



ARTICLE

ECRS-TRPO: An Edge Cache Replacement Strategy Based on Trust Region Policy Optimization

Yuan Feng¹, Jiahao Ying², Yahui Sun³, Jianwei Zhang^{3,*} and Zengyu Cai^{2,*}

¹School of Electronic Information, Zhengzhou University of Light Industry, Zhengzhou, China

²School of Computer Science and Artificial Intelligence, Zhengzhou University of Light Industry, Zhengzhou, China

³College of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou, China

*Corresponding Authors: Jianwei Zhang. Email: mailzjw@163.com; Zengyu Cai. Email: mailczy@163.com

Received: 27 May 2026; Accepted: 14 August 2026; Published: 22 October 2026

ABSTRACT: Edge caching is a key technology for improving video transmission efficiency, but traditional cache replacement strategies struggle to adapt to dynamic network environments and heterogeneous user requests. To address this issue, a novel cache replacement framework named Edge Cache Replacement Strategy based on Trust Region Policy Optimization (ECRS-TRPO) is proposed. This framework leverages Trust Region Policy Optimization (TRPO), a deep reinforcement learning algorithm, to dynamically adjust cached content. Through interactive learning between the agent and the caching environment, long-term rewards are maximized. Furthermore, a Transformer-based soft hit mechanism is introduced. When a cache miss occurs, similar content is provided as a fast alternative access, thereby reducing transmission latency. Experimental results on two benchmark datasets show that the proposed ECRS-TRPO achieves average hit rate improvements of 3.75% and 11.9% over seven baseline models under increasing cache capacity settings, respectively. These results verify its superior performance in edge cache replacement tasks.

KEYWORDS: Edge network; trust region policy optimization; soft hit mechanism; cache replacement; deep reinforcement learning

1 Introduction

With the rapid development of 5G technology, network traffic has grown explosively, and video content transmission has become the dominant component [1]. Edge computing, which proactively caches user-preferred content at the network edge, has emerged as a key technology for alleviating network congestion and reducing latency [2]. However, edge nodes have limited storage capacity. When the cache becomes full, decisions must be made regarding which content to evict. This decision-making process directly affects system performance. Traditional cache replacement strategies, such as Least Recently Used (LRU) and Least Frequently Used (LFU), are simple and easy to implement, but they often fail to achieve optimal performance in dynamic environments. For instance, under non-stationary request patterns in mobile edge networks, LRU often exhibits significant performance degradation compared to its performance under stable patterns. Such degradation arises because recency-based eviction cannot capture abrupt shifts in content popularity or the spatial-temporal locality of user interests across edge servers, and this issue is further amplified in heterogeneous environments where user preferences vary significantly across regions and time slots.

Reinforcement learning, which learns policies through interaction with the environment and progressively optimizes behavior, is well-suited for edge cache replacement tasks. To this end, a cache

management framework named Edge Cache Replacement Strategy based on Trust Region Policy Optimization (ECRS-TRPO) is proposed. This framework adopts Trust Region Policy Optimization (TRPO) as its core learning algorithm. Unlike generic deep reinforcement learning approaches, TRPO enforces a Kullback-Leibler (KL) divergence constraint during policy updates, ensuring stable and sample efficient learning. Furthermore, a Transformer-based soft hit mechanism is designed. Specifically, when the requested content is not cached but similar content exists in the cache, the similar content is recommended as an alternative. Consequently, backhaul requests and access latency are reduced.

The main contributions of this paper are as follows:

- (1) A TRPO-based edge cache replacement framework, named ECRS-TRPO, is proposed. By enabling the agent to interact and learn from the edge caching environment, the cache replacement strategy is adaptively optimized, and cache hit rates are significantly improved.
- (2) A Transformer-based soft hit mechanism is designed. This mechanism provides rapid alternative access upon cache misses, enhances user experience, and improves network resource utilization.
- (3) Extensive experiments are conducted on two benchmark datasets. The proposed method is compared with seven baseline models, demonstrating its effectiveness in edge video cache replacement tasks.

2 Related Work

This section provides an overview of related research in three main areas: mobile edge networks, reinforcement learning, and edge video cache replacement.

2.1 Mobile Edge Networks

Mobile Edge Computing (MEC) represents a key architectural paradigm that pushes computing and storage resources from centralized cloud data centers to the network edge, such as base stations or access points [3,4]. By deploying caching and content delivery functionalities closer to end users, this edge infrastructure significantly reduces service latency, alleviates backhaul bandwidth pressure, and improves the quality of experience for latency-sensitive applications, including video streaming, online gaming, and virtual reality. In a typical mobile edge network architecture, edge servers are co-located with base stations, enabling direct content delivery to mobile users within the same cell, while a central cloud server maintains a complete repository of all content.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make sequential decisions by interacting with its environment [5,6]. At each time step, the agent observes the current state of the environment, selects an action according to its policy, and receives a reward signal as feedback, while the environment transitions to a new state. The goal of the agent is to maximize the cumulative discounted reward over the long term. Unlike supervised learning, RL does not require labeled input-output pairs. Instead, it relies on a trial-and-error exploration process, making it particularly suitable for edge caching. Among various Deep Reinforcement Learning (DRL) algorithms, TRPO stands out for its stable policy updates via KL constraints, which is adopted as the core optimizer of the ECRS-TRPO framework.

2.3 Edge Video Cache Replacement

Cache replacement is a core mechanism in edge video caching, determining which content items to retain or evict when the cache becomes full. Traditional strategies, such as LRU [7], which evicts the content that has not been accessed for the longest time, and LFU [8], which evicts the content with the lowest access

frequency, are simple to implement but lack the ability to adapt to non-stationary request patterns or capture temporal correlations across different time scales.

With the rapid advancement of deep learning, DRL methods have been introduced to cache replacement tasks, demonstrating superior adaptability and generalization. For instance, Advantage Actor-Critic (A2C) [9] combines policy and value networks to learn from advantage functions, yet it suffers from slow convergence in large state spaces. Deep Q-Network (DQN) [10] approximates Q-values with deep neural networks but suffers from overestimation bias and sensitivity to hyperparameters. Proximal Policy Optimization (PPO) [11] uses a clipped surrogate objective to constrain policy updates, while Quantile Regression Deep Q-Network (QRDQN) [12] models Q-value distributions via quantile regression to handle reward uncertainty. Despite their improvements, these methods still face challenges in balancing policy stability, computational overhead, and long-term dependency modeling.

To address these limitations, the TRPO algorithm is adopted in this paper. TRPO enforces a KL-divergence constraint during policy updates, strictly controlling the update step size. This enhances training stability and sample efficiency, enabling the agent to better capture complex request patterns and dynamic content popularity in edge environments.

3 Research Methodology

In this chapter, the edge video caching architecture based on deep reinforcement learning is introduced in detail. The chapter is divided into three parts: the edge video caching replacement environment, the edge video caching strategy, and the ECRS-TRPO agent.

3.1 Edge Video Cache Replacement Environment

This paper focuses on an edge video caching system under a two-tier computing network architecture. Computing and caching services are migrated from the cloud to the network edge, for example, mobile base station (MBS), and edge servers support both direct hits and recommender-based soft hits. Users connect to base stations via wireless links. Due to the heterogeneity of user requests, the distribution of cached content across different edge servers exhibits diverse characteristics [13]. To reduce system costs and content management complexity, it is assumed that each user requests content directly from the locally serving edge server or from the cloud server.

3.1.1 Edge Caching Environment with Soft Hit Mechanism

The system comprises B base stations connected to a cloud server, each with an edge server of capacity K_b . Users request M video contents of size K_m . The cloud stores all contents. Edge servers cache a set F of contents. F terminal devices are randomly distributed. Each user is served by the nearest base station per time slot. User requests follow a Zipf distribution [14]:

$$p_i = \frac{R_i^{-\rho}}{\sum_{i=1}^I R_i^{-\rho}} \quad (1)$$

where ρ controls the skewness of the distribution, and R_i is the popularity rank of content i .

An edge video caching environment with a soft hit mechanism is further proposed. Given personalized user preferences, if an edge server does not cache the requested content but stores similar content, the system can recommend it as an alternative. Upon a request, the system first checks for a direct cache hit. If the content is not cached but similar content exists, the user may either fetch from the cloud, which incurs higher delay,

or accept the similar substitute for faster access. The architecture of the mobile edge caching environment with the soft hit mechanism is shown in Fig. 1.

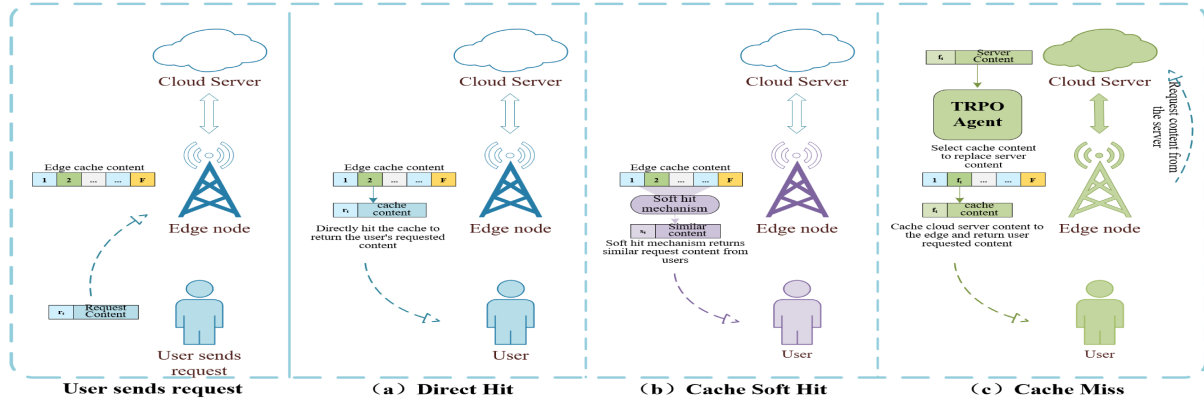


Figure 1: Edge cache environment architecture with soft hit mechanism.

Due to the introduction of the soft hit mechanism, the transmission delay of user-requested content needs to consider three scenarios. When the requested content is already cached on the MBS, a direct hit, the delay is $T_{1,u,i} = \frac{d_i}{c_u}$. When the edge server does not cache the requested content but stores similar content, a soft hit, the delay is $T_{2,u,i} = \frac{d_i}{c_u}$, with acceptability α_i . When neither the requested content nor any similar content is cached, a cache miss, the user must fetch from the cloud and the ECRS-TRPO agent selects a cached item to replace, resulting in delay $T_{3,u,i} = \frac{d_i}{c_u} + \frac{d_i}{r_i}$. Here d_i is content size, c_u the transmission rate, and r_i the backhaul rate.

Let x_i denote the cache state of content i , where 1 indicates cached, and y_i the soft hit state, where 1 indicates soft hit. The total transmission delay is:

$$T_{u,i} = x_i T_{1,u,i} + (1 - x_i) y_i T_{2,u,i} + (1 - x_i)(1 - y_i) T_{3,u,i} \quad (2)$$

The soft hit mechanism provides a fast alternative in case of a miss, effectively reducing delay and alleviating cloud burden [15].

3.1.2 Transformer-Based Soft Hit Mechanism

To reduce backhaul requests and improve cache hit rates, a soft-hit mechanism based on content similarity matching and temporal recommendation models is proposed. Methods based solely on static similarity, such as cosine similarity, fail to capture the temporal evolution of user interests [16]. Therefore, a Transformer-based sequential recommendation model [17] is introduced to capture the time-dependent nature of user behavior.

A rating matrix R is constructed from user historical interactions, where $R_{u,c}$ is the rating, with 0 indicating no interaction. The similarity between contents is computed using cosine similarity as standard definition. When user-requested content c_q is not cached, the top k most similar contents are retrieved as the candidate set.

Due to the dynamic nature of user interests, relying solely on static similarity for recommendations fails to meet personalized needs. Therefore, after obtaining the candidate set, a Transformer-based sequential recommendation model [17] is further introduced to rank the candidate contents, improving the accuracy of the recommendations.

Let $I_s = \{c_{t-n}, c_{t-n+1}, \dots, c_{t-1}\}$ be the historical behavior sequence of user u , where n denotes the historical window size. The Transformer encoder is then employed to extract the user’s current interest state h_u :

$$h_u = \text{Transformer}(I_s) \quad (3)$$

Next, for each candidate content c'_i in the candidate set, we match it with the user’s interest representation h_u to measure its suitability as a recommendation item:

$$s_i = V \cdot \Phi(W_1 \cdot h_u + W_2 \cdot c'_i + b) \quad (4)$$

where V , W_1 and W_2 are parameters, Φ is the ReLU activation function, and b is the bias term.

A score vector $S = \{s_1, s_2, \dots, s_k\}$ for the candidate contents is obtained. To further integrate content similarity with user preferences, a weighted fusion mechanism is introduced. Both the content similarity $S(c_q, c'_i)$ and the user interest matching score s_i computed by the Transformer are considered, resulting in the final recommendation score P_i for each candidate content:

$$P_i = \lambda \cdot S(c_q, c'_i) + (1 - \lambda) \cdot s_i \quad (5)$$

where λ is a weight parameter that controls the balance between similarity matching and personalized recommendation.

Finally, the candidate contents are ranked based on the aggregated score P_i , and the top k_{fin} contents are selected as the final soft hit recommendations.

3.2 ECRS-TRPO: TRPO-Based Cache Replacement Agent

Trust Region Policy Optimization (TRPO) is an advanced deep reinforcement learning algorithm that introduces a KL divergence constraint by limiting the magnitude of policy updates [18]. This ensures the stability of the learning process while improving policy performance. In this paper, TRPO is applied to the edge cache replacement problem within the ECRS-TRPO framework, enabling the agent to learn user request patterns and network states in real time, thereby optimizing the cache replacement strategy and improving content recommendation and cache management efficiency. The architecture of the ECRS-TRPO agent is shown in Fig. 2.

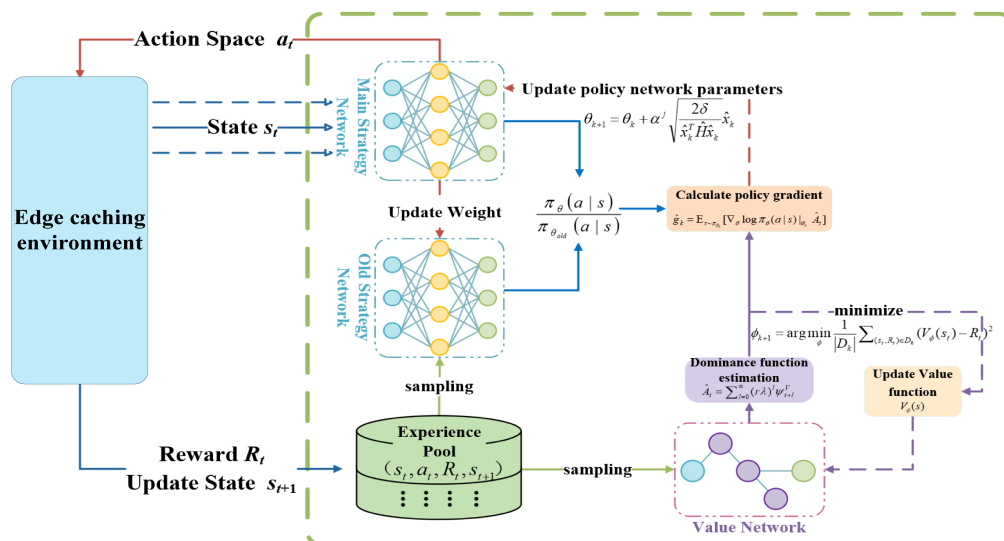


Figure 2: Architecture of the ECRS-TRPO Agent.

3.2.1 Policy Representation and Optimization

In the edge cache replacement task, the policy network $\pi_\theta(a|s)$ represents the probability distribution over actions a given a state s . The objective of policy optimization is to maximize the expected cumulative future reward $\eta(\pi_\theta)$:

$$\eta(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t R_t \right] \quad (6)$$

here, $\mathbb{E}$ denotes the expectation over trajectories τ sampled from the policy π_θ , $\gamma \in [0, 1]$ is the discount factor used to balance short-term and long-term rewards, and R_t is the immediate reward at time step t .

To ensure stability during policy updates, a KL divergence constraint is imposed [18]:

$$\mathbb{E}_t [D_{KL}(\pi_{\theta_{old}} \parallel \pi_\theta)] \leq \delta \quad (7)$$

where D_{KL} measures the divergence between the new policy π_θ and the old policy $\pi_{\theta_{old}}$, and δ is the maximum allowable KL divergence, typically set to 0.01.

3.2.2 Advantage Function Estimation

The advantage function $\hat{A}(s, a)$ is used to measure the relative performance of action a in state s , serving as guidance for policy optimization. Let $Q(s, a)$ denote the state-action value function and $V(s)$ the state value function, defined as the expected discounted cumulative reward. The advantage function is given by:

$$\hat{A}(s, a) = Q(s, a) - V(s) \quad (8)$$

here, when $\hat{A}(s, a) > 0$, action a outperforms the average, else, it performs worse.

In practice, TRPO typically adopts Generalized Advantage Estimation (GAE) [19] to balance bias and variance in advantage function estimation. The formula is:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \psi_{t+l}^V \quad (9)$$

here, γ is the discount factor controlling the importance of future rewards, range $[0, 1]$, and λ is the smoothing factor, default 1, used to trade off bias and variance. The term $\psi_t^V = R_t + \gamma V(s_{t+1}) - V(s_t)$ represents the temporal-difference (TD) error at time step t [20], capturing the discrepancy between the immediate reward and the estimated value of the next state.

3.2.3 Policy Update

In TRPO, policy parameters are updated using the policy gradient method. The policy gradient is:

$$\hat{g}_k = \mathbb{E}_{\tau \sim \pi_{\theta_k}} \left[\nabla_\theta \log \pi_\theta(a|s) |_{\theta_k} \hat{A}_t \right] \quad (10)$$

here, $\hat{g}_k$ is the policy gradient, $\nabla_\theta \log \pi_\theta(a|s)$ represents the gradient of the log-probability of action a with respect to the policy parameters θ , indicating the direction to update the parameters θ .

To satisfy the KL constraint, TRPO uses the conjugate gradient method to compute the update direction, followed by backtracking line search for step size. The parameter update is:

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H} \hat{x}_k}} \hat{x}_k \quad (11)$$

where $\hat{x}_k$ is the approximate search direction, $\hat{H}$ is the Hessian matrix of the KL divergence [21], and α^j is the step size determined by line search.

3.2.4 Value Function Update

The objective of the value function $V(s)$ is to estimate the long-term cumulative reward of state s , serving as a critical component in policy evaluation. In TRPO, a separate value function network $V_\phi(s)$ is trained to estimate state values. The optimization objective is defined as:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|} \sum_{(s_t, R_t) \in \mathcal{D}} (V_\phi(s_t) - R_t)^2 \quad (12)$$

here, ϕ denotes the parameters of the value function network, $\mathcal{D}_k$ represents the sampled trajectory data, and R_t is the discounted cumulative reward at time step t .

By minimizing MSE, the value network accurately estimates state returns, facilitating advantage computation. In summary, TRPO integrates KL constraints, policy gradient, and value fitting to adaptively optimize edge caching, improving hit rates and system performance.

3.3 Edge Video Caching Strategy

To improve the utilization efficiency of edge cache space, an effective cache replacement strategy that can adapt to dynamically changing user requests is necessary. Upon each request arrival, the base station must determine whether the requested content already exists in the cache. If not, it must decide, based on the policy, whether to replace existing cached content to accommodate the new request.

The proposed ECRS-TRPO is modeled as a Markov Decision Process (MDP) [22], which consists of three components: state representation, action space, and reward function, as is shown in Fig. 3.

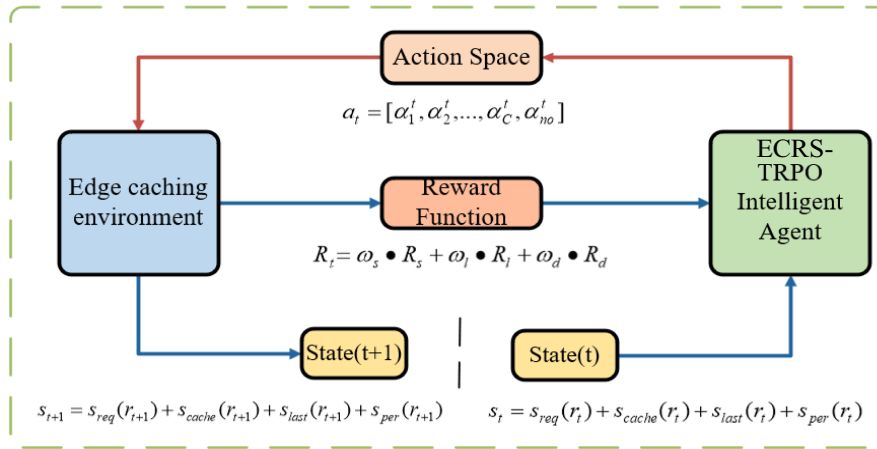


Figure 3: Edge caching strategy of ECRS-TRPO.

3.3.1 Action Space

In cache replacement, the action space defines the set of operations an agent can execute to optimize system performance. A discrete replacement strategy is adopted: upon a cache miss, the agent selects a cache slot to replace or maintains the current cache state. Let cache capacity be C and assume one request per time slot. At time step t , the action space is:

$$a_t = [\alpha_1^t, \alpha_2^t, \dots, \alpha_C^t, \alpha_{no}^t] \quad (13)$$

where α_i^t indicates whether to replace the content in slot i .

If selected, the requested content replaces slot i , and the resulting state s_{t+1} is evaluated by the RL model. The no-operation action α_{no}^t preserves the current cache state and ignores the request. Action selection is subject to three constraints:

- (1) At most one replacement per decision step.
- (2) α_i^t is allowed only when the requested content yields neither a direct hit nor a soft hit.
- (3) The agent is encouraged to evict an appropriate cached content to maintain cache quality.

3.3.2 State Representation

To effectively model the state representation in the cache management task, a multidimensional cache feature vector is designed based on request history and cache slots. This vector aims to comprehensively capture the current state of the caching environment.

First, basic access features are constructed to describe the access frequency of both the current request and the resources in cache slots. Let the cache capacity be C , and divide the request history window into short-term (T_s), medium-term (T_m), and long-term (T_l) periods. The current request frequency feature s_{req} records the number of times the requested resource r_t has been accessed within the three time windows:

$$s_{req}(r_t) = [f_{T_s}(r_t), f_{T_m}(r_t), f_{T_l}(r_t)] \quad (14)$$

where r_t is the requested content and f_T counts accesses within window T .

The cache slot frequency feature captures access counts for each cached resource:

$$s_{cache}(r_t) = [f_{T_s}(c_1), f_{T_m}(c_1), f_{T_l}(c_1), f_{T_s}(c_2), \dots, f_{T_s}(c_C), f_{T_m}(c_C), f_{T_l}(c_C)] \quad (15)$$

here, c_i denotes the resource stored in cache slot i . The basic feature dimension is $(1 + C) \times 3$.

To capture temporal dynamics, two additional features are introduced. The last access time feature:

$$s_{last}(r_t) = [l_1, l_2, \dots, l_C] \quad (16)$$

where l_i denotes the time interval since the last access of the resource in cache slot i .

The residence time feature:

$$s_{per}(r_t) = [p_1, p_2, \dots, p_C] \quad (17)$$

where p_i indicates the current residence time of the resource in slot i .

By concatenating all the above features, the final state representation s_t is formulated as:

$$s_t = s_{req}(r_t) + s_{cache}(r_t) + s_{last}(r_t) + s_{per}(r_t) \quad (18)$$

With total dimension:

$$\dim(s_t) = [(1 + C) \times 3] + [2 \times C] = 5C + 3 \quad (19)$$

This representation integrates historical access frequency and temporal evolution, enabling the DRL agent to perceive both spatial and temporal dynamics of the caching system, thereby improving policy accuracy.

3.3.3 Reward Function

In the framework of deep reinforcement learning, the reward function is the core mechanism for policy optimization by the agent, responsible for evaluating the effectiveness of each action. To guide the agent in balancing short-term hit rate, long-term hit potential, and content diversity in the cache replacement task, a multi-objective integrated reward function is designed. This function consists of three components: short-term reward, long-term reward, and diversity reward. It is defined as follows:

$$R_t = w_s \cdot R_s + w_l \cdot R_l + w_d \cdot R_d \quad (20)$$

here, R_s denotes the short-term reward, which evaluates the immediate benefits of cache hits and replacement behavior at the current time step. R_l represents the long-term reward, which reflects the potential cache hit capability over a future time window, and R_d indicates the diversity reward, encouraging diversification of cached content. The corresponding weight coefficients are set as $w_s = 0.5$, $w_l = 0.3$, and $w_d = 0.2$.

The short-term reward assesses the impact of cache hits and replacement behavior on performance at the current time step and is defined as:

$$R_s = H_t \cdot r_{hit} + \Delta_{miss} \cdot p_{miss} + \Delta_{soft} \cdot r_{soft} + R_{future} + R_{rep-pen} \quad (21)$$

where H_t is the number of cache hits at the current time step, and r_{hit} is the reward for each hit, with default value 1. Δ_{miss} is the variation in the number of cache misses, with p_{miss} as the penalty per miss, default -1 . Δ_{soft} indicates the change in soft hits, and r_{soft} is the reward per soft hit, default 0.3. The future hit reward is $R_{future} = \alpha \cdot f_{past}(r_{in})$ and the replacement penalty is $R_{rep-pen} = \beta \cdot p_{miss}(r_{out})$, with $\alpha = 0.5$ and $\beta = 10$. Here $f_{past}(r_{in})$ is the past request frequency of the incoming content, and $p_{miss}(r_{out})$ is an extra penalty if the evicted content is requested immediately after replacement.

The long-term reward evaluates the potential cache hit situation over a future time window and is defined as:

$$R_l = \gamma \cdot \sum_{t=1}^{T_f} \prod(r_t \in cache) \quad (22)$$

where γ is the scaling factor used to normalize the long-term reward, default 0.001, T_f represents the length of the future time window, default 1000, and $\prod(r_t \in cache)$ is an indicator function that equals 1 if resource r_t is present in the cache and 0 otherwise.

The diversity reward encourages the cache to store a broader variety of resources to meet diverse user demands. It is defined as:

$$R_d = \delta \cdot \frac{|\text{unique}(cache)|}{C} \quad (23)$$

where δ is the weighting factor for the diversity reward, default 0.2, $|\text{unique}(cache)|$ denotes the number of distinct resources in the cache, and C is the cache capacity.

To adapt to dynamically changing request patterns, the hit reward and miss penalty are adaptively adjusted based on the current cache hit rate η_t . The details are omitted for brevity.

In summary, the proposed reward function comprehensively considers immediate caching performance, long-term benefits, and content coverage capability, enabling the agent to effectively balance multiple objectives and improve both the performance and generalizability of the overall caching strategy.

4 Experiments and Analysis

This section presents experimental studies on seven baseline models using two benchmark datasets. The datasets, evaluation metrics, parameter settings, baseline models, and experimental results are described.

4.1 Datasets

Two real-world recommender system benchmark datasets are used: MovieLens 10M and Tenrec. They differ significantly in data scale, user behavior patterns, and sparsity, enabling a comprehensive evaluation of the adaptability and robustness of caching strategies. To prevent temporal leakage, the timestamp-sorted request sequence of each dataset was chronologically split into training (80%), validation (10%), and test (10%) sets, ensuring all test requests occur later in time than training requests.

MovieLens 10M (ML-10M): A standard movie rating benchmark in recommender system research with relatively stable user request behaviors. During preprocessing, ratings equal to or greater than 3 were retained, the data were sorted by user ID and timestamp, the most recent 10 interactions per user were kept, and a global sort by timestamp was performed to form a continuous request sequence.

Tenrec: A large-scale multi-platform recommendation benchmark proposed jointly by Tencent and Westlake University, covering a wide range of recommendation scenarios with diverse user behavior patterns [23]. Preprocessing followed the same steps as for MovieLens.

The statistics of the preprocessed datasets are shown in Table 1.

Table 1: Statistics of the preprocessed dataset.

Datasets	Users	Items	Inters	Avg u	Avg i	Sparsity
ML-10M	6955	5700	69,550	10.00	12.20	99.82%
Tenrec	11,361	20,535	104,889	9.23	5.11	99.96%

Note: Avg u represents the average number of interactions per user, and Avg i represents the average number of interactions per item.

4.2 Performance Evaluation Metrics

Hit Rate (HR) is adopted as the primary evaluation metric:

$$HR = \frac{\text{Hit Count}}{\text{Total Count}} \times 100\% \quad (24)$$

where *Hit Count* denotes the number of contents successfully returned from the cache, and *Total Count* represents the total number of user requests.

A higher hit rate indicates more efficient cache resource management. Due to the inherent randomness of deep reinforcement learning methods, each experiment is repeated 20 times. Both the mean and median hit rates are reported, reflecting the expected performance and robustness of the model, respectively. The values following the mean values represent the half-widths of the 95% confidence intervals.

4.3 Parameter Settings and Optimization

Experiments were conducted in a custom-built mobile edge video caching simulation environment using the PyTorch framework on an NVIDIA RTX 3090 GPU.

All DRL models shared consistent settings: cache size varied by dataset, and the number of training steps was uniformly set to 500,000. For the soft-hit mechanism, the maximum candidate set for similarity

matching was $k = 100$, and the final number of selected soft hits was $k_{fin} = 3$. Multiple temporal windows were used: short-term 100, mid-term 300, and long-term 1000. The core hyperparameter settings of the ECRS-TRPO framework are presented in [Table 2](#).

Table 2: Core hyperparameter settings of the ECRS-TRPO framework.

Parameter	Value	Physical Meaning
GAE smoothing factor	$\lambda_{GAE} = 1.0$	Controls the trade-off between bias and variance in advantage estimation
KL divergence constraint limit	$\delta_{KL} = 0.01$	Restricts the difference between old and new policy distributions to ensure stable updates
Short-term reward weight	$w_s = 0.5$	Weight for immediate hit/miss feedback
Long-term reward weight	$w_l = 0.3$	Weight for guiding future potential hits
Diversity reward weight	$w_d = 0.2$	Weight for encouraging cached content diversity
Replacement penalty coefficient	$\beta = 10.0$	Extra penalty for evicting frequently requested content
Total training timesteps	500,000	Ensures sufficient policy convergence
Independent runs	20	Enables statistical significance testing

For evaluation, all models were trained offline on the training set, with the validation set used for model selection. After training, the learned policy was evaluated on the test set without further updates. The cache was initialized as empty, and an episode was defined as a full traversal of the dataset. Training was set to 500,000 steps, and 20 test episodes were run per configuration.

Baseline models included LRU, LFU, Random, A2C, DQN, PPO, and QRDQN. Each DRL model was independently run 20 times per dataset. LRU, LFU are deterministic policies and were run once.

4.4 Baseline Models

To validate the superior performance of the proposed method, three basic caching algorithms and four reinforcement learning-based caching algorithms are selected as baseline models:

LRU [7]: Removes the least recently used content.

LFU [8]: Removes the content with the lowest access frequency.

Random: Randomly selects a cached item for replacement.

A2C [9]: An actor-critic based reinforcement learning algorithm.

DQN [10]: A value-based deep Q-network algorithm.

PPO [11]: A policy optimization algorithm that uses a clipped surrogate objective to constrain policy updates.

QRDQN [12]: An improved DQN using quantile regression to model Q-value distributions.

4.5 Experimental Results and Analysis

The mean and median hit rates under varying cache sizes on the MovieLens 10M and Tenrec datasets were evaluated. The effect of the soft-hit mechanism was also analyzed separately.

4.5.1 Evaluation on the ML-10M Dataset

The mean and median hit rates of all models under different cache sizes (from 400 to 800) are presented in Table 3. ECRS-TRPO consistently outperformed all other models across all settings.

Table 3: Comparison of mean and median hit rates on the ML-10M dataset.

Metric	LRU	LFU	Random	A2C	DQN	PPO	QRDQN	ECRS-TRPO	Improve
Mean@400	0.5291	0.4825	0.4838	0.4813 $\pm$ 0.0008	0.6012 $\pm$ 0.0032	0.6324 $\pm$ 0.0029	0.5435 $\pm$ 0.0050	0.6359 $\pm$ 0.0022	+0.55%
Mean@500	0.5880	0.5408	0.5418	0.5183 $\pm$ 0.0010	0.6548 $\pm$ 0.0034	0.6824 $\pm$ 0.0030	0.5859 $\pm$ 0.0045	0.6935 $\pm$ 0.0017	+1.63%
Mean@600	0.6351	0.5988	0.5877	0.5507 $\pm$ 0.0004	0.6894 $\pm$ 0.0064	0.7177 $\pm$ 0.0032	0.6201 $\pm$ 0.0051	0.7440 $\pm$ 0.0018	+3.66%
Mean@700	0.6731	0.6452	0.6255	0.6256 $\pm$ 0.0003	0.7323 $\pm$ 0.0040	0.7644 $\pm$ 0.0014	0.6749 $\pm$ 0.0044	0.7872 $\pm$ 0.0014	+2.98%
Mean@800	0.7058	0.6799	0.6573	0.6692 $\pm$ 0.0004	0.7730 $\pm$ 0.0028	0.8133 $\pm$ 0.0011	0.7133 $\pm$ 0.0028	0.8245 $\pm$ 0.0007	+1.38%
Median@400	0.5291	0.4825	0.4836	0.4813	0.6022	0.6317	0.5423	0.6352	+0.55%
Median@500	0.5880	0.5408	0.5420	0.5183	0.6546	0.6823	0.5867	0.6930	+1.57%
Median@600	0.6303	0.5988	0.5873	0.5505	0.6888	0.7166	0.6196	0.7435	+3.75%
Median@700	0.6731	0.6452	0.6260	0.6255	0.7331	0.7639	0.6742	0.7875	+3.09%
Median@800	0.7058	0.6799	0.6572	0.6691	0.7723	0.8121	0.7106	0.8249	+1.58%

According to the analysis, the highest hit rates across all cache sizes were achieved by ECRS-TRPO. For example, under the Mean@600 and Median@600 metrics, ECRS-TRPO outperformed the second-best algorithm by 3.66% and 3.75%, respectively. Even under small cache size at 400, a modest improvement of 0.55% was achieved by ECRS-TRPO, reflecting its stability and reliability. As cache size increased, ECRS-TRPO maintained its advantage, achieving approximately 3.0% improvement at both Mean@700 and Median@700, validating its adaptability and effectiveness in larger cache capacity settings.

The trend of average hit rate (mean) on the ML-10M dataset is shown in Fig. 4. Traditional caching strategies, including LRU, LFU, and Random, consistently underperformed DRL models. Among DRL models, ECRS-TRPO maintained a leading position across the entire cache size range, with its average hit rate steadily increasing from about 65% to nearly 85%, demonstrating superior cache scheduling capability and generalization performance.

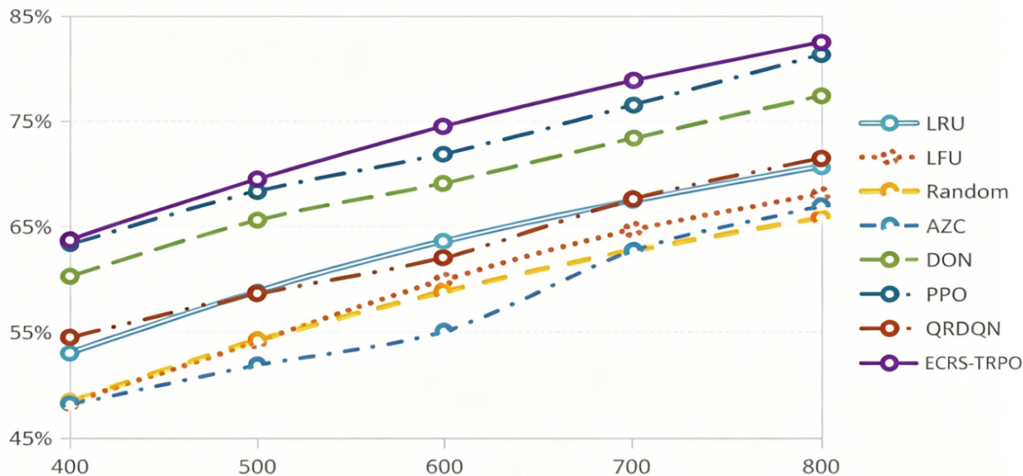


Figure 4: Trend of average hit rate (mean) on the ML-10M dataset.

Experiments on ML-10M show that ECRS-TRPO achieves higher hit rates and better scalability than existing methods in edge video cache replacement. As cache capacity increases, ECRS-TRPO maintains high hit rates with excellent stability, demonstrating strong potential for real-world deployment.

4.5.2 Evaluation on the Tenrec Dataset

The mean and median hit rates on the Tenrec dataset under different cache sizes (from 500 to 1500) are presented in Table 4. ECRS-TRPO again outperformed all other models across all settings.

Table 4: Comparison of mean and median hit rates on the Tenrec dataset.

Metric	LRU	LFU	Random	A2C	DQN	PPO	QRDQN	ECRS-TRPO	Improve
Mean@500	0.2851	0.4192	0.2506	0.3088 $\pm$ 0.0001	0.5378 $\pm$ 0.0044	0.6401 $\pm$ 0.0056	0.4083 $\pm$ 0.0051	0.6778 $\pm$ 0.0025	+5.89%
Mean@800	0.3907	0.5043	0.3396	0.3989 $\pm$ 0.0001	0.6105 $\pm$ 0.0044	0.6833 $\pm$ 0.0027	0.4862 $\pm$ 0.0069	0.7646 $\pm$ 0.0028	+11.90%
Mean@1000	0.4442	0.5430	0.3851	0.5218 $\pm$ 0.0009	0.6709 $\pm$ 0.0032	0.7433 $\pm$ 0.0033	0.5836 $\pm$ 0.0079	0.8044 $\pm$ 0.0030	+8.22%
Mean@1200	0.4889	0.5737	0.4242	0.6932 $\pm$ 0.0007	0.7521 $\pm$ 0.0034	0.8135 $\pm$ 0.0036	0.7149 $\pm$ 0.0032	0.8315 $\pm$ 0.0022	+2.21%
Mean@1500	0.5426	0.6121	0.4740	0.8663 $\pm$ 0.0005	0.8638 $\pm$ 0.0028	0.8755 $\pm$ 0.0033	0.8656 $\pm$ 0.0035	0.8826 $\pm$ 0.0021	+0.81%
Median@500	0.2851	0.4192	0.2504	0.3088	0.5322	0.6425	0.4089	0.6778	+5.49%
Median@800	0.3907	0.5043	0.3400	0.3964	0.6112	0.6838	0.4819	0.7652	+11.90%
Median@1000	0.4442	0.5430	0.3852	0.5143	0.6700	0.7442	0.5839	0.8043	+8.08%
Median@1200	0.4889	0.5737	0.4243	0.6931	0.7522	0.8138	0.7149	0.8315	+2.17%
Median@1500	0.5426	0.6121	0.4741	0.8663	0.8636	0.8753	0.8662	0.8836	+0.95%

The highest hit rates across all cache sizes were achieved by ECRS-TRPO. At cache size 500, a mean hit rate of 0.6778 was achieved by ECRS-TRPO, an improvement of 5.89%. When cache size increased to 800, its advantage became more pronounced, reaching Mean@800 of 0.7646 and Median@800 of 0.7652, an improvement of more than 11.9% over the second-best model, PPO [11]. This indicates that ECRS-TRPO possesses stronger policy learning capability under constrained resources and complex access patterns typical of edge environments. As cache size continued to increase from 1000 to 1500, ECRS-TRPO maintained its leading position. Overall, ECRS-TRPO proves highly competitive under small and medium cache capacities and exhibits stable and superior hit rate performance in large-capacity scenarios, further validating its effectiveness and advancement in optimizing cache strategies on real-world data such as Tenrec.

The trend of average hit rate (mean) on the Tenrec dataset is shown in Fig. 5. Traditional caching strategies performed relatively poorly overall. DRL-based models exhibited significantly better trends. Among them, ECRS-TRPO consistently outperformed all others across all cache capacities, with steadily increasing hit rates, demonstrating strong stability and adaptability. Both DQN and PPO showed smooth upward trends but slightly lagged behind ECRS-TRPO at each stage. The gap between ECRS-TRPO and PPO was most pronounced at cache sizes of 800 and 1000, indicating ECRS-TRPO's significant advantage under moderate cache conditions.

On Tenrec, ECRS-TRPO also achieved the best hit rates, validating its strong generalization and robustness under complex dynamic request patterns. The advantage is most significant under moderate cache capacities, showing that ECRS-TRPO can effectively adapt to diverse edge caching environments and provide stable, efficient management.

4.5.3 Validation of the Soft Hit Mechanism

A comparison between direct hits and soft hits under ECRS-TRPO on both datasets, along with the difference (soft minus direct), is shown in Fig. 6. The soft-hit mechanism consistently achieved a higher hit rate than the direct strategy across all datasets, indicating that soft hits effectively improve cache utilization. The difference between the two strategies shows distinct trends across datasets. On the ML-10M dataset, the benefit of soft hits remained relatively stable as cache size increased. On the Tenrec dataset, the gap gradually diminished but remained positive. These results further validate the effectiveness of the soft-hit mechanism

in enhancing cache strategy performance, particularly under complex data access patterns, by fully leveraging cache resources and improving overall hit rate, thereby enhancing system robustness and adaptability.

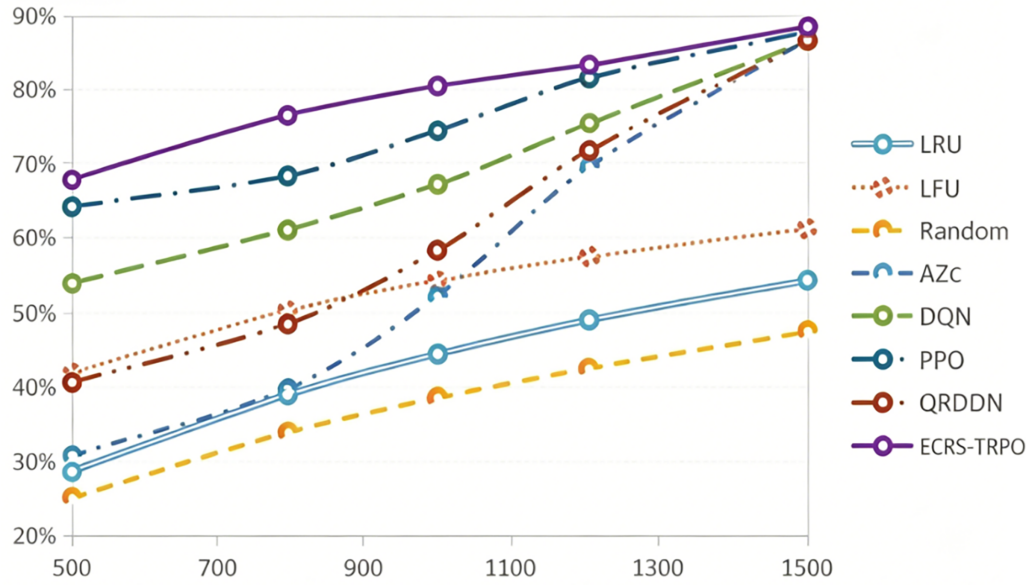


Figure 5: Trend of average hit rate (mean) on the Tenrec dataset.

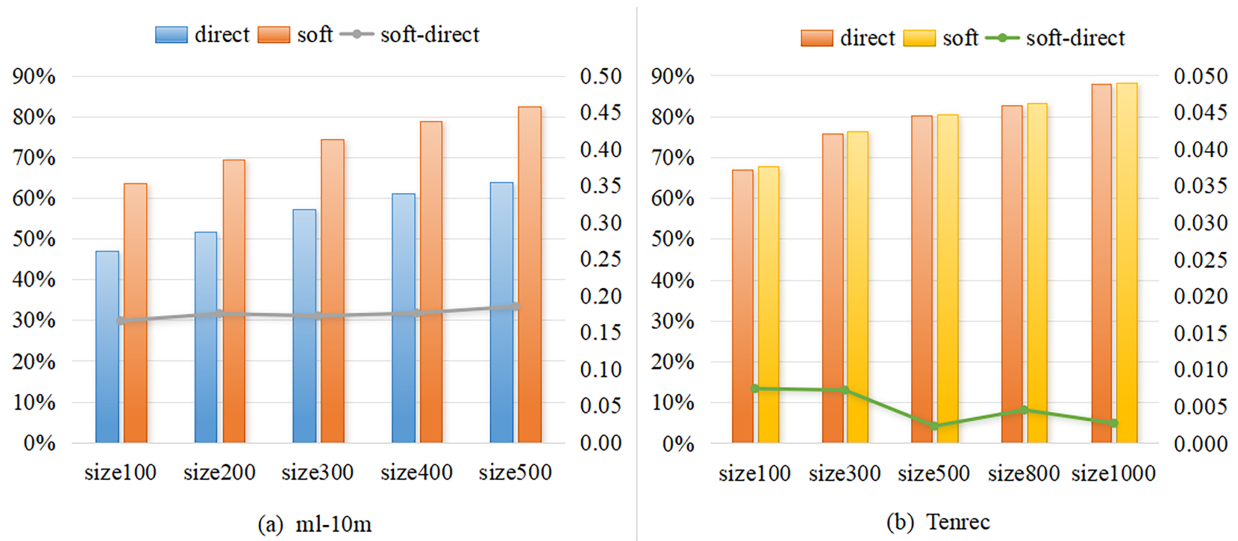


Figure 6: Comparison between soft hit and direct hit.

The proposed soft-hit mechanism consistently improves hit rates across all datasets and cache configurations. By recommending similar content upon cache misses, it effectively utilizes limited resources, reduces backhaul latency, and enhances system adaptability to diverse access patterns, further validating the superiority of the ECRS-TRPO framework.

4.5.4 Ablation Study

To quantify the contribution of each proposed component, we conducted a cumulative ablation experiment on the full MovieLens dataset with cache size fixed at 400. Starting from a baseline with DQN, base frequency features, and sparse reward, we successively added the multi-objective reward, full 5C + 3 state representation, Transformer-based soft hit, and TRPO.

As shown in Table 5, the hit rate improved from 59.12% to 63.89% as components were progressively integrated, yielding a cumulative gain of 8.07 percentage points. The full state representation contributed the largest gain, improving the hit rate by 3.34 percentage points over the previous configuration, confirming that the inclusion of last access time and cached duration features effectively captures the temporal dynamics of request patterns. The multi-objective reward delivered a gain of 1.73 percentage points, validating the benefit of long-term and diversity rewards in policy guidance. The soft hit mechanism further improved the hit rate by 2.37 percentage points, demonstrating that recommending similar content upon cache misses can effectively reduce backhaul overhead. TRPO added a gain of 0.42 percentage points, which aligns with its superior performance of 63.59% at cache size 400 in the main experiment. Additionally, as each component was progressively added, the standard deviation of hit rate decreased from 0.0031 to 0.0008, indicating that these modules not only improve hit rate but also enhance the stability of the learned policy.

Table 5: Cumulative ablation results on the ML-10M dataset.

Configuration	Hit Rate (Mean $\pm$ Std)	Improve
Baseline	0.5912 $\pm$ 0.0031	
+ Reward	0.6014 $\pm$ 0.0027	+1.73%
+ State	0.6215 $\pm$ 0.0018	+3.34%
+ SoftHit	0.6362 $\pm$ 0.0012	+2.37%
+ TRPO	0.6389 $\pm$ 0.0007	+0.42%

We present Cumulative ablation results on the MovieLens dataset with cache size 400. The baseline uses DQN with base frequency features and a sparse reward. The following components are incrementally added: multi-objective reward, full 5C + 3 dimensional state representation, Transformer-based soft hit mechanism, and TRPO.

5 Conclusion

In this paper, a deep reinforcement learning-based edge cache replacement strategy named ECRS-TRPO was proposed, which adopts Trust Region Policy Optimization as its policy optimizer. By enabling the TRPO agent to interact and learn adaptively from the edge caching environment, the cache replacement strategy was optimized, significantly improving both cache hit rate and overall system performance. Furthermore, by analyzing the unique requirements of edge caching systems, a soft-hit mechanism was introduced, which stores resources similar to user requests to enable faster alternative access. This not only enhanced user experience but also improved network resource utilization. Experimental results demonstrated the effectiveness of the proposed approach across multiple benchmark datasets.

Limitations

Despite the promising experimental results, several limitations of the ECRS-TRPO framework should be acknowledged regarding its practical deployment.

First, the TRPO algorithm requires second-order optimization involving Hessian matrix approximation, which incurs higher computational overhead compared to first-order methods such as PPO during the training phase. On edge servers with limited processing capacity, this high training cost may hinder frequent online policy updates or on-device retraining. A potential direction is to adopt more efficient trust-region implementations or to offload training to cloud servers while deploying only the trained policy for inference, since during inference both TRPO and PPO perform a forward pass of comparable complexity.

Second, the Transformer-based soft hit mechanism relies on sufficient historical interaction data to capture user preferences accurately. For cold-start users or newly published content with sparse historical records, the recommendation quality may degrade. Incorporating content-side information, such as titles and genres, or applying meta-learning techniques could help mitigate this issue.

Third, the current evaluation is conducted in a simulation environment with simplified network assumptions. Factors such as fluctuating wireless channel conditions, base station handover, and user mobility are not fully captured. Deploying and validating the proposed strategy on real edge server testbeds remains an essential step toward practical application.

In future work, the integration of richer user behavior data and dynamic network conditions will be explored to further improve the adaptability of the ECRS-TRPO framework. Additionally, multimodal data [24] and federated learning [25] will be incorporated to further optimize video recommendation and caching strategies, enhancing both video delivery efficiency and cache replacement performance.

Acknowledgement: The authors would like to thank the experimental team for their strong cooperation, the tutor for their full support in the research process.

Funding Statement: This research is partially supported by the National Natural Science Foundation of China (62072416), Key Research and Development Special Project of Henan Province (221111210500) and Key Technologies R&D Program of Henan Province (232102211053, 242102211071, 242102210142).

Author Contributions: Study conception and design: Yuan Feng, Jiahao Ying; data collection: Zengyu Cai; analysis and interpretation of results: Jianwei Zhang, Yahui Sun; draft manuscript preparation: Yuan Feng, Jiahao Ying. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data is available upon request from the authors.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cao K, Liu Y, Meng G, Sun Q. An overview on edge computing research. *IEEE Access*. 2020;8:85714–28. doi:10.1109/access.2020.2991734.
2. Walia GK, Kumar M, Gill SS. AI-empowered fog/edge resource management for IoT applications: a comprehensive review, research challenges, and future perspectives. *IEEE Commun Surv Tutor*. 2024;26(1):619–69. doi:10.1109/COMST.2023.3338015.
3. Mach P, Becvar Z. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor*. 2017;19(3):1628–56. doi:10.1109/COMST.2017.2682318.
4. Yang N, Chen S, Zhang H, Berry R. Beyond the edge: an advanced exploration of reinforcement learning for mobile edge computing, its applications, and future research trajectories. *IEEE Commun Surv Tutor*. 2025;27(1):546–94. doi:10.1109/COMST.2024.3405075.
5. Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. *IEEE Commun Surv Tutor*. 2017;19(4):2322–58. doi:10.1109/COMST.2017.2745201.

6. Sutton RS, Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge, MA, USA: The MIT Press; 2018.
7. Zhao Y, Zhang W, Zhou L, Cao W. A survey on caching in mobile edge computing. *Wirel Commun Mob Comput*. 2021;2021(1):5565648. doi:10.1155/2021/5565648.
8. Robinson JT, Devarakonda MV. Data cache management using frequency-based replacement. In: *Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: ACM; 1990. p. 134–42. doi:10.1145/98457.98523.
9. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*; 2016 Jul 19–24; New York, NY, USA. p. 1928–37.
10. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529–33. doi:10.1038/nature14236.
11. Niknia F, Wang P, Wang Z, Agarwal A, Rezaei AS. Attention-enhanced prioritized proximal policy optimization for adaptive edge caching. *IEEE Trans Veh Technol*. 2025;74(8):12853–63. doi:10.1109/TVT.2025.3552084.
12. Dabney W, Rowland M, Bellemare M, Munos R. Distributional reinforcement learning with quantile regression. In: *Proceedings of the AAAI Conference on Artificial Intelligence*; 2018 Feb 2–7; New Orleans, LA, USA. p. 2892–901. doi:10.1609/aaai.v32i1.11791.
13. Zhao M, Nakhai MR. A unified federated deep Q learning caching scheme for scalable collaborative edge networks. *IEEE Trans Mob Comput*. 2024;23(12):10855–66. doi:10.1109/TMC.2024.3382824.
14. Abolhassani B, Tadrous J, Eryilmaz A. Optimal load-splitting and distributed-caching for dynamic content over the wireless edge. *IEEE/ACM Trans Netw*. 2023;31(5):2178–90. doi:10.1109/TNET.2023.3244039.
15. Zhu H, Cao Y, Wang W, Jiang T, Jin S. Deep reinforcement learning for mobile edge caching: review, new features, and open issues. *IEEE Netw*. 2018;32(6):50–7. doi:10.1109/MNET.2018.1800109.
16. Meybodi ZH, Mohammadi A, Hou M, Rahimian E, Heidarian S, Abouei J, et al. Multi-content time-series popularity prediction with multiple-model transformers in MEC networks. *Ad Hoc Netw*. 2024;157(6):103436. doi:10.1016/j.adhoc.2024.103436.
17. Kang WC, McAuley J. Self-attentive sequential recommendation. In: *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*; 2018 Nov 17–20; Singapore. p. 197–206. doi:10.1109/ICDM.2018.00035.
18. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P. Trust region policy optimization. In: *Proceedings of the 32nd International Conference on Machine Learning*; 2015 Jul 6–11; Lille, France. p. 1889–97.
19. Schulman J, Moritz P, Levine S, Jordan MI, Abbeel P. High-dimensional continuous control using generalized advantage estimation. In: *Proceedings of the International Conference on Learning Representations (ICLR)*; 2016 May 2–4; San Juan, Puerto Rico.
20. Chadi MA, Mousannif H. Understanding reinforcement learning algorithms: the progress from basic Q-learning to proximal policy optimization. *arXiv:2304.00026*. 2023.
21. Elsayed M, Farrahi H, Dangel F, Mahmood AR. Revisiting scalable hessian diagonal approximations for applications in reinforcement learning. In: *Proceedings of the 41st International Conference on Machine Learning*; 2024 Jul 21–27; Vienna, Austria. p. 12448–68. doi:10.5555/3692070.3692566.
22. Gao T, Chen B, Mi Q. A survey of Markov model in reinforcement learning. In: *Proceedings of the 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIC)*; 2022 Feb 21–24; Jeju Island, Republic of Korea. p. 284–7.
23. Yuan G, Yuan F, Li Y, Kong B, Li S, Chen L, et al. Tenrec: a large-scale multipurpose benchmark dataset for recommender systems. In: *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*; 2022 Nov 28–Dec 9; New Orleans, LA, USA. p. 11480–93. doi:10.52202/068431-0834.
24. Xu P, Zhu X, Clifton DA. Multimodal learning with transformers: a survey. *IEEE Trans Pattern Anal Mach Intell*. 2023;45(10):12113–32. doi:10.1109/TPAMI.2023.3275156.
25. Jia N, Qu Z, Ye B, Wang Y, Hu S, Guo S. A comprehensive survey on communication-efficient federated learning in mobile edge environments. *IEEE Commun Surv Tutor*. 2025;27(6):3710–41. doi:10.1109/comst.2025.3535957.