Adaptive Deep Energy-Aware Edge Caching for Multimedia IoT in Edge–Fog Networks

Shilpa Bagade, Anjani Devi Thanneru, R. Sahith, V. Bharathi Devarakonda, and Anna Geethanjali

***Abstract*—Edge caching has emerged as a key enabler for latency-sensitive multimedia Internet of Things (IoT) applications by bringing content closer to end users. However, existing caching strategies often fail to adapt to dynamic network conditions and do not jointly optimize multiple performance objectives, particularly energy efficiency i n e dge–fog e nvironments. This paper proposes an adaptive deep reinforcement learning (DRL)-based energy-aware edge caching framework for multimedia IoT networks. The approach models caching as a sequential decision-making problem and dynamically learns optimal content placement policies using real-time network states and user demand patterns. A double deep Q-network (DDQN)-based framework is developed with a multi-objective optimization model that jointly improves cache hit ratio, reduces server access, and minimizes energy consumption. An energy-aware reward mechanism is designed to guide efficient c aching d ecisions, w hile t he edge–fog architecture enables scalable deployment. The model operates without prior knowledge of traffic d istributions, m aking i t suitable for heterogeneous IoT scenarios. Simulation results demonstrate that the proposed framework significantly outperforms baseline methods in terms of cache efficiency, e nergy utilization, and reduced server dependency, highlighting its effectiveness for intelligent edge–fog IoT networks.**

***Index Terms*—Edge caching, deep reinforcement learning, energy efficiency, m ultimedia I oT, e dge–fog networks.**

I. Introduction

The rapid proliferation of the Internet of Things (IoT) has led to an unprecedented growth in multimedia data traffic, d riven b y a pplications s uch a s s mart c ities, autonomous vehicles, healthcare monitoring, and industrial automation. These applications impose stringent requirements on latency, bandwidth utilization, and energy efficiency. Traditional cloud-centric architectures are often inadequate to meet these demands due to high communication delays, excessive backhaul usage, and increased dependence on remote servers. As a result, edge–fog computing paradigms have emerged as a promising solution by bringing computation and storage resources closer to end users.

Edge caching has become a key enabling technology in edge–fog environments, allowing frequently requested content to be stored at edge nodes, thereby reducing latency and alleviating network congestion. However, designing efficient caching strategies for multimedia IoT environments remains a significant challenge due to dynamic user demands, heterogeneous device capabilities, limited storage resources, and energy constraints. Comprehensive surveys highlight that conventional caching approaches fail to adapt to dynamic network conditions, resulting in suboptimal performance [1], [2].

Recent research has addressed specific challenges in edge caching from multiple perspectives. In terms of information freshness, Age of Information (AoI)-aware caching strategies have gained attention in vehicular IoT scenarios. For instance, the use of autonomous aerial vehicles (AAVs) for AoI-aware caching improves data freshness while maintaining energy efficiency [3]. Similarly, in industrial IoT environments, energy-efficient cache update mechanisms have been proposed to balance freshness and energy consumption [4].

Another important research direction focuses on intelligent and collaborative caching strategies. Federated and deep reinforcement learning-based approaches have been explored to dynamically optimize content placement in edge-cloud environments under varying network conditions [5]. Additionally, multimedia caching systems powered by renewable energy sources aim to balance energy consumption and service quality [6]. Resource optimization techniques further enhance system efficiency in distributed edge environments [7].

From a networking perspective, Information-Centric Networking (ICN) and Software-Defined Networking (SDN) have enabled efficient in-network caching and content delivery. In Internet of Medical Things (IoMT) environments, dynamic cache scheduling frameworks have been developed to improve both latency and reliability [8]. Similarly, clustering-based and popularity-driven caching approaches improve cache utilization and reduce redundancy in IoT networks [9]. Intelligent cache management techniques have also been proposed for delay-sensitive applications [10].

Despite these advancements, most existing approaches rely on heuristic or semi-adaptive mechanisms that lack the ability to jointly optimize multiple performance metrics in highly dynamic environments. In particular, the combined optimization of cache hit ratio, latency, energy consumption, and server access reduction remains a challenging problem.

Manuscript received December 23, 2025; revised January 16, 2026. Date of publication July 8, 2026. Date of current version July 8, 2026. The associate editor prof. Jelena Čulić Gambiroža has been coordinating the review of this manuscript and approved it for publication.

S. Bagade was with the Department of Information and Technology, Malla Reddy University, Hyderabad, India (e-mail: shilpa.me1437@gmail.com).

A. Devi Thanneru is with the Department of Information Technology, Vasavi Engineering College, India (e-mail: anjali.nagineni@gmail.com).

R. Sahith is with the Department of Computer Science and Engineering, CVR College of Engineering, India (e-mail: r.sahith@cvr.ac.in).

V. Bharathi Devarakonda is with the ECE Department, Aditya University, India (e-mail: vbharathid@adityauniversity.in).

A. Geethanjali is with the Vardhaman College of Engineering, India (e-mail: geethanjalimanga@gmail.com).

Digital Object Identifier (DOI): 10.24138/jcomss-2025-0284

To address these limitations, machine learning techniques, particularly deep reinforcement learning (DRL), have recently been introduced for intelligent caching decisions. DRL-based approaches enable dynamic adaptation by learning optimal caching policies through interaction with the environment. However, existing DRL-based solutions often overlook energy-aware optimization or fail to fully integrate edge–fog architectural dynamics.

Motivated by these challenges, this paper proposes an adaptive deep reinforcement learning-based energy-aware edge caching framework for multimedia IoT applications in intelligent edge–fog network environments. The proposed approach dynamically optimizes content placement by considering network dynamics, content popularity, and energy constraints, thereby improving caching efficiency and reducing reliance on remote servers. An energy-aware optimization model is formulated, and a DRL-based caching policy is developed to enable adaptive decision-making under varying conditions.

The main contributions of this paper are summarized as follows:

- An adaptive energy-aware edge caching framework is proposed for multimedia IoT services in edge–fog environments.
- An optimization model is formulated to improve cache hit ratio while achieving server access reduction and minimizing energy consumption.
- A deep reinforcement learning-based caching policy is designed to dynamically adapt content placement based on network state variations.
- Extensive simulation results demonstrate the effectiveness of the proposed approach in terms of improved performance compared to existing baseline methods.

The remainder of this paper is organized as follows. Section II presents the system model. Section III describes the proposed methodology. Section IV discusses performance evaluation and comparative analysis. Finally, Section V concludes the paper and outlines future research directions.

II. Related Work and System Model

A. Related Work

Energy-efficient caching and resource optimization in IoT-enabled edge networks have attracted significant research attention due to the rapid growth of multimedia applications and latency-sensitive services. Early studies in information-centric IoT (ICN-IoT) demonstrated that collaborative caching strategies can significantly reduce redundant transmissions and improve energy efficiency [11]. Similarly, performance analyses of caching mechanisms in ICN-based IoT highlight the importance of network-layer caching in reducing access delay and improving data delivery efficiency [12].

Recent advancements focus on joint optimization of communication and caching. For instance, the integration of multicast transmission with in-network caching improves energy efficiency in green IoT systems [13]. In vehicular edge environments, mobility-aware mechanisms such as energy-efficient reservation systems enhance throughput and packet delivery under dynamic conditions [14]. Furthermore, online caching approaches such as Online Multi-Agent Edge Caching OL-MEDC adapt to time-varying content requests, optimizing storage utilization and retrieval latency [15].

Energy-aware scheduling has also been widely explored in IoT healthcare systems, where constrained devices require efficient workload management to prolong operational lifetime [16]. Comprehensive surveys emphasize that prediction-based caching, collaborative strategies, and joint caching–computation optimization remain key research directions in mobile edge computing [17]. In addition, multi-hop multimedia routing and edge-assisted caching techniques have demonstrated substantial reductions in energy consumption and transmission overhead [18].

Recent research further highlights the role of Software-Defined Networking (SDN) in enabling flexible cache orchestration, although challenges such as controller scalability and cache coherency persist [19]. Foundational work on heterogeneous edge networks confirms that optimal cache placement significantly reduces energy consumption per request [20]. Smart caching mechanisms in ICN combined with edge computing also reduce multimedia streaming delays through cooperative and semantic-aware caching [21].

To improve cache efficiency, collaborative filtering techniques have been applied to predict content popularity and enhance cache hit ratios [22]. Efficient data sharing and caching mechanisms further reduce retrieval latency in ICN-IoT environments [23]. Centralized caching control strategies have also been proposed to globally optimize energy efficiency while respecting local constraints [24]. Additionally, in-memory caching techniques across multi-tier architectures significantly reduce processing delay and energy consumption [27].

With the emergence of artificial intelligence, deep reinforcement learning (DRL) has become a promising approach for adaptive caching. DRL-based frameworks dynamically learn caching policies based on network conditions and user demand patterns [25]. Advanced approaches incorporating attention mechanisms, mobility prediction, and multi-agent collaboration further enhance caching performance in dynamic IoT environments [32], [33]. Meta-reinforcement learning and distributed service caching techniques also improve scalability and adaptability in large-scale systems [30], [31], [34]. Despite these advancements, challenges remain in achieving a balanced trade-off between energy efficiency, latency, and cache utilization.

B. System Model

We consider a hierarchical edge computing architecture consisting of *edge nodes*, *fog nodes*, and a centralized *cloud server*, designed to support multimedia IoT applications. Edge nodes are located close to end-users and are responsible for handling content requests with minimal latency. Fog nodes provide intermediate processing and caching capabilities, while the cloud server offers large-scale storage and computational resources.The proposed architecture shown in Fig. 1 presents a hierarchical framework integrating IoT, edge, fog, and cloud layers. The IoT layer generates data and initiates

content requests, while the edge layer performs local caching and processing. The fog layer coordinates resource allocation and optimizes caching decisions using deep reinforcement learning. Finally, the cloud layer handles global model training and long-term storage. This multi-layer design enables efficient data management, reduced latency, and improved overall network performance.

Each edge node is equipped with limited cache storage and serves user requests based on locally stored content or by retrieving data from neighboring nodes or the cloud. The system operates under dynamic conditions characterized by varying content popularity, user mobility, and network traffic patterns.

To address these challenges, we propose an Adaptive Deep Energy-Aware Edge Caching (ADEE-EC) framework that integrates deep reinforcement learning with energy-aware decision-making. The objective is to jointly optimize energy consumption, latency, and cache hit ratio.

When a content request arrives, the edge node first checks its local cache. If the requested content is available, it is served immediately, minimizing delay and energy usage. Otherwise, the ADEE-EC decision module determines whether to fetch the content from neighboring nodes or the cloud server. This decision is based on learned policies that consider energy cost, retrieval delay, and content popularity.

After content delivery, the caching system updates its storage by either retaining or replacing content based on its predicted future demand. The DRL agent continuously updates its policy using observed system feedback, enabling adaptive and intelligent caching decisions over time.

The overall optimization objective can be expressed as a joint minimization of energy consumption and latency while maximizing cache efficiency:

$$\min \ \mathcal{J} = \alpha E + \beta L - \gamma H \tag{1}$$

where $\mathcal{J}$ denotes the overall objective function. E represents the total energy consumption (in Joules), L denotes the end-to-end latency (in milliseconds), and H is the cache hit ratio (dimensionless). The coefficients α, β, and γ are non-negative weighting factors that control the relative importance of energy, latency, and cache performance, respectively.

This formulation enables the proposed framework to achieve a balanced trade-off among conflicting performance metrics, making it suitable for large-scale, dynamic IoT environments.

III. Proposed ADEE-EC Framework

This section presents the adaptive deep reinforcement learning–based energy-aware edge caching framework. The proposed caching policy is learned using the Double Deep Q-Network (DDQN) algorithm , which mitigates overestimation bias in Q-learning and improves training stability [35].The proposed ADEE-EC framework operates across a hierarchical IoT architecture consisting of IoT devices, an edge caching layer, fog/coordination nodes, and a cloud layer. Each layer contributes distinct intelligence, coordination, and processing capabilities that collectively support energy-efficient and adaptive caching for multimedia IoT data. The methodology integrates deep reinforcement learning (DRL), mobility adaptation, and popularity prediction to enable autonomous and energy-aware caching decisions in ICN-based IoT networks.During deployment, the proposed model performs only inference at the edge, which involves low-complexity forward propagation, enabling near-real-time caching decisions. At the IoT device layer, multimedia sensors, cameras, and mobile users continuously generate data requests and content streams. These devices act as the primary consumers of cached data and provide real-time contextual information including request frequency, mobility patterns, residual device energy, and link quality. This information forms the state input for the ADEE-EC learning engine. Since IoT devices are energy-constrained, ADEE-EC prioritizes offloading computation and caching decisions to the edge to minimize device-side processing cost. At the edge caching layer, ADEE-EC performs its core decision-making. This layer hosts the DRL-based caching agent responsible for selecting optimal caching actions such as storing, replacing, forwarding, or discarding content. The DRL agent models caching as a Markov Decision Process (MDP), where the system state includes predicted content popularity, user mobility likelihood, and node energy profiles. A reward function is designed to jointly optimize cache hit ratio, energy efficiency, and reduced dependency on upstream cloud servers. The edge also executes the content popularity prediction model, which uses historical request patterns to estimate future demand, generating a content utility score. This utility score is combined with mobility-aware access predictions to prioritize which multimedia objects should be placed in edge caches to maximize future hit probability. By performing deep learning-driven caching locally, the edge layer becomes the central point for Energy-Efficient, Communication-Efficient caching, as reflected in the diagram. The fog/coordination layer aggregates information from multiple edge nodes and provides regional decision optimization. This layer enables cooperative caching through coordination between edges, minimizing redundancy and ensuring that distributed caches collectively cover a broader set of high-value multimedia content. Fog nodes also handle load balancing by dynamically reallocating caching responsibilities based on traffic density, user movement, or energy depletion at specific edge nodes. They further assist the DRL agent by providing global state summaries—such as cluster-level popularity and inter-edge movement patterns—that improve training stability and convergence. The cloud layer serves as a long-term repository for full multimedia content and analytics. While ADEE-EC seeks to reduce cloud dependency, the cloud remains responsible for storing content that cannot be cached due to capacity limits. It also performs offline training for the deep learning models, periodically updating the DRL agent parameters at the edge and fog layers. Cloud-to-fog updates ensure that the DRL model evolves with long-term trends while still benefiting from real-time, local adaptations performed at the edge. The objective function is defined as Cache capacity constraint is given by

$$\sum_{f} s(fx_i, f(t) \leq C_i \, \forall_{i,t}) \tag{2}$$

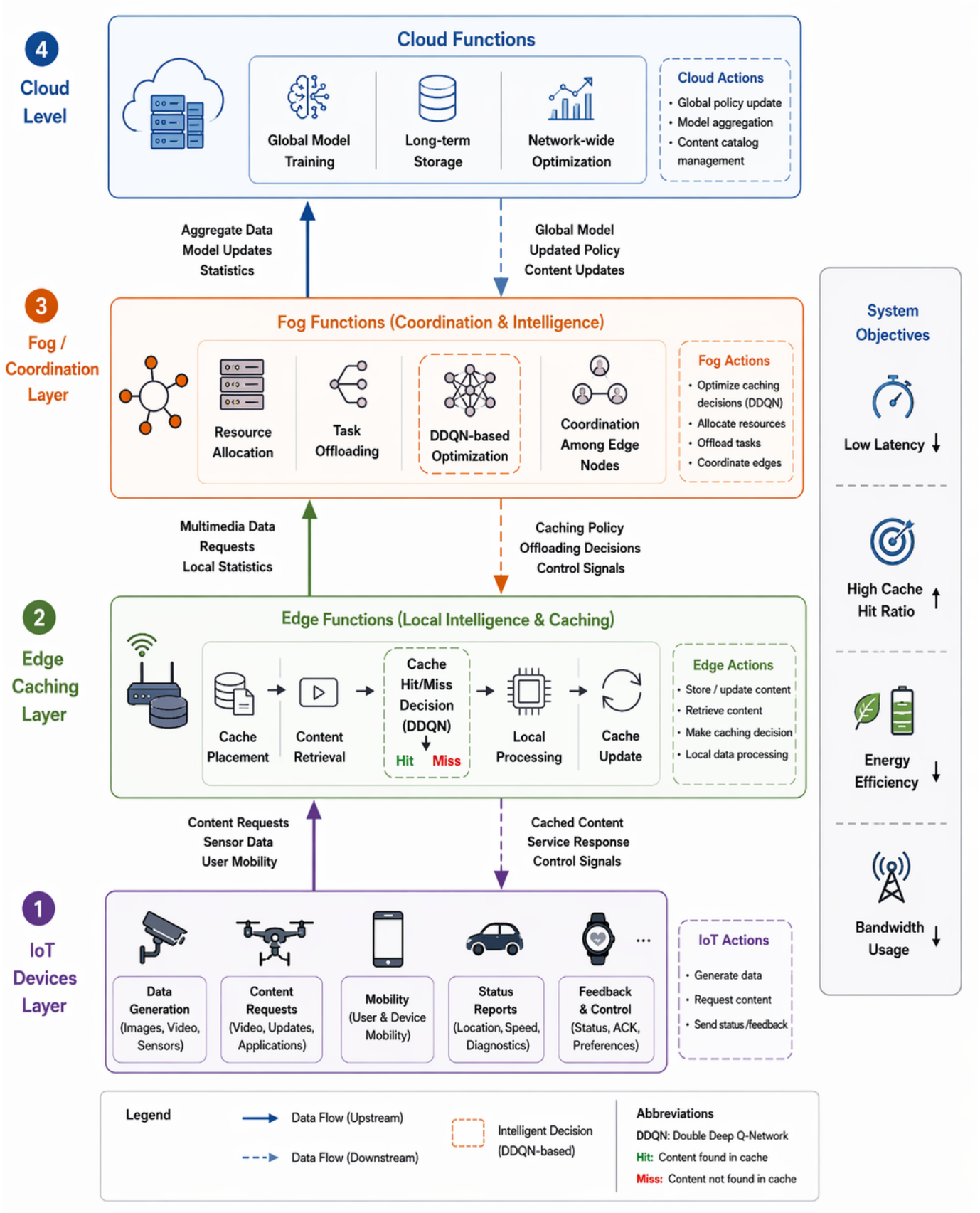

Fig. 1. Architecture

The cache hit probability at node i for content f, given that a request for f arrives at node i at time t, is defined as follows:

$$\Pr[\text{hit}_{i,f}(t)] = x_{i,f}(t) + \big(1 - x_{i,f}(t)\big) \sum_{j \in \mathcal{N}_i} w_{i,j}(t)\, x_{j,f}(t) \quad (3)$$

where N_i is neighborhood (nearby edge/fog nodes) and $w_{i,j}(t)$is probability of fetching from j (network routing/ICN forwarding weights). Algorithm 1 explains the step-by-step procedure used to compute the energy consumption within the Adaptive Deep Energy-Aware Edge Caching (ADEE-EC) framework. It outlines how different energy components—such as sensing, data transmission, reception, processing, and collaborative operations—are quantified for each IoT node in the edge–fog environment. The algorithm systematically integrates predefined energy coefficients α (energy), β (latency), γ and operational parameters to estimate the total energy cost associated with caching decisions. This computation enables ADEE-EC to intelligently evaluate the energy impact of storing or retrieving content, ensuring that caching actions are optimized for both efficiency and sustainability across the network. Expected latency per request for content f at node i

Algorithm 1 Energy Model Computation for ADEE-EC

Input: Current cache state, request arrival rate, transmission mode, device parameters, and system weights α (energy), β (latency), γ (cache-hit).
Initialize: Energy counters for caching, transmission, and processing.
Determine the type of operation:
- Cache hit (local serve)
- Fetch from neighboring edge node
- Fetch from cloud/server

if operation is a local cache hit **then**
    Set energy consumption to a low value (only processing and minimal device activation).
    Mark latency as minimal.
else if operation is fetch from neighbor **then**
    Set energy consumption to medium level (wireless communication + processing).
    Set latency to moderate.
else
    Operation is cloud/server fetch.
    Set energy consumption to high level (long-range communication + processing).
    Mark latency as high.
end if
Adjust final energy score using system weight α to emphasize energy sensitivity.
Adjust latency score using weight β to reflect delay tolerance.
Adjust cache-hit impact using weight γ.
Update node-level total energy consumption for this request.
Return updated energy values to the DRL agent.

$$L_{i,f}(t) = l_i^{\text{local}}\, x_{i,f}(t) + \sum_{j \in N_i} l_{i,j}(t)\, w_{i,j}(t)\, (1 - x_{i,f}(t)) + l^{\text{cloud}}\, (1 - x_{i,f}(t)) \prod_{j \in N_i} (1 - x_{j,f}(t)) \quad (4)$$

Combines local fetch, neighbor fetch, and cloud fetch latencies ℓ. Total energy consumed at node i during interval t (approx):

$$E_i(t) = E_i^{\text{store}}(t) + E_i^{\text{tx}}(t) + E_i^{\text{rx}}(t) + E_i^{\text{proc}}(t) \quad (5)$$

$$E_i^{\text{store}}(t) = K_i^{\text{store}} \sum_f s_f x_{i,f}(t) \quad (6)$$

$$E_i^{\text{tx}}(t) = \sum_f K_i^{\text{tx}} \lambda_{i,f}(t)(1 - \Pr[\text{hit}_{i,f}(t)])s_f, \quad (7)$$

$$E_i^{\text{rx}}(t) = \sum_f K_i^{\text{rx}} \lambda_{i,f}(t)(1 - \Pr[\text{hit}_{i,f}(t)])s_f, \quad (8)$$

$$E_i^{\text{proc}}(t) = K_i^{\text{proc}} \text{proc\_load}_i(t). \quad (9)$$

Mobility-aware availability for consumer u obtaining file f from node i:

$$\Pr[\text{available}_{u,i,f}(t)] \approx \pi_{u,i}(t) x_{i,f}(t), \quad (10)$$

Objective function:

$$\min_{\{x_{i,f}(t)\}} J(t) = \alpha \sum_i E_i(t) + \beta \sum_{i,f} \lambda_{i,f}(t) L_{i,f}(t) - \gamma \sum_{i,f} \lambda_{i,f}(t) \Pr[\text{hit}_{i,f}(t)]. \quad (11)$$

Overall, the mathematical formulations and energy computation models provide the foundation for enabling intelligent cache placement and eviction decisions within the ADEE-EC framework. By integrating DRL-driven optimization, mobility-aware updates, and popularity-based predictions, the framework supports a fully autonomous caching mechanism that minimizes energy consumption while maximizing cache hit efficiency. These equations collectively ensure that every caching action—whether performed at the IoT, edge, fog, or cloud layer—remains adaptive, resource-aware, and aligned with real-time network dynamics. This comprehensive modeling establishes the groundwork for the performance evaluation presented in the subsequent sections.

IV. Deep Reinforcement Learning–Based Edge Caching Framework

A. System Model and Problem Formulation

Content popularity is modeled using the Zipf distribution [37], which reflects realistic multimedia access patterns where a small number of popular contents dominate user requests.

The objective of the proposed framework is to jointly optimize cache efficiency, server access, and energy consumption in an edge–fog-enabled IoT environment. Specifically, the problem is formulated as a multi-objective optimization task.

$$\max_{\pi} \; J = \alpha H_{cache} - \beta S_{access} - \gamma E_{total} \quad (12)$$

where H_{cache} denotes the cache hit ratio, S_{access} represents the number of server (cloud) accesses, and E_{total} is the total energy consumption of the system. The coefficients α, β, and

γ are weighting parameters that control the trade-off among performance metrics.

The objective is to maximize cache hit ratio while minimizing server access and overall energy consumption. This formulation enables efficient content placement and retrieval decisions under dynamic network conditions.

The total energy consumption is modeled as the sum of transmission, caching, and computation energy, i.e., $E_{total} = E_{tx} + E_{cache} + E_{comp}$. Energy efficiency is defined as the ratio of successfully delivered data to total energy consumption. This formulation is used to evaluate the performance of the proposed framework.

1) Network Architecture: We consider a hierarchical edge–fog–cloud network architecture designed to support delay-sensitive multimedia IoT applications. The architecture consists of a set of edge nodes deployed in close proximity to end users, an intermediate fog layer with moderate computation and storage capabilities, and a centralized cloud server that maintains the complete multimedia content library. Each edge node serves user requests within its coverage area and is equipped with limited caching and computational resources. When a requested content item is not available locally, the request is forwarded to the fog or cloud layer, resulting in increased service latency and energy consumption.

2) Caching Model: Let $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ denote the multimedia content library. User requests arrive at edge nodes according to a stochastic process and exhibit time-varying popularity characteristics. Each edge node maintains a cache with finite capacity C_i. Upon receiving a content request, the edge node must decide whether to cache the requested content, retain the existing cached contents, or evict low-utility contents when the cache is full. The caching decision directly impacts cache hit ratio, service latency, and energy consumption.

3) State, Action, and Reward Definition: The dynamic edge caching problem is formulated as a Markov Decision Process (MDP). To ensure scalability and practical deployment at resource-constrained edge nodes, a compact state representation is adopted. At each decision epoch t, the DRL agent observes the state:

$$s_t = \left[\rho_t,\ \hat{p}_t,\ q_t,\ \hat{E}_t\right], \tag{13}$$

where ρ_t denotes the cache utilization ratio, $\hat{p}_t$ represents the estimated popularity of the currently requested content, q_t is the request queue length at the edge node, and $\hat{E}_t$ denotes the normalized energy consumption state. All state variables are normalized to the range $[0, 1]$.

The action space is defined as:

$$\mathcal{A} = \{a_0, a_1, \ldots, a_{C_i}\}, \tag{14}$$

where a_0 denotes forwarding the request to the fog or cloud without caching, and a_k $(k = 1, \ldots, C_i)$ corresponds to evicting the k-th cached content to store the newly requested content.

The reward function jointly optimizes cache efficiency, service latency, and energy consumption, and is defined as:

$$r_t = h_t - \beta d_t - \gamma e_t, \tag{15}$$

where h_t is the cache hit indicator, d_t denotes the normalized end-to-end service delay, e_t represents the normalized energy consumption, and β and γ are weighting coefficients.

TABLE I
NEURAL NETWORK ARCHITECTURE OF THE DRL AGENT

Layer	*Type*	*Neurons*	*Activation*
Input Layer	Fully Connected	N_s	–
Hidden Layer 1	Fully Connected	256	ReLU
Hidden Layer 2	Fully Connected	128	ReLU
Hidden Layer 3	Fully Connected	64	ReLU
Output Layer	Fully Connected	N_a	Linear

B. DRL Model Architecture

To learn an efficient caching policy, the proposed framework employs a Double Deep Q-Network (DDQN) to mitigate the overestimation bias and instability of conventional Q-learning. The DRL agent consists of an online network for action selection and a target network for stable Q-value estimation. Both networks share an identical architecture, as summarized in Table I.

The neural network takes the compact state vector as input and consists of three fully connected hidden layers with 256, 128, and 64 neurons, respectively. Rectified Linear Unit (ReLU) activation functions are employed in the hidden layers to improve convergence and computational efficiency. The output layer produces Q-values corresponding to all feasible caching actions. The lightweight network design ensures low inference latency and minimal memory overhead, making it suitable for deployment on edge devices.

C. Training Strategy and Hyperparameter Settings

The DRL agent is trained using an experience replay mechanism to reduce correlation among training samples and improve learning stability. An ϵ-greedy exploration strategy is adopted, where the exploration rate is initially set to a high value and gradually decayed to encourage exploitation of the learned policy. The Adam optimizer is used to update network parameters, and gradient clipping is applied to prevent unstable parameter updates.

Training is conducted over 4,000 episodes, each consisting of 200 decision steps. The complete hyperparameter configuration used in the experiments is summarized in Table II, ensuring full reproducibility of the learning process.

D. Simulation Environment and Implementation Details

The proposed framework is evaluated using a custom simulation environment implemented in Python with the PyTorch deep learning library. Training is performed offline using historical request traces, while the trained DRL model is deployed online at edge nodes for real-time inference. The simulation parameters, including network size, cache capacity, content library size, and request distribution, are summarized in Table III. The lightweight architecture ensures that online inference incurs negligible computational overhead, making the proposed approach suitable for practical edge deployment.

TABLE II
HYPERPARAMETER CONFIGURATION

Hyperparameter	*Value*
Learning Rate (α)	0.0005
Discount Factor (γ)	0.95
Batch Size	64
Replay Buffer Size	1×10^5
Target Network Update Interval	1,000 steps
Exploration Strategy	ϵ-greedy
Initial Exploration Rate (ϵ_0)	1.0
Final Exploration Rate ($\epsilon_{\min}$)	0.05
Optimizer	Adam
Gradient Clipping	1.0
Training Episodes	4,000
Steps per Episode	200

TABLE III
SIMULATION PARAMETERS AND EXPERIMENTAL SETUP

Parameter	*Value*
Number of edge nodes	5–20
Number of fog nodes	1–3
Cloud server	1 (centralized)
Cache capacity per edge node	10–50 contents
Content library size	1,000 items
Content size	Uniform (1–10 MB)
Request arrival process	Poisson
Content popularity distribution	Zipf ($\alpha = 0.8$)
Wireless data rate	6 Mbps
Backhaul link delay	20–50 ms
Edge processing delay	2–5 ms
Simulation time per episode	200 time steps
Number of training episodes	4,000
DRL framework	PyTorch
Training mode	Offline
Inference mode	Online at edge nodes

V. PERFORMANCE EVALUATION AND COMPARISON

A. Simulation Setup

Simulation parameters are selected to reflect realistic multimedia IoT scenarios. The simulation environment is implemented using the NS-3 network simulator [36], which provides realistic modeling of network protocols and communication scenarios.The performance of the proposed Adaptive Deep Energy-Aware Edge Caching (ADEE-EC) framework is evaluated in a simulated IoT–edge computing environment consisting of IoT devices, edge nodes, and a cloud server deployed over a 1000 × 1000 m area. IoT devices generate multimedia content requests following a Poisson distribution, while content popularity follows a Zipf model. Edge nodes are equipped with limited cache storage and are connected to the cloud via high-bandwidth links.

Wireless communication channels incorporate path loss and log-normal shadowing, and node mobility is modeled using the random waypoint model. The energy consumption model accounts for local processing, transmission, and idle states, capturing the dominant energy costs in edge caching systems. Performance is evaluated using cache hit ratio, energy efficiency, latency, and server hit rate.

The simulation environment is implemented in Python using PyTorch, enabling flexible evaluation of caching policies under diverse traffic conditions. Consistent with prior DRL-based caching studies, the focus is on algorithmic validation rather than packet-level network simulation, which is left for future work.

The ADEE-EC framework employs a Double Dueling Deep Q-Network (D3QN) to balance learning stability and runtime efficiency. The neural network consists of two fully connected hidden layers with ReLU activation functions, followed by separate value and advantage streams, resulting in approximate 1.8×10^5 trainable parameters. This lightweight architecture facilitates deployment at edge nodes.

Training is performed offline using historical request traces to avoid computational overhead during online operation. A mini-batch training strategy is adopted with a batch size of 64 and a replay buffer of 100,000 transitions. The Adam optimizer is used with a learning rate of 1×10^{-4}, and a target network is periodically updated to ensure stable convergence. Retraining is triggered only when significant changes in content demand patterns are observed.

All training experiments are conducted on a workstation equipped with an NVIDIA GPU, while inference is executed on CPU-based edge hardware. During online operation, ADEE-EC requires only a single forward pass to make caching decisions, resulting in sub-millisecond inference latency, which is negligible compared to network transmission and content delivery delays in multimedia IoT environments.

B. Performance Metrics

Cache Hit Ratio (CHR): Cache Hit Ratio (CHR) is defined as the proportion of user content requests that are successfully served directly from the edge cache without accessing the remote server. It is an important metric for evaluating the efficiency of caching mechanisms. Mathematically, it is expressed as:

$$CHR = \frac{N_{\text{cache}}}{N_{\text{total}}} \tag{16}$$

where N_{cache} denotes the number of requests served from the cache, and N_{total} represents the total number of user requests. A higher CHR indicates better caching performance and reduced latency.

Server Hits: Server Hits represent the number of requests that are not satisfied by the edge cache and must be served by the central server. This metric reflects the load on the core network and is given by:

$$N_{\text{server}} = N_{\text{total}} - N_{\text{cache}} \tag{17}$$

Minimizing server hits helps in reducing backhaul traffic and improving overall system efficiency.

For all figures, the term "Strategy Index" represents different simulation instances corresponding to varying network conditions, request patterns, and caching decisions evaluated during the experiments. Consistent color coding is used across all figures, where each color corresponds to a specific caching method for clear comparison. All percentage-based results are computed relative to the baseline method (LCE), which represents a conventional caching strategy. The improvements

indicate the relative gain of the proposed and comparative methods over this baseline.

C. Cache Hit Ratio Analysis

As shown in Figure 2, when edge nodes are equipped with a 10 GB cache, only a limited portion of popular content can be stored. The ADEE-EC model improves cache hits by prioritizing high-demand content, significantly reducing server dependence and access latency.

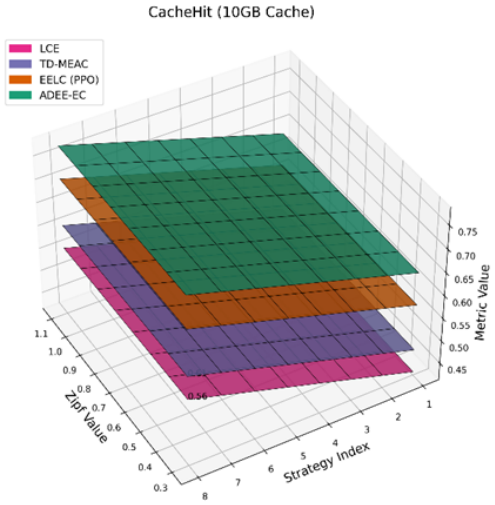

Fig. 2. Cache hit ratio (%) of different caching strategies at 10 GB cache capacity, averaged over five runs (higher is better).

With a larger 32 GB cache capacity (Figure 3), edge nodes can store a wider set of content. This leads to substantially higher cache hit ratios, improved content availability, and reduced cloud interactions. ADEE-EC further boosts performance by learning optimal caching patterns from predicted request behavior.

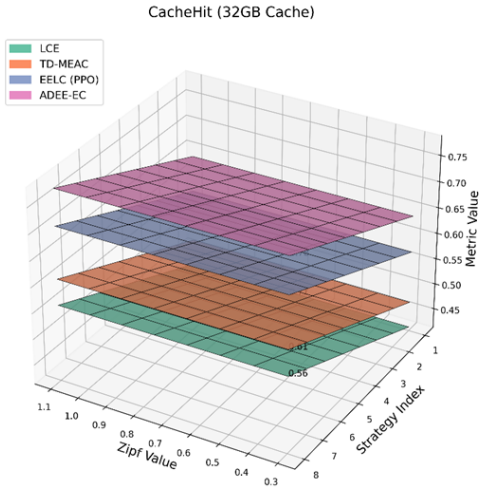

Fig. 3. Cache hit ratio (%) of different caching strategies at 32 GB cache capacity, averaged over five runs (higher is better).

D. Energy Efficiency Comparison

Figure 4 illustrates the energy efficiency of the caching schemes at a 10 GB cache size. ADEE-EC achieves 34.8% higher energy efficiency than ELC and significantly outperforms EELC due to its adaptive DRL-driven decision-making. EELC provides moderate improvement but is constrained by static strategies.

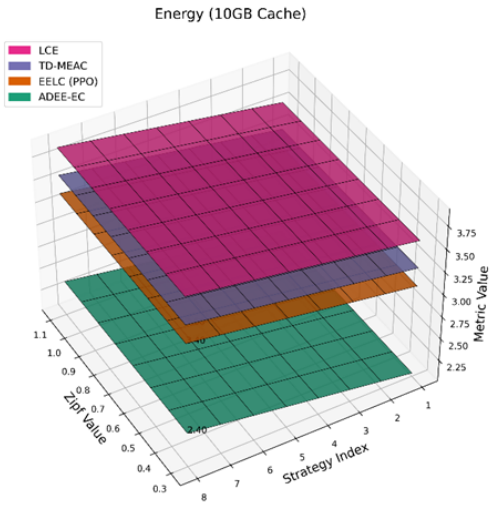

Fig. 4. Energy efficiency (%) of different caching strategies at 10 GB cache capacity, averaged over five runs (higher is better).

At 32 GB (Figure 5), ADEE-EC maintains superior performance by fully utilizing the larger cache. The DRL model benefits from expanded storage, enabling more energy-optimal placement strategies.

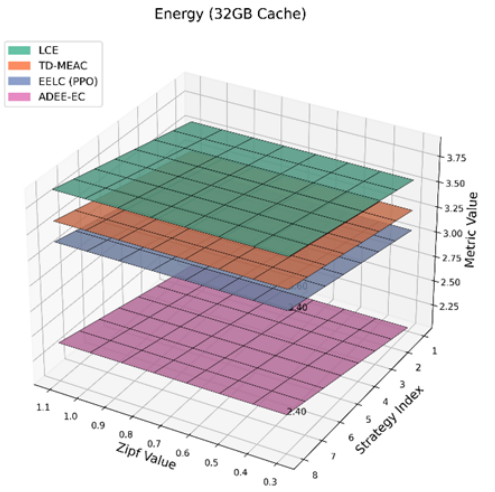

Fig. 5. Energy efficiency (%) of different caching strategies at 32 GB cache capacity, averaged over five runs (higher is better).

E. Server Hit Analysis

Figure 6 shows the server hits at a 10 GB cache size. ADEE-EC reduces server hits by 26.5% compared to ELC, attributed to its predictive caching and adaptive learning capabilities.

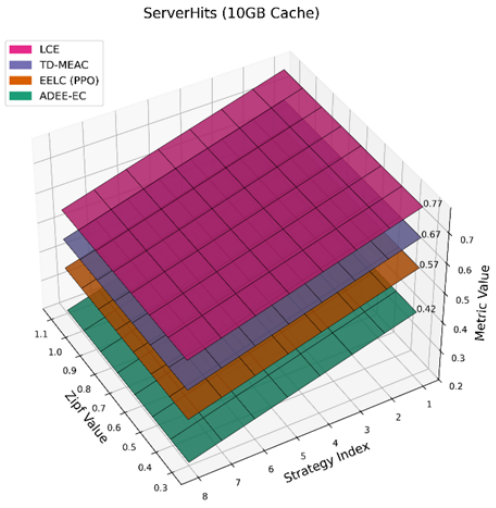

Fig. 6. Server hit reduction (%) of different caching strategies at 10 GB cache capacity, averaged over five runs (higher is better).

At 32 GB (Figure 7), all methods show fewer server hits due to increased local availability, but ADEE-EC remains the best performer by leveraging intelligent, demand-aware caching.

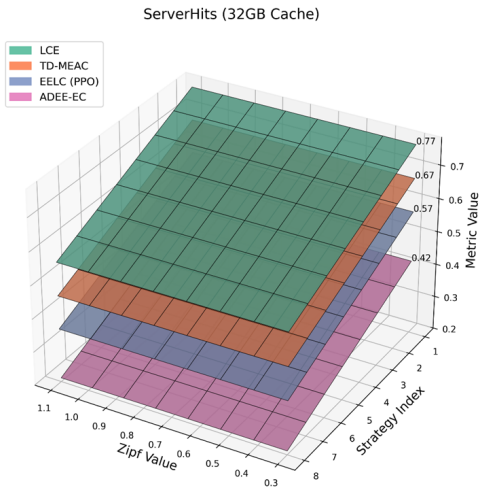

Fig. 7. Server hit reduction (%) of different caching strategies at 32 GB cache capacity, averaged over five runs (higher is better).

F. Comparison With Existing Methods

The proposed ADEE-EC approach is compared with several representative caching strategies from the literature, including both traditional and learning-based approaches.

LCE (Leave Copy Everywhere) is a classical caching strategy widely used in content-centric networking due to its simplicity, where content is replicated along the delivery path.

TD-MEAC is a reinforcement learning-based caching method that dynamically updates caching decisions using temporal-difference learning. Recent works on deep reinforcement learning-based caching demonstrate improved adaptability in dynamic environments [29], [30], [33].

TABLE IV
PERFORMANCE COMPARISON OF CACHING STRATEGIES. ENERGY EFFICIENCY, CACHE HIT RATIO, SERVER HIT REDUCTION, AND LATENCY REDUCTION ARE REPORTED AS PERCENTAGES (%). HIGHER VALUES INDICATE BETTER PERFORMANCE. GAIN DENOTES THE ABSOLUTE IMPROVEMENT OF ADEE-EC OVER THE BEST-PERFORMING BASELINE.

Metric	*Cache*	*LCE*	*TD-MEAC*	*EELC*	*ADEE-EC*	*Gain (%)*
Energy Efficiency (%)	10 GB	6.8	12.6	23.1	34.8	11.7
	32 GB	8.3	14.7	23.9	36.4	12.5
Cache Hit Ratio (%)	10 GB	7.4	8.9	12.5	19.6	7.1
	32 GB	7.8	9.8	10.3	18.9	8.6
Server Hit Reduction (%)	10 GB	9.3	11.2	17.9	26.5	8.6
	32 GB	10.1	13.5	21.3	30.4	9.1
Latency Reduction (%)	10 GB	11.6	14.2	25.3	37.1	11.8

EELC (Energy-Efficient Learning-based Caching) focuses on optimizing energy consumption while maintaining acceptable caching performance. Recent advancements in energy-aware and attention-based caching mechanisms further enhance efficiency [32], [34].

In contrast, the proposed *ADEE-EC* framework integrates deep reinforcement learning with adaptive energy-aware decision-making. This enables efficient caching in dynamic multimedia IoT environments.

Table IV presents a comprehensive comparison of these methods under different cache capacities. The results show that ADEE-EC consistently outperforms baseline approaches across all performance metrics, including energy efficiency, cache hit ratio, server hit reduction, and latency reduction.

G. Energy and Performance Modeling

The ADEE-EC algorithm utilizes deep reinforcement learning to adaptively manage edge caching decisions by balancing energy efficiency and content delivery performance. The step-by-step procedure of the proposed approach is presented in Algorithm 2.

Algorithm 2 Adaptive Deep Energy-Aware Edge Caching

Input: Requests R, devices E, cache size C, learning rate α, discount factor γ, parameters θ
Initialize cache state, energy budget, and replay buffer
for each request $r \in R$ **do**
    Observe state s_t
    Choose action a_t using policy derived from θ
    if content is cached **then**
        Serve locally
    else
        Fetch from cloud/neighbor
    end if
    Compute reward and store transition
    Update θ using sampled mini-batch
    Adapt α, γ, and exploration parameters
end for
return Updated cache policy

VI. Discussion

A. Practical Feasibility of the Proposed Framework

Although the evaluation of the proposed ADEE-EC framework is conducted in a simulation environment, the design of the system is aligned with practical deployment architectures used in modern edge–cloud collaborative computing systems. In the proposed framework, the computationally intensive training phase of the reinforcement learning model is performed offline using centralized computing resources, while only the lightweight inference stage is executed at the edge nodes. This separation significantly reduces the computational burden on resource-constrained IoT devices and allows the framework to operate efficiently in real-time environments. Furthermore, the inference process mainly involves forward propagation through the trained neural network, which requires relatively low computational overhead and can be executed within milliseconds on modern embedded edge processors. The simulation environment also incorporates dynamic variations in node density, traffic patterns, and communication behavior to emulate realistic multimedia IoT deployment scenarios. Therefore, the proposed ADEE-EC framework is designed to be compatible with practical edge computing infrastructures and can be deployed in real-world multimedia IoT systems with minimal modification. Although the evaluation of the proposed ADEE-EC framework is conducted in a simulation environment, the design of the system is aligned with practical deployment architectures used in modern edge–cloud collaborative computing systems. In the proposed framework, the computationally intensive training phase of the reinforcement learning model is performed offline using centralized computing resources, while only the lightweight inference stage is executed at the edge nodes. This separation significantly reduces the computational burden on resource-constrained IoT devices and allows the framework to operate efficiently in real-time environments.

Furthermore, the inference process mainly involves forward propagation through the trained neural network, which requires relatively low computational overhead and can be executed within milliseconds on modern embedded edge processors. The simulation environment also incorporates dynamic variations in node density, traffic patterns, and communication behavior to emulate realistic multimedia IoT deployment scenarios. Therefore, the proposed ADEE-EC framework is designed to be compatible with practical edge computing infrastructures and can be deployed in real-world multimedia IoT systems with minimal modification.

The inference latency of the proposed ADEE-EC framework is minimal, as caching decisions are performed using a trained deep reinforcement learning model. The inference process involves only forward propagation through a lightweight neural network, consisting primarily of matrix multiplications and activation functions, which can be efficiently executed on edge devices. In practical edge computing environments, this results in millisecond-level decision latency, making the proposed approach suitable for near-real-time multimedia IoT applications. Furthermore, since training is performed offline, only low-complexity inference is required during deployment.

B. Deployment Considerations

Although the proposed ADEE-EC framework is evaluated using a Python/PyTorch-based simulator, this setup is consistent with standard practice in deep reinforcement learning research and allows controlled evaluation of algorithmic performance. In practical deployments, the trained DRL model can be maintained at the cloud or fog layer and periodically updated at edge nodes.

The model update frequency from cloud to edge can be configured based on traffic dynamics and environmental variability. Since inference at the edge relies on a lightweight neural network, model updates are expected to occur infrequently, thereby limiting coordination overhead. Furthermore, edge nodes can continue operating with previously deployed models during update intervals, ensuring uninterrupted service.

C. Coordination Overhead and Model Staleness

Coordination overhead primarily arises during model dissemination rather than during online inference. As the proposed framework performs inference locally at edge nodes, per-request communication with the cloud is not required, significantly reducing signaling overhead. Potential model staleness may occur if content popularity changes rapidly; however, this effect can be mitigated through periodic retraining using recent request traces and adaptive update intervals.

D. Limitations and Future Work

The current study focuses on algorithmic validation and performance analysis under controlled simulation settings. Network-level simulation using tools such as ns-3, which would enable fine-grained modeling of protocol behavior and link dynamics, is not included in this work. We identify this as a limitation of the present study and plan to incorporate ns-3–based network-level evaluation and real-world testbed deployment as part of future work.

VII. Conclusion

This paper presented an adaptive deep energy-aware edge caching (ADEE-EC) framework for multimedia IoT applications in edge–fog environments. The proposed approach integrates deep reinforcement learning with energy-aware optimization, mobility modeling, and content popularity prediction to enable intelligent caching decisions under dynamic network conditions. Simulation results demonstrate that the ADEE-EC framework provides substantial performance improvements over existing methods. With a 10 GB cache, the proposed approach improves energy efficiency by 34.8%, increases cache hit ratio by 19.6%, and reduces server hits by 26.5% compared to ELC, while achieving an additional 11% gain over EELC. Under a 32 GB cache configuration, energy efficiency further increases to 36.4%, cache hit ratio reaches 18.9%, and server hits decrease by 30.4%. These improvements are primarily due to reduced redundant transmissions, adaptive content placement, and efficient utilization of edge and fog resources.

From an energy efficiency perspective, the proposed framework minimizes transmission and computation overhead by

dynamically optimizing caching decisions based on network state variations. This results in improved energy utilization, making the approach suitable for large-scale and energy-constrained IoT environments.

The findings highlight the practical relevance of the proposed framework for latency-sensitive and resource-constrained applications in ICN-enabled IoT systems. Future work will focus on extending the model to real-world deployments, incorporating heterogeneous network scenarios, and exploring advanced learning techniques to further enhance scalability and robustness.

REFERENCES

[1] B. Shafaie, M. K. Rafsanjani, V. Arya, and B. B. Gupta, "A Survey on Edge/Fog Caching Based on Deep Learning Algorithms," Arch Computat Methods Eng, Oct. 2025, doi: 10.1007/s11831-025-10417-2.

[2] Y. Khan *et al.*, "Content caching in mobile edge computing: A survey," *Cluster Comput.*, vol. 27, no. 7, pp. 8817–8864, Oct. 2024, doi: 10.1007/s10586-024-04459-7.

[3] Y. Xiao, Z. Lin, X. Cao, Y. Chen, and X. Lu, "AoI energy-efficient edge caching in AAV-assisted vehicular networks," *IEEE Internet Things J.*, vol. 12, no. 6, pp. 6764–6774, Mar. 2025, doi: 10.1109/JIOT.2024.3492535.

[4] J. Zhao, Y. Wang, X. Qin, Y. Yan, and Z. Fei, "Energy-efficient cache update and content delivery for optimizing information freshness of industrial applications," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 4508–4522, Feb. 2024, doi: 10.1109/JIOT.2023.3299505.

[5] W. Feng *et al.*, "Federated Deep Reinforcement Learning for Multimodal Content Caching in Edge-Cloud Networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 3, pp. 2188–2201, May 2025, doi: 10.1109/TNSE.2025.3545924.

[6] M. S. Hossain, Y. Hao, L. Hu, J. Liu, G. Wei, and C. Min, "Immersive multimedia service caching in edge cloud with renewable energy," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 20, no. 6, pp. 1–23, Jun. 2024, doi: 10.1145/3643818.

[7] S. Li, Y. Ma, Y. Zhang, and Y. Xie, "Towards Enhanced Energy Aware Resource Optimization for Edge Devices Through Multi-cluster Communication Systems," *J. Grid Comput.*, vol. 22, no. 2, p. 56, Jun. 2024, doi: 10.1007/s10723-024-09773-3.

[8] S. Alourani, M. Tahir, and M. S. Khan, "Dynamic and energy efficient cache scheduling framework for IoMT over ICN," *Appl. Sci.*, vol. 13, no. 21, p. 11840, Oct. 2023, doi: 10.3390/app132111840.

[9] S. Kumar and G. Bathla, "An efficient fuzzy hyper-edge clustering and popularity-based caching scheme for CCN-enabled IoT networks," *Multimedia Tools Appl.*, vol. 83, no. 15, pp. 44753–44780, Oct. 2023, doi: 10.1007/s11042-023-17284-8.

[10] V. Meena, K. Krithivasan, P. Rahul, and T. S. Praba, "Toward an intelligent cache management: In an edge computing era for delay sensitive IoT applications," *Wireless Pers. Commun.*, vol. 131, no. 2, pp. 1075–1088, Jul. 2023, doi: 10.1007/s11277-023-10469-2.

[11] S. Wang, H. Chen, and Y. Wang, "Collaborative caching for energy optimization in content-centric Internet of Things," *IEEE Trans. Comput. Soc. Syst.*, vol. 9, no. 1, pp. 230–238, Feb. 2022, doi: 10.1109/TCSS.2021.3087197.

[12] M. A. Naeem, R. Ullah, Y. Meng, R. Ali, and B. A. Lodhi, "Caching content on the network layer: A performance analysis of caching schemes in ICN-based Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6477–6495, May 2022, doi: 10.1109/JIOT.2021.3110977.

[13] X. Zhang, Y. Ren, J. Wang, and T. Lv, "Joint design of multicast transmission and in-network caching for green Internet of Things," *IEEE Sensors J.*, vol. 22, no. 12, pp. 12404–12414, Jun. 2022, doi: 10.1109/JSEN.2022.3174156.

[14] F. Javed *et al.*, "A novel energy-efficient reservation system for edge computing in 6G vehicular ad hoc network," *Sensors*, vol. 23, no. 13, p. 5817, Jun. 2023, doi: 10.3390/s23135817.

[15] X. Xia *et al.*, "OL-MEDC: An online approach for cost-effective data caching in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2021, doi: 10.1109/TMC.2021.3107918.

[16] U. U. Tariq *et al.*, "Energy-aware scheduling of streaming applications on edge-devices in IoT-based healthcare," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 2, pp. 803–815, Jun. 2021, doi: 10.1109/TGCN.2021.3056479.

[17] Y. Zhao, W. Zhang, L. Zhou, and W. Cao, "A survey on caching in mobile edge computing," *Wireless Commun. Mobile Comput.*, vol. 2021, no. 1, p. 5565648, Jan. 2021, doi: 10.1155/2021/5565648.

[18] F. Al-Turjman, B. D. Deebak, and L. Mostarda, "Energy aware resource allocation in multi-hop multimedia routing via the smart edge device," *IEEE Access*, vol. 7, pp. 151203–151214, 2019, doi: 10.1109/ACCESS.2019.2945797.

[19] S. S. Jazaeri, P. Asghari, S. Jabbehdari, and H. H. S. Javadi, "Toward caching techniques in edge computing over SDN-IoT architecture: A review of challenges, solutions, and open issues," *Multimedia Tools Appl.*, vol. 83, no. 1, pp. 1311–1377, Jan. 2024, doi: 10.1007/s11042-023-15657-7.

[20] F. Gabry, V. Bioglio, and I. Land, "On energy-efficient edge caching in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3288–3298, Dec. 2016, doi: 10.1109/JSAC.2016.2611845.

[21] Y. Tang *et al.*, "A smart caching mechanism for mobile multimedia in information centric networking with edge computing," *Future Gener. Comput. Syst.*, vol. 91, pp. 590–600, Feb. 2019, doi: 10.1016/j.future.2018.08.019.

[22] D. Gupta *et al.*, "Edge caching based on collaborative filtering for heterogeneous ICN-IoT applications," *Sensors*, vol. 21, no. 16, p. 5491, Aug. 2021, doi: 10.3390/s21165491.

[23] X. Wang and X. You, "Efficient data sharing and caching for information-centric IoT," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 18074–18081, May 2024, doi: 10.1109/JIOT.2024.3360218.

[24] H. Asmat *et al.*, "Energy-efficient centrally controlled caching contents for information-centric Internet of Things," *IEEE Access*, vol. 8, pp. 126358–126369, 2020, doi: 10.1109/ACCESS.2020.3008193.

[25] H. Asmat *et al.*, "Enhancing edge-linked caching in information-centric networking for Internet of Things with deep reinforcement learning," *IEEE Access*, vol. 12, pp. 154918–154932, 2024, doi: 10.1109/ACCESS.2024.3483455.

[26] S. Gupta *et al.*, "Intelligent resource optimization for scalable and energy-efficient heterogeneous IoT devices," *Multimedia Tools Appl.*, vol. 83, no. 35, pp. 82343–82367, Mar. 2024, doi: 10.1007/s11042-024-18176-1.

[27] J. Xu, K. Ota, and M. Dong, "Saving energy on the edge: In-memory caching for multi-tier heterogeneous networks," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 102–107, May 2018, doi: 10.1109/MCOM.2018.1700909.

[28] O. A. Khashan *et al.*, "Smart energy-efficient encryption for wireless multimedia sensor networks using deep learning," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 5745–5763, 2024, doi: 10.1109/OJCOMS.2024.3442855.

[29] Y. Choi and Y. Lim, "Deep reinforcement learning for edge caching with mobility prediction in vehicular networks," *Sensors*, vol. 23, no. 3, p. 1732, 2023, doi: 10.3390/s23031732.

[30] H. Sakr and M. Elsabrouty, "Meta-reinforcement learning for edge caching in vehicular networks," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, pp. 4607–4619, 2023, doi: 10.1007/s12652-023-04583-z.

[31] "Cooperative caching algorithm based on multi-agent meta reinforcement learning," *Computer Networks*, vol. 242, p. 110247, 2024, doi: 10.1016/j.comnet.2024.110247.

[32] X. Teng *et al.*, "Attention mechanism-aided deep reinforcement learning for dynamic edge caching," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10197–10213, 2024, doi: 10.1109/JIOT.2023.3321234.

[33] Z. Zhang *et al.*, "Multi-agent deep reinforcement learning for edge caching in Internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8324–8338, 2023, doi: 10.1109/TITS.2023.3245678.

[34] "Distributed service caching with deep reinforcement learning for sustainable edge computing," *Digit. Commun. Netw.*, vol. 11, no. 5, pp. 1447–1456, 2025, doi: 10.1016/j.dcan.2024.08.012.

[35] Y. Wang, Z. Wang, B. Hu, Y. Huang, T. Du, and E. H. Yap, "Double Deep Q Network-Enabled Low-Altitude Wireless Caching Networks Optimisation," in *Proc. 2025 Int. Conf. Computer, Internet of Things and Smart City (CIoTSC)*, Suzhou, China, 2025, pp. 1–6, doi: 10.1109/CIoTSC67482.2025.11413024.

[36] M. Ottens, J. Deutschmann, K.-S. Hielscher, and R. German, "Performance Evaluation of ns-3 Real-Time Emulation," *IEEE Access*, vol. 13, pp. 59544–59559, 2025, doi: 10.1109/ACCESS.2025.3555478.

[37] L. Salvati, L. S. Alaimo, and A. Muolo, "An operational classification of rural and urban areas based on Zipf's Law," *Annals of Operations Research*, 2025, doi: 10.1007/s10479-025-06859-3.

Shilpa Bagade is currently working as an Associate Professor at Malla Reddy University. She has 15 years of teaching and research experience and has served at various institutions. She received the B.Tech degree in Electronics and Communication Engineering and the M.E. degree in Microwave and Radar Engineering. She completed the Ph.D. degree from KL University with research focused on optimal video data transmission methods in 5G networks using an improved H.265 protocol. Her research interests include 5G communication networks, Internet of Things (IoT), vehicular communication systems, edge computing, artificial intelligence, wireless sensor networks, and multimedia data transmission. She has published research articles in reputed journals and conferences.

Anjani Devi Thanneru received the Ph.D. degree in VLSI Systems from GITAM University, Visakhapatnam, India. She is currently working as an Assistant Professor at Vasavi College of Engineering, India, and has 15 years of teaching and research experience. Her research interests include Internet of Things (IoT), VLSI systems, and embedded systems. She has published research articles in reputed journals and conferences.

R. Sahith received the M.Tech degree in Computer Science and Engineering from JNTUH, India, and is currently pursuing the Ph.D. degree in Informatics from the Department of Computer Science and Engineering, Osmania University, Hyderabad, India. He is currently working as a Senior Assistant Professor in the Department of Computer Science and Engineering at CVR College of Engineering, Hyderabad, India. His research interests include programming and machine learning.

V. Bharathi Devarakonda is currently working as an Assistant Professor in the Department of Electronics and Communication Engineering at Aditya University. She received the B.Tech degree in Electronics and Communication Engineering from Kakinada Institute of Engineering and Technology and the M.Tech degree in VLSI System Design from Pragati Engineering College. Her academic interests include low-power VLSI design and digital electronics. She has published research articles in national and international journals and has participated in conferences, seminars, workshops, and faculty development programs.

Anna Geethanjali is currently working as an Assistant Professor in the Department of Electronics and Communication Engineering at Vardhaman College of Engineering, Hyderabad. She received the M.Tech degree in VLSI System Design from Aurora's Scientific and Technological Research Academy, Hyderabad, affiliated to JNTUH, in 2015, and the B.Tech degree in Electronics and Communication Engineering from Sree Chaitanya College of Engineering, JNTUH, in 2012. She has eight years of teaching experience. Her research interests include low-power VLSI design. She has published research articles in national and international journals and has participated in conferences, seminars, and faculty development programs.