. By approximately episode 200, the loss stabilizes near zero and flattens out, suggesting that the agent has reached training convergence and is no longer undergoing major updates. The two overlapping loss curves likely represent slightly different training runs or components that converge to similar performance levels. This consistent reduction in training loss further confirms the stability and effectiveness of the HiDeR-GQ learning process.

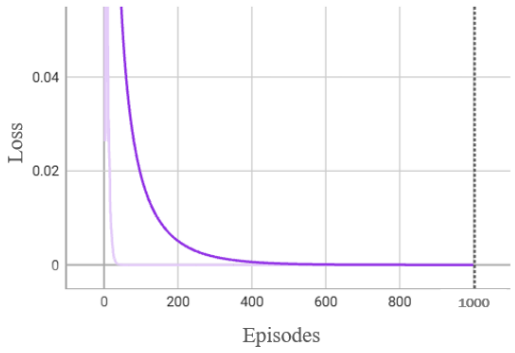


Fig. 4. Convergence Behavior of Training Loss Across Episodes

E. Latency and Communication Cost

Average task completion time per DAG under each method is shown below (lower is better). HiDeR-GQ achieves the lowest latency, while local-only execution is the worst. DRL-based offloading significantly outperforms static heuristics.

Figure 5(a) presents the average task completion times for each method. As expected, the Local-only baseline has the

highest latency ($\sim 100ms$) because all tasks are computed on resource-constrained devices. The Edge-only approach reduces latency modestly ($\sim 80ms$) by offloading to faster servers, but network transmission delays limit the gains. The standard DDQN agent approach achieve significantly lower latencies by intelligently selecting which tasks to offload. A standard DDQN, despite lacking DAG awareness, performs better than heuristic strategies ($\sim 70ms$) and shows evidence of learning some beneficial offloading (e.g., offloading large tasks and keeping small ones local). The Federated DDQN performed better ($\sim 60ms$) by leveraging shared learning across devices, but its policy is still limited by not understanding task dependencies deeply.

The Graph RL agent without quantization and HiDeR-GQ achieved the best performance ($\sim$ 50–$55ms$), marking a 30% improvement over standard DDQNThe latency difference between HiDeR-GQ and Graph RL is minimal ($45ms$ vs.$50ms$). This indicates that quantization does not degrade decision quality. HiDeR-GQ's slight advantage likely stems from its hierarchical optimization, which jointly considers task placement and resource allocation. Importantly, these latency gains are achieved without incurring high communication overhead. In federated RL, exchanging model updates can be costly. We measured the total communication volume of model parameters exchanged during training between devices and servers. The communication overhead model updates data exchanged during training for each method. HiDeR-GQ's quantized updates drastically cut communication costs, compared to the unquantized federated methods. Local-only, edge-only, and a standalone DDQN incur essentially no model communication. Figure 5(b) highlights a major advantage of HiDeR- GQ: communication efficiency. The local-only, edge-only, and single-agent DDQN baselines have zero model communication costs since they do not share any learning across devices.

The Federated DDQN baseline periodically shares a full DQN model. In our simulation, this amounted to about $8MB$ of data exchanged per training round (shown as the bar for "Fed DDQN"). The Graph RL (no quantization) method was even more communication-heavy ($\sim$ $12MB$) because the GNN-based model is larger and likely requires more frequent synchronization to converge. By contrast, HiDeR-GQ (with quantization) transmitted far less: roughly $3MB$ in our setup, which is a 75% reduction in communication load compared to the uncompressed Graph RL. This matches expectations from federated learning literature, quantifying model updates (e.g. $8-bit$ vs $32-bit$ floats) can cut bandwidth by 4 or more. For example, one study reported over 93% reduction in communication overhead with quantized federated training, with virtually no loss in accuracy.

In our results, HiDeR-GQ's $8-bit$ quantization compresses updates to 25% of their original size. This efficiency is critical in MEC settings where devices may connect over wireless links with limited throughput. HiDeR-GQ delivers superior task latency performance and maintains a low communication footprint during learning. HiDeR-GQ demonstrates a practical approach by using quantization to make federated RL viable in bandwidth-constrained environments.

F. Effect of Quantization on Performance

We further examine how the quantization level in HiDeR-GQ affects performance. Quantization involves using lower bit-width representations for model parameters and updates (e.g. $8-bit$ integers instead of 32-bit floats). There is an inherent trade-off: using fewer bits reduces data transmission size and communication cost, but aggressive quantization may degrade learning accuracy or hinder convergence due to reduced precision. We varied the bit-width used in HiDeR-GQ from no quantization $32-bit$ down to $4-bit$ and measured the final achieved performance and the communication overhead. Performance and overhead are shown relative to the unquantized $32-bit$ baseline.

In Figure 6(a), the blue line represents the final achieved performance (cumulative reward—higher is better), while the red line indicates communication overhead (lower is better). Using $8-bit$ quantization retains 95–100% of the performance while reducing communication to 25%. As the bit-width decreases from 32 to 16 to 8 bits, the communication overhead drops proportionally, while performance remains nearly unchanged. Our $8-bit$ setting yields 25% of the data size of $32-bit$. The performance, on the other hand, remains almost flat through 16-bit and $8-bit$, there is negligible loss $8-bit$ achieved 98% of the reward and latency performance of $32-bit$ in our tests.

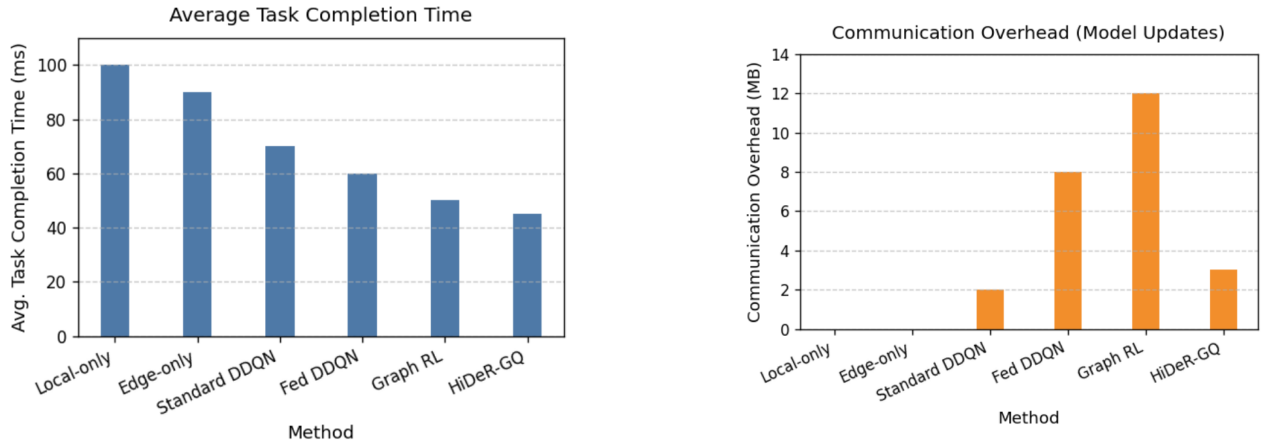
However, pushing quantization to extremes $4-bit$ started to affect performance noticeably. At $4-bit$, the final reward dropped to 80–85% of the baseline and the latency improvement reduced accordingly. In Figure 6(a), this is reflected as a noticeable dip in the blue line. While 4-bit communication overhead is extremely low only 12% of the uncompressed model the performance trade-off becomes significant. In some runs, HiDeR-GQ agents with 4-bit quantization diverged early or converged to a suboptimal policy. In contrast, 8-bit quantization proved robust, achieving strong compression with minimal accuracy loss. It offers an effective balance between model compactness and learning fidelity, making it the preferred setting for HiDeR-GQ in bandwidth-constrained environments.

G. Deadline Violation Rate

We compare the reliability of each method in meeting application deadlines. Each task has a deadline, we track the percentage of tasks that fail to complete within their allotted time under each strategy. This metric is defined as the deadline violation rate.

HiDeR-GQ significantly reduces deadline violations compared to baseline approaches. As shown in Figure 6(b), the trends mirror the latency results. The Local-only approach has the highest deadline miss rate 30% in our test and many tasks cannot finish in time when only the slow device CPU is used, especially for large DAGs. The Edge-only method performs moderately better, reducing violations to approximately 15%, as offloading to faster servers helps meet deadlines.

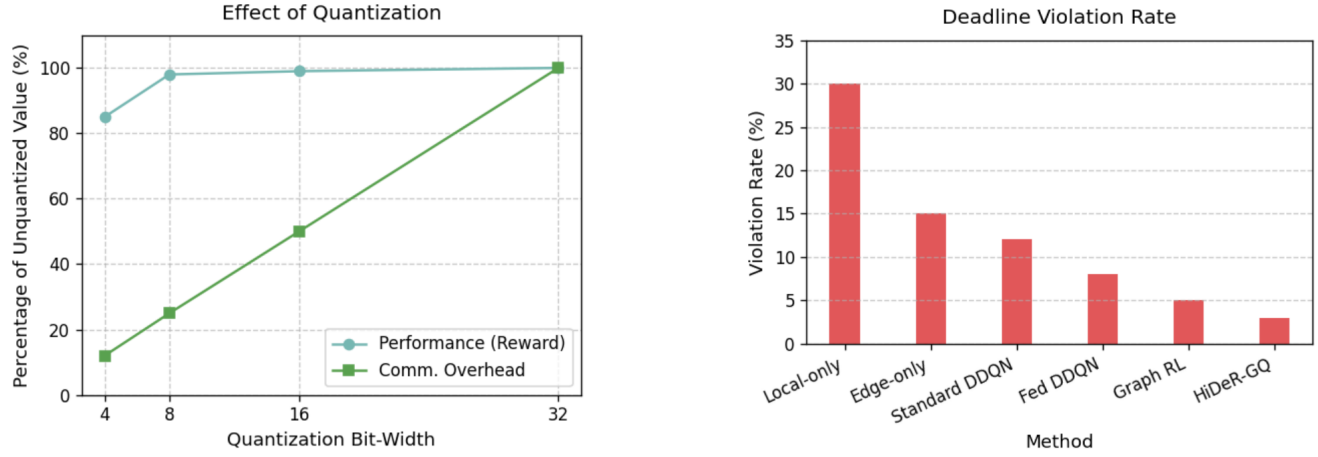
However, it still suffers when network latency or edge queuing delays are significant. The standard DDQN baseline learns to offload some tasks bringing the miss rate down to 12%. Federated DDQN improves further to 8%, benefiting



(a) Task completion

(b) Communication overhead

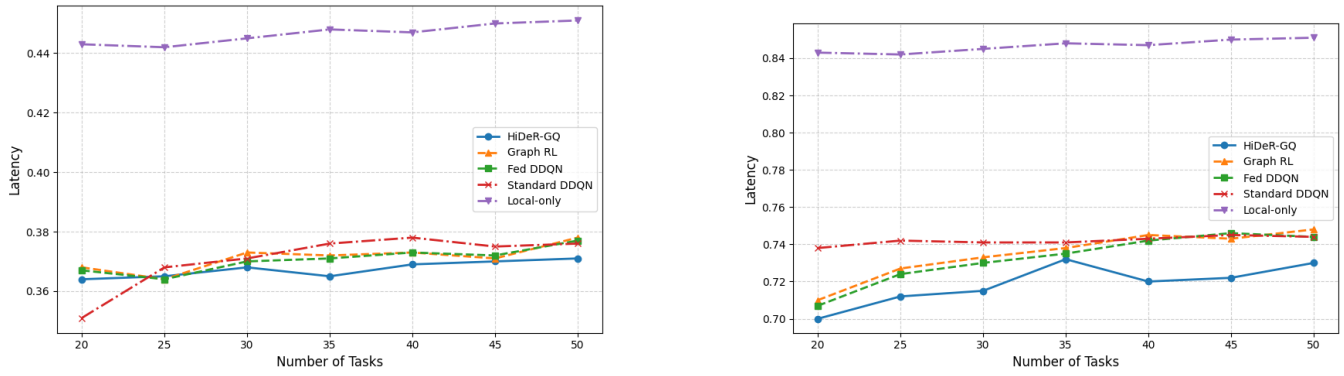
Fig. 5. Comparative Analysis of Average Task Completion Time and Communication Overhead Across Offloading Methods



(a) Effect of Quantization on Performance

(b) Deadline Violation Rate

Fig. 6. Effect of Quantization Bit-Width on Performance, Communication Overhead, and Deadline Violation Rate



(a) Latency vs. task numbers (20) for m=5 and ES=2

(b) Latency vs. task numbers (50) for m=10 and ES=5

Fig. 7. Latency Sensitivity to Task Volume Under Varying Edge Server and User Configurations

from shared knowledge across devices. The Graph RL agent (without quantization) reduces violations to 5%, while HiDeR-GQ achieves the lowest rate at just 3%. This demonstrates that the HiDeR-GQ policy enables nearly all tasks to meet their deadlines under varying conditions.

H. Impact of Task Number on Latency

We evaluate system latency as the number of tasks increases from 200 to 800, considering two deployment scenarios: 5 mobile devices with 2 edge servers, and 10 mobile devices with 5 edge servers (see Figure 7(a) and 7(b)). It maintains

the lowest latency as the number of tasks increases and achieves the best performance in minimizing system cost. Its ability to maintains the lowest latency as the number of tasks increases but still achieves the best performance in minimizing the system cost. This superior performance highlights the model's robustness in managing both resource constraints and task dependencies. The latency reduction trend remains stable across the different numbers of tasks and even slightly better with the larger workload. This result confirms that HiDeR-GQ not only scales well with workload but also maintains performance under demanding multi-device, multi-server conditions.

VI. Conclusion

In summary, this paper presents HiDeR-GQ, a novel Hierarchical Dependency Reduction and quantized graph Q-learning framework for optimizing task offloading in federated MEC environments. The proposed approach holistically combines graph-based DRL with federated learning and model quantization to address the unique challenges of offloading interdependent tasks. The primary contributions include a GATv2-based Q-network for effective encoding of task dependency graphs, a personalized federated DDQN algorithm that accelerates convergence and preserves privacy across distributed edge nodes, and a latency-aware dependency pruning strategy that minimizes end-to-end delays by intelligently relaxing or removing low-impact task dependencies.

Simulation results demonstrate that HiDeR-GQ enables context-aware, low-latency decision-making for smart, edge-connected systems, such as wearables, AR devices, and mobile sensors. The framework consistently outperforms baseline schemes in terms of task completion latency and communication overhead, achieving up to 20–30% latency reduction. Although energy consumption was not directly measured, the observed reductions in computation time and communication overhead suggest potential improvements in energy efficiency, particularly for battery-constrained mobile devices. Quantization and hierarchical task processing further enhanced communication and computational efficiency. HiDeR-GQ required fewer communication rounds and demonstrated faster policy convergence, making it well-suited for resource-constrained edge networks. These results highlight the practical impact of our contributions, demonstrating that structured task knowledge and personalized federated learning can enhance task offloading in dynamic mobile edge scenarios.

There are several promising directions to extend this work. Adaptive quantization methods could further optimize the trade-off between model update size and learning accuracy under varying network conditions. Additionally, incorporating multi-agent coordination mechanisms would allow multiple edge servers or mobile devices to jointly manage task dependencies across distributed workflows. Finally, fine-grained personalized federated learning techniques can be applied to enable each edge node's Q-network better adapts to its specific workload and hardware constraints. By pursuing these directions, HiDeR-GQ can evolve into a robust and scalable framework for next-generation intelligent consumer edge computing systems.

References

[1] J.-H. Syu, J. C.-W. Lin, G. Srivastava, and K. Yu, "A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 1023–1033, Nov. 2023.

[2] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6g: Vision, architectural trends, and future directions," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74–80, 2022.

[3] C. Xu, Z. Tang, H. Yu, P. Zeng, and L. Kong, "Digital twin-driven collaborative scheduling for heterogeneous task and edge-end resource via multi-agent deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3056–3069, 2023.

[4] B. Tan, L. Ai, M. Wang, and J. Wang, "Toward a task offloading framework based on cyber digital twins in mobile edge computing," *IEEE Wireless Communications*, vol. 30, no. 3, pp. 157–162, 2023.

[5] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk, and T. Q. Duong, "Digital twin-aided intelligent offloading with edge selection in mobile edge computing," *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 806–810, 2022.

[6] Y. Wang, J. Fang, Y. Cheng, H. She, Y. Guo, and G. Zheng, "Cooperative end-edge-cloud computing and resource allocation for digital twin enabled 6g industrial iot," *IEEE Journal of Selected Topics in Signal Processing*, vol. 18, no. 1, pp. 124–137, 2023.

[7] E. Bozkaya, T. Bilen, M. Erel-Özçevik, and Y. Özçevik, "Energy-aware task scheduling for digital twin edge networks in 6g," in *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*. IEEE, 2023, pp. 1–6.

[8] X. Li, B. Chen, J. Fan, J. Kang, J. Ye, X. Wang, and D. Niyato, "Cloud-edge-end collaborative intelligent service computation offloading: A digital twin driven edge coalition approach for industrial iot," *IEEE Transactions on Network and Service Management*, vol. 21, no. 6, pp. 6318–6336, 2024.

[9] Y. Chen, W. Gu, J. Xu, Y. Zhang, and G. Min, "Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning," *China Communications*, vol. 20, no. 11, pp. 164–175, 2023.

[10] Z. Zhang, Y. Li, C. Huang, Q. Guo, C. Yuen, and Y. L. Guan, "Dnn-aided block sparse bayesian learning for user activity detection and channel estimation in grant-free non-orthogonal random access," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 000–12 012, 2019.

[11] H. Yu, Z.-H. Tan, Z. Ma, R. Martin, and J. Guo, "Spoofing detection in automatic speaker verification systems using dnn classifiers and dynamic acoustic features," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4633–4644, 2017.

[12] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[13] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[14] J. Liu, Y. Zhang, J. Ren, and Y. Zhang, "Auction-based dependent task offloading for iot users in edge clouds," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 4907–4921, 2023.

[15] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[16] X. Chen, J. Cao, Y. Sahni, S. Jiang, and Z. Liang, "Dynamic task offloading in edge computing based on dependency-aware reinforcement learning," *IEEE Transactions on Cloud Computing*, vol. 12, no. 2, pp. 594–608, Apr. 2024.

[17] Y. Li, X. Ge, B. Lei, X. Zhang, and W. Wang, "Joint task partitioning and parallel scheduling in device-assisted mobile edge networks," *IEEE Internet of Things Journal*, vol. 11, no. 8, pp. 14 058–14 075, 2024.

[18] Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, and F. Yang, "Dependency-aware task scheduling in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4961–4971, 2020.

[19] J. Fang, D. Qu, H. Chen, and Y. Liu, "Dependency-aware dynamic task offloading based on deep reinforcement learning in mobile-edge computing," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 1403–1415, 2023.

[20] L. X. Nguyen, Y. K. Tun, T. N. Dang, Y. M. Park, Z. Han, and C. S. Hong, "Dependency tasks offloading and communication resource allocation in collaborative uav networks: A metaheuristic approach," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9062–9076, 2023.

[21] Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11, pp. 3245–3262, 2021.

[22] A. Abid, P. Sinha, A. Harpale, J. Gichoya, and S. Purkayastha, "Optimizing medical image classification models for edge devices," in *Distributed Computing and Artificial Intelligence, Volume 1: 18th International Conference 18*. Springer, 2022, pp. 77–87.

[23] Z. Guan, H. Huang, Y. Su, H. Huang, N. Wong, and H. Yu, "Aptq: Attention-aware post-training mixed-precision quantization for large language models," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

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

[25] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[26] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[27] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "Eedto: An energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2020.

[28] J. Zhang, J. Chen, X. Bao, C. Liu, P. Yuan, X. Zhang, and S. Wang, "Dependent task offloading mechanism for cloud–edge–device collaboration," *Journal of Network and Computer Applications*, vol. 216, p. 103656, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804523000795

[29] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[30] Y. Huang, "Deep q-networks," *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 135–160, 2020.

[31] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 11 853–11 864, 2020.

[32] Z. Liu, L. Huang, Z. Gao, M. Luo, S. Hosseinalipour, and H. Dai, "Ga-drl: Graph neural network-augmented deep reinforcement learning for dag task scheduling over dynamic vehicular clouds," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 4226–4242, 2024.

[33] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1517–1530, 2021.

[34] C. Liu, H. Wang, M. Zhao, J. Liu, X. Zhao, and P. Yuan, "Dependency-aware online task offloading based on deep reinforcement learning for iov," *Journal of Cloud Computing*, vol. 13, no. 1, p. 136, 2024.

[35] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[36] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[37] Z. Li, T. Li, H. Liu, and T.-T. Chan, "Personalized federated deep reinforcement learning for heterogeneous edge content caching networks," in *2024 22nd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2024, pp. 313–320.

[38] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Information Fusion*, vol. 85, pp. 1–22, 2022.

[39] H. Shi, B. Liu, E. Wang, W. Han, J. Wang, S. Cui, and L. Wu, "Cooperative multi-agent reinforcement learning framework for edge intelligence-empowered traffic light control," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 7373–7384, 2024.

[40] H. Moudoud, Z. Abou El Houda, and B. Brik, "Advancing security and trust in wsns: A federated multi-agent deep reinforcement learning approach," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1540–1549, 2024.

[41] A. Kara, N. Saldi, and S. Yüksel, "Q-learning for mdps with general spaces: Convergence and near optimality via quantization under weak continuity," *Journal of Machine Learning Research*, vol. 24, no. 199, pp. 1–34, 2023.

[42] M. K. Mondal, S. Banerjee, D. Das, U. Ghosh, M. S. Al-Numay, and U. Biswas, "Toward energy-efficient and cost-effective task offloading in mobile edge computing for intelligent surveillance systems," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4087–4094, 2024.

[43] X. Huang, Y. Zhang, Y. Qi, C. Huang, and M. S. Hossain, "Energy-efficient uav scheduling and probabilistic task offloading for digital twin-empowered consumer electronics industry," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2145–2154, 2024.

[44] J. Chi, X. Zhou, F. Xiao, Y. Lim, and T. Qiu, "Task offloading via prioritized experience-based double dueling dqn in edge-assisted iiot," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 575–14 591, 2024.

Manjinder Kaur received the M.Tech. degree in computer science and engineering from Guru Nanak Dev University, Jalandhar, India, in 2020, and is currently pursuing the Ph.D. degree at the School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield, U.K. Her research advances task partitioning in distributed computing through deep learning, with additional contributions in network optimization, TinyML models, and AI-driven solutions. Her broader interests encompass machine learning, data analysis, and intelligent systems, with a strong focus on translating innovative algorithms into real-world engineering and computing applications.

Hui Cheng received the B.Sc. and M.Sc. degrees in computer science from Northeastern University, Shenyang, China, in 2001 and 2004, and the Ph.D. degree in computer science from Hong Kong Polytechnic University, Hong Kong, in 2007. He was a Senior Lecturer with Liverpool John Moores University, Liverpool, U.K., from October 2013 to June 2018. He is currently a Senior Lecturer with the University of Hertfordshire, Hatfield, U.K. His research interests include edge computing, artificial intelligence, and mobile networks.

Pandelis Kourtessis joined the University of Hertfordshire in 2003, where he is Professor in Cognitive Networks and Associate Dean for Research and Enterprise. His research portfolio includes projects funded by FP7, H2020, ESA, UKRI, the ICS, and industry, and he has supervised 18 Ph.D. and Eng.D. completions. He has served as General Chair, Co-Chair, and Programme Committee Member for IEEE events, and on scientific committees and expert panels for European technology platforms and Networks of Excellence. Author of over 200 peer-reviewed publications, his work has been featured in journals, magazines, white papers, and international workshops, where he is often invited to present on cognitive networks and future communication systems. He has co-edited a Springer book, contributed to an IET 5G softwarization volume, and in REF2021 (UoA General Engineering) submitted five 3*–4* journal papers and led an impact case study on a next-generation video broadcasting system developed with BBC R&D and Global Invacom Group Ltd., resulting in two international patents, a joint press release, and an IEEE BCS SG invitation to contribute to a service amendment to the IEEE 802.11 standard.