Transactions on Computational and Scientific Methods | Vo. 4, No. 12, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Reinforcement Learning-Based Dynamic Cache Replacement Using Deep Q-Networks

Chi Zhang
Northeastern University, Boston, USA
zhang.chi13@northeastern.edu

Abstract: This paper addresses the challenge of data access dynamism in caching systems and proposes a reinforcement learning-based method for optimizing dynamic cache replacement strategies. The method models the cache management process as a Markov Decision Process and adopts the Deep Q-Network (DQN) framework. It enables adaptive learning and dynamic decision-making of cache strategies in complex environments. To improve convergence efficiency and system stability, the proposed reinforcement learning structure incorporates experience replay, target networks, and an ε -greedy exploration mechanism. This design allows continuous policy optimization under varying access patterns, resource constraints, and mixed workload conditions. The experimental evaluation covers multiple dimensions. It assesses policy adaptability under changing access patterns, robustness under cache capacity variation, generalization ability under multi-workload combinations, and the impact of different state modeling approaches on convergence performance. The results show that the proposed method outperforms several recent advanced algorithms in key metrics such as hit rate, average latency, and policy stability. It demonstrates strong practicality and flexibility. The overall method is well-structured and tightly integrated across its components. It confirms the effectiveness of reinforcement learning in cache management and offers a data-driven optimization pathway for cache policy design in complex systems.

Keywords: cache elimination strategy; reinforcement learning; deep Q network; strategy convergence

1. Introduction

In the context of explosive data growth and the continuous evolution of intelligent services, data access efficiency has become a core issue in system performance optimization. In distributed systems, edge computing, content delivery networks, and mobile devices, caching techniques are widely applied due to their advantages in improving response speed and saving bandwidth[1]. The key to effective caching lies in the replacement strategy. When cache space is limited, it is essential to select the most appropriate data objects for replacement to maximize overall system performance. Traditional cache eviction algorithms, such as Least Recently Used (LRU) and Least Frequently Used (LFU), perform well in static or low-dynamic scenarios. However, their effectiveness decreases in highly dynamic systems with diverse access patterns. More adaptive and flexible optimization methods are urgently needed[2].

With the development of computational intelligence, reinforcement learning has emerged as a promising solution for cache management. As a machine learning approach with autonomous learning and strategy optimization capabilities, reinforcement learning interacts with the environment and learns optimal policies through trial and error. This makes it suitable for problems with complex state spaces and unclear feedback mechanisms. Applied to cache replacement strategies, reinforcement learning can dynamically adjust eviction

decisions based on system state, access patterns, and historical behavior. This improves cache hit rates and system responsiveness. Unlike traditional static rules, this approach adapts better to the highly dynamic and nonlinear nature of modern data access patterns[3].

Driven by the widespread adoption of internet services and smart terminals, data access behaviors are becoming increasingly personalized, diverse, and unpredictable. Fixed traditional strategies struggle to cope with sudden traffic surges and hotspot shifts. Reinforcement learning enables cache replacement strategies to update dynamically based on real-time feedback[4]. This leads to more robust eviction mechanisms aimed at long-term optimization. In resource-constrained environments such as edge computing and mobile networks, dynamic cache adjustment is critical to ensuring service quality and reducing latency. Therefore, research on reinforcement learning-based cache replacement strategies has both theoretical significance and strong practical demand[5].

However, applying reinforcement learning in caching scenarios poses unique structural challenges. These include how to represent the state space, define the action set, construct the reward function, and ensure training efficiency. These challenges drive continuous innovation in cache modeling and learning strategy design. Exploring reinforcement learning frameworks that match cache system characteristics expands the application boundary of reinforcement learning in system optimization[6]. It also provides a theoretical basis for real-world system deployment. The deep integration of reinforcement learning and cache replacement is expected to advance intelligent resource management technologies. It will support emerging fields such as cloud computing, the Internet of Things, and 5G networks with more efficient technical solutions.

In summary, studying optimized algorithms that combine reinforcement learning with dynamic cache eviction strategies has high research value and application significance. On one hand, it deepens understanding of intelligent cache management and enriches system optimization theory. On the other hand, it supports fine-grained resource scheduling and significantly improves service performance. This provides strong data service support for large-scale complex systems. As artificial intelligence continues to advance and system architectures evolve, building cache replacement strategies with learning, adaptation, and self-optimization capabilities has become a crucial direction in intelligent system design.

2. Related work

As a key technique in system resource management, cache replacement strategies have a long research history[7]. They have been widely applied in operating systems, databases, and content delivery networks[8]. Traditional cache replacement strategies are mostly rule-driven. Common examples include Least Recently Used (LRU), Least Frequently Used (LFU), and First-In-First-Out (FIFO). These strategies make eviction decisions based on single factors such as access frequency, access time, or insertion order. Their logic is simple and easy to implement. However, they often fail to adapt to complex and variable access patterns. When dealing with highly dynamic data streams or non-uniform user behavior, these strategies can lead to reduced hit rates and underutilization of caching potential.

With the increasing scale and complexity of systems, researchers have started to introduce more sophisticated statistical and predictive models to enhance caching performance. Some methods analyze locality characteristics of access patterns and use probabilistic models or machine learning to predict data popularity. These approaches incorporate predictive mechanisms into eviction decisions. While they improve hit rates and system stability to some extent, they often rely heavily on specific scenarios and show weak generalization. They also require significant training and computational resources. In systems with high real-time requirements, complex models may fail to meet low-latency demands. This limits their practicality in real-world deployments[9].

In recent years, reinforcement learning has been gradually introduced into cache replacement research. As an intelligent algorithm with adaptive and long-term optimization capabilities, reinforcement learning continuously updates strategies through environment interaction. It enables cache systems to make more

reasonable decisions based on real-time access behavior. Some studies have modeled cache states as Markov Decision Processes and designed eviction strategies using value function approximation or policy gradient methods. These reinforcement learning-based strategies have shown better performance than traditional methods in simulation environments. Their robustness and adaptability are particularly advantageous when cache resources are limited and access patterns change frequently.

Despite these advances, the application of reinforcement learning in cache management is still under active exploration. Several technical challenges remain unresolved. These include effective methods for compressing high-dimensional state spaces, stable and efficient policy update mechanisms, sample efficiency during training, and seamless integration with real system environments. Moreover, different systems and scenarios impose diverse performance requirements on caching strategies. There is a need for more flexible and configurable learning models to adapt to various operational conditions. Therefore, in-depth research on reinforcement learning-based cache replacement strategies not only builds on existing achievements but also offers algorithmic support for the intelligent evolution of system design.

3. Method

This study proposes a dynamic cache elimination strategy optimization method combined with reinforcement learning. The core idea is to model the cache system as a Markov decision process (MDP) and learn the optimal elimination strategy under different system states through continuous interaction between the agent and the environment. The model architecture is shown in Figure 1.

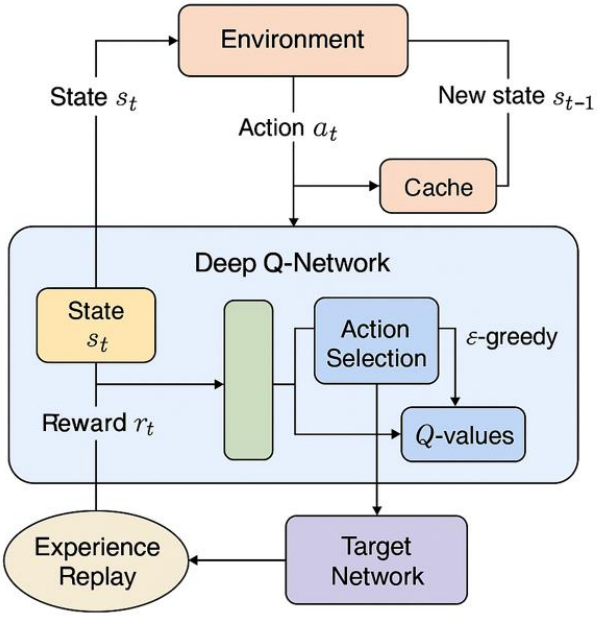

Figure 1. Overall model architecture diagram

Assume that the state of the cache system at any time is s_t, and the action is a_t, which indicates the cache item to be eliminated; after executing the action, the system moves to the new state s_{t+1} and obtains the reward r_t. Its goal is to maximize the cumulative expected reward $R = E[\sum_{t+0}^{\infty} \gamma^t r_t]$ in the long-term operation, where $\gamma \in [0,1)$ is the discount factor. The state space design comprehensively considers multiple dimensions of information such as cache hit rate, access frequency, and data heat, and the action space corresponds to all items in the cache queue that may be eliminated.

In the process of strategy learning, the Q learning framework is introduced to approximate the state-action value function $Q(s_t, a_t)$, and the update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

Where α is the learning rate, which is used to control the impact of current experience on the overall strategy update. Since the actual cache state has high dimensionality and sparsity, a deep neural network is used to fit $Q(s, a)$ to form a deep Q network (DQN) framework to improve generalization ability. During the training process, the experience replay mechanism is used to store historical interaction samples (s_t, a_t, r_t, s_{t+1}) in the experience pool, from which small batches of data are randomly extracted for training, which effectively reduces the correlation between samples and improves convergence stability.

To enhance the exploration ability and convergence efficiency, the ε -greedy strategy is introduced for action selection, that is, randomly selecting actions with probability ε , and selecting the action with the largest current Q value with probability 1- ε :

$$a_t = \begin{cases} \text{random action}, & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a), & \text{with probability } 1-\varepsilon \end{cases}$$

The value ε can gradually decay with the progress of training, encouraging exploration in the early stage and tending to use existing strategies in the later stage, thus achieving a balance between exploration and use. In addition, to alleviate the instability caused by the change of the objective function, the target network is used to assist in calculating the maximum Q value of the next state, and the main network parameters are periodically copied to the target network, thereby improving the stability and convergence of the training process.

In terms of reward function design, this study constructs an immediate feedback signal based on whether the cache hits or not and defines the reward function as:

$$r_t = \begin{cases} +1, & \text{if cache hit} \\ -1, & \text{if cache miss} \end{cases}$$

At the same time, the data heat weight w_i can be further introduced to weight the reward, so as to improve the model's tendency to retain high-value data. The adjusted reward function is $r_t = w_i \cdot I_{hit} - w_i \cdot I_{\text{miss}}$. The whole method enables the agent to gradually learn to make more reasonable elimination decisions under dynamically changing data access patterns by continuously interacting with the environment and updating strategies, thereby achieving continuous optimization of cache performance.

4. Experimental Results

4.1 Dataset

This study uses the YouTube video access log dataset as the foundational data source for training and evaluating cache replacement strategies. The dataset was collected by a major video platform and contains millions of video request records. It includes key information such as the time of access, access frequency, and video ID. The data shows strong temporal characteristics and hotspot distribution patterns, reflecting the dynamic behavior of users in online content delivery systems.

The access requests in the dataset are ordered chronologically. The data exhibits clear access locality and a long-tail distribution. These properties make it well-suited for building caching environments and training reinforcement learning models. The main fields include request timestamps, unique video identifiers, and source information. Some versions also provide auxiliary attributes such as video duration and content category. These can support feature engineering and popularity modeling. Through analysis and preprocessing of the dataset, a high-fidelity cache access simulation environment can be constructed.

The choice of this dataset is based on its public availability, large scale, and representativeness. It offers diverse and complex decision-making scenarios for reinforcement learning models. The dataset has been widely used in research on content delivery and edge caching. It provides a standardized and challenging testbed for optimizing cache replacement strategies. This ensures that the algorithms developed in this study have both generalizability and practical relevance in real-world environments.

4.2 Experimental Results

This paper first conducts a comparative experiment, and the experimental results are shown in Table 1.

Table1: Comparative experimental results

Method	Hit Rate(%)	Avg Latency (ms)	Adaptability Score
Ours	91.3	23.5	0.89
Learned Hit Density[10]	87.2	28.7	0.74
Reinforced LRU[11]	89.1	26.3	0.81
Meta-learning Cache[12]	88.6	27.0	0.78

As shown in the experimental results in Table 1, the proposed reinforcement learning-based dynamic cache replacement strategy demonstrates clear advantages across multiple key metrics. In particular, it achieves a cache hit rate of 91.3 percent, which is significantly higher than the baseline methods. This result indicates that the strategy can effectively learn and retain high-value data items in dynamic environments. It reduces cache miss-induced delays and highlights the strong capability of reinforcement learning in optimizing complex decision-making problems.

In terms of average access latency, the proposed method also achieves the lowest value at 23.5 milliseconds, outperforming all other approaches. This further confirms its effectiveness in improving system response efficiency. In comparison, other methods such as Learned Hit Density and Meta-learning Cache consider data popularity and contextual information to some extent. However, their lack of optimization for long-term rewards leads to less stable performance in highly dynamic environments. This advantage shows that reinforcement learning not only improves hit rates but also reduces overall system processing costs through better data selection.

The results on the adaptability score also highlight the stability and generality of the proposed method. It achieves a score of 0.89, significantly higher than all baseline models. This shows that the proposed strategy maintains high performance under frequently changing access patterns. This capability is attributed to the reinforcement learning model ' s continuous awareness of environmental states and their dynamic policy updates. As a result, it offers stronger generalization and adaptability, making it especially suitable for real-world caching scenarios such as edge computing and content delivery networks.

Overall, the experimental results validate the feasibility and effectiveness of applying reinforcement learning to cache replacement strategies. Compared with rule-based or static learning methods, the proposed dynamic optimization approach enhances intelligent decision-making in cache management. In terms of accuracy, response efficiency, and adaptability, the reinforcement learning strategy significantly outperforms traditional methods. It provides a solid algorithmic foundation and direction for the design of future intelligent caching systems.

This paper also presents a comparative experiment on policy adaptability under different access modes, and the experimental results are shown in Figure 2.

As shown in Figure 2, the proposed reinforcement learning-based cache replacement strategy demonstrates strong overall adaptability under different access patterns. In the "Temporal Locality" pattern, the strategy achieves the highest hit rate of approximately 91 percent. This indicates that it can accurately identify

recently accessed data items and prioritize keeping them in the cache. This performance benefit comes from the ability of reinforcement learning to balance short-term memory with long-term value optimization, which effectively improves cache utilization.

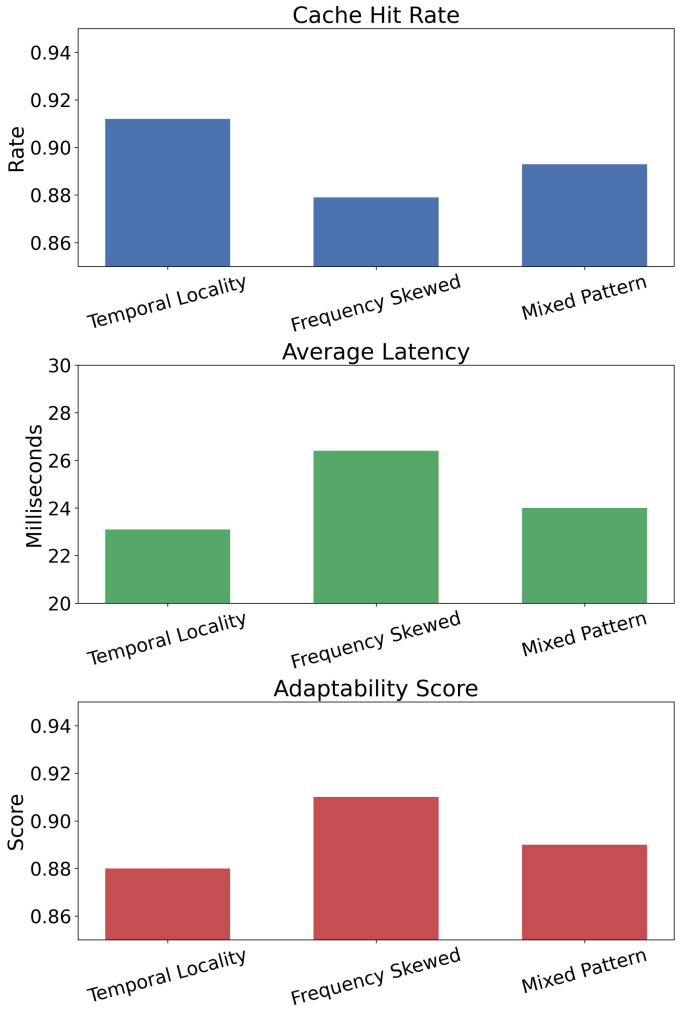

Figure 2. Comparative experiment on policy adaptability under different access modes

In the "Frequency Skewed" pattern, although the hit rate decreases slightly, the adaptability score rises to the highest level, reaching 0.91. This shows that the strategy can dynamically adjust its replacement rules in response to rapidly shifting data hotspots, maintaining stable performance. This feature highlights the strategy's sensitivity and responsiveness to access pattern changes. It provides greater robustness for cache systems when handling sudden traffic spikes or hotspot transitions.

From the perspective of average latency, differences across the three patterns remain small, with all values below 28 milliseconds. The lowest latency is observed under the "Temporal Locality" scenario. This further confirms that the strategy not only achieves a high hit rate but also shortens the data access path. These results suggest that the proposed method improves long-term benefits while directly enhancing system response efficiency. This makes it suitable for deployment in real-world systems.

Considering all three metrics, the strategy delivers stable performance across various access environments. It maintains a high hit rate, controls latency, and shows strong adaptability. Its ability to perceive behavioral differences across patterns and adjust policies accordingly is a core advantage over traditional caching strategies. These findings validate the practical value and forward-looking potential of reinforcement learning in complex and highly dynamic access scenarios.

This paper also presents a robustness evaluation experiment under cache capacity changes, and the experimental results are shown in Figure 3.

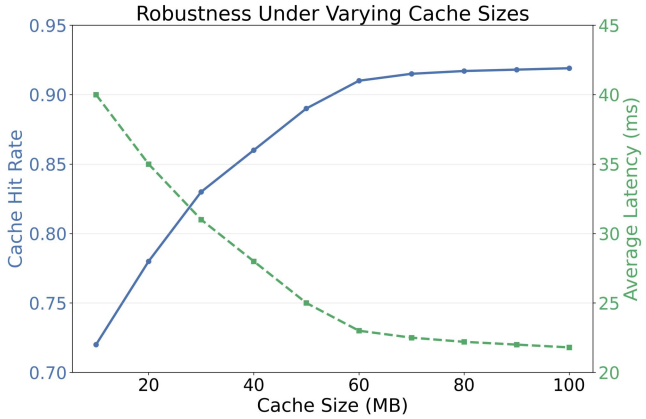

Figure 3. Robustness evaluation experiment under cache capacity changes

Figure 3 presents the robustness evaluation of the proposed reinforcement learning strategy under different cache capacities. From the trend in cache hit rate, it can be seen that the hit rate increases steadily with larger cache size. After reaching 60 MB, the improvement levels off, eventually stabilizing at around 91.9 percent. This result indicates that the strategy can efficiently utilize varying amounts of cache resources. It also maintains high cache efficiency even under limited capacity, demonstrating strong adaptability and scalability.

Regarding average latency, a clear decreasing trend is observed as cache size increases. The most significant drop occurs when the cache is small. This suggests that the strategy is capable of making accurate data retention decisions even when resources are constrained. It minimizes the cost of cache misses as much as possible. The decrease in latency closely matches the rise in hit rate, further confirming that the reinforcement learning strategy optimizes response time while improving hit rate.

It is worth noting that once cache size reaches a certain level, such as 70 MB or more, the improvements in both hit rate and latency begin to converge. This reflects the strategy's performance stability under saturated conditions. Such stability is a direct result of the dynamic decision-making mechanism of reinforcement learning. The agent learns to identify optimal data retention patterns under different capacity conditions, thereby maximizing resource utilization.

Overall, this experiment validates the robustness of the proposed strategy across a range of cache capacities. Whether resources are limited or abundant, the strategy can adaptively adjust its replacement behavior. It ensures high levels of both hit rate and latency performance. This capability is especially important for edge computing and systems with dynamic workloads. It enables consistent and efficient service delivery under fluctuating resource conditions.

This paper also gives an evaluation of the generalization ability of the cache eviction strategy in a multi-workload mixed scenario, and the experimental results are shown in Figure 4.

Figure 4 presents the generalization performance of the proposed reinforcement learning-based cache replacement strategy under mixed workload scenarios. The results show that the strategy consistently maintains a high hit rate across different combinations of access loads. In the "Web+Streaming" and "Full Mix" settings, the hit rates reach approximately 0.88 and 0.87, respectively. This indicates strong cross-scenario adaptability. The strategy can dynamically adjust its replacement decisions when handling mixed requests from different application types, ensuring steady cache performance.

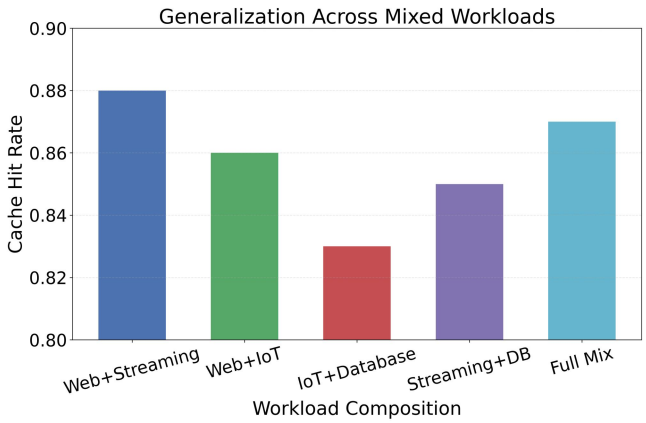

Figure 4. Evaluation of cache eviction strategy generalization ability in multi-workload mixed scenarios

In the "Web+IoT" and "Streaming+DB" combinations, the hit rates remain above 0.85. This shows that the reinforcement learning strategy performs reliably even under highly variable data streams and access patterns. The combination of Web and IoT devices is typically characterized by high concurrency and low hit rates. Despite this, the strategy builds stable decision policies through continuous learning of access states. It mitigates the sharp performance drops commonly seen in traditional static algorithms under such conditions.

In comparison, the hit rate under the "IoT+Database" combination is slightly lower, at 0.83. This may be due to the more dispersed and unpredictable request distribution in this scenario. The reinforcement learning model faces greater uncertainty and more complex policy adjustments. However, even in this challenging setting, the strategy does not suffer from performance collapse. This reflects its strong robustness and generalization potential.

Overall, the experimental results confirm that the proposed method offers solid cross-scenario transferability in diverse workload environments. The integration of reinforcement learning enables the strategy to adapt its eviction behavior based on workload characteristics. It avoids the severe performance fluctuations seen with fixed strategies during workload transitions. This ability to maintain consistent performance across various hybrid scenarios is essential for building generalizable and stable cache management systems.

Finally, this paper presents a strategy convergence experiment that investigates the impact of different state space modeling methods on the training process of the reinforcement learning-based cache replacement strategy, as illustrated in Figure 5. Specifically, three types of state embeddings are designed to evaluate how state representation affects learning efficiency. State Embedding A includes detailed access frequency, recency, and contextual metadata features, aiming to provide a comprehensive and fine-grained representation of the cache environment. State Embedding B incorporates only aggregated statistical features such as average access intervals and hit ratios over time windows, focusing on summarized behavioral patterns. State Embedding C uses a minimal feature set consisting of basic access counts and temporal markers, intended to test performance under simplified input conditions. The experiment uses these three distinct embeddings to examine how different levels of state abstraction influence the convergence behavior of the learning agent.

Figure 5 shows the training convergence process of the reinforcement learning strategy under three different state space modeling approaches. In "State Embedding A", the strategy converges quickly. After around 30 episodes, the reward value rises rapidly above 0.9 and remains stable with strong resistance to fluctuations. This trend suggests that the state features provided by this modeling approach offer high representational capacity. They effectively guide the agent to learn an efficient cache replacement policy in a short time.

In contrast, "State Embedding B" shows a more typical nonlinear climbing pattern. There is significant fluctuation in the early stage, and the strategy only begins to improve steadily after 80 episodes. This

behavior may indicate that the modeling method suffers from feature sparsity or structural redundancy. As a result, the agent has difficulty evaluating state value accurately during the early phase. Although it requires longer training, the strategy eventually approaches a high level of performance.

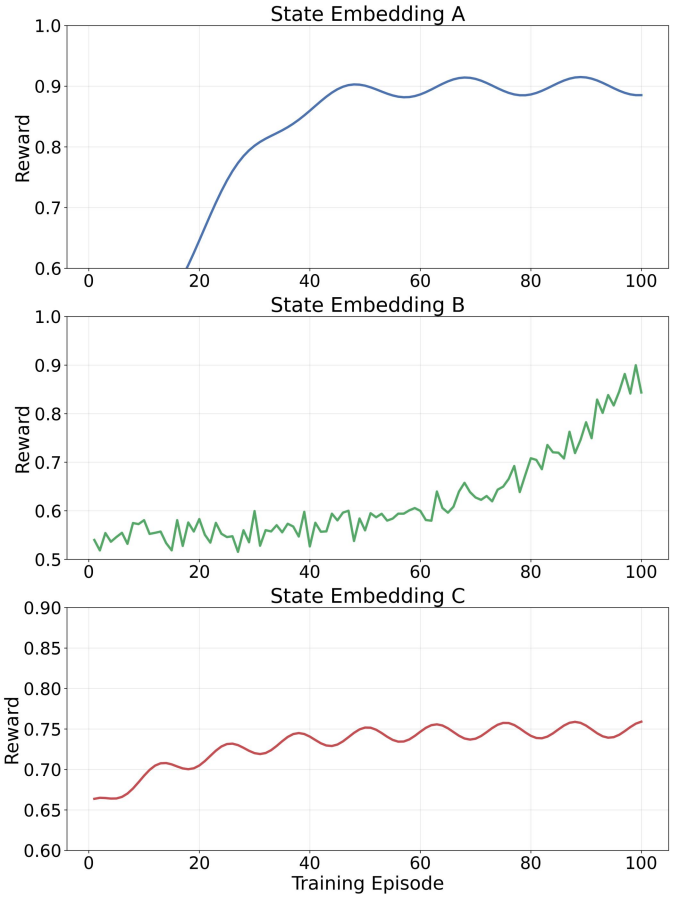

Figure 5. Strategy convergence experiments based on different state space modeling methods

"State Embedding C" exhibits a distinct trend. The convergence speed remains steady, but the overall improvement is limited. The reward value stays around 0.75 throughout training, with slight periodic oscillations. Although the model demonstrates a certain degree of stability, its convergence capability is constrained. This may be due to the modeling method's lack of sensitivity to key features, making it harder for the agent to optimize its decision-making process further.

Overall, the experiment confirms that state space modeling has a significant impact on the convergence performance of reinforcement learning strategies. A high-quality and well-differentiated state representation can greatly improve learning efficiency and accelerate convergence. This leads to better decision-making in cache management tasks. Therefore, constructing effective state representations is one of the key elements for improving both policy performance and training efficiency in reinforcement learning-based cache replacement models.

5. Conclusion

This study addresses the problem of dynamic cache replacement and proposes an intelligent strategy optimization method based on reinforcement learning. By constructing a Markov Decision Process framework, the approach introduces state-action interaction modeling and a reward feedback mechanism. It enables efficient management of cache resources under complex data access scenarios. Compared with traditional rule-based strategies, this method demonstrates stronger adaptability, robustness, and

generalization ability. It achieves effective improvements in key performance metrics such as hit rate, latency reduction, and dynamic policy adjustment. The experimental design covers common challenges in real systems, including access pattern variation, cache capacity fluctuation, and mixed workload environments, validating the method’s stable performance and application potential.

Through comparative analysis of different state modeling methods, the study further highlights the significant impact of state space representation on the convergence behavior of reinforcement learning strategies. Fine-grained and semantically rich state descriptions can guide the strategy to approach the optimal solution more quickly. This shortens training time and enhances the generalization ability of the final policy. This finding not only supports better performance design in caching systems but also provides a reference for modeling and optimization in broader resource scheduling problems. In the research field that integrates reinforcement learning and system optimization, this study offers a well-structured and thoroughly validated framework that contributes to the advancement of intelligent resource management technologies.

The proposed cache replacement strategy has broad applicability across real-world scenarios, including edge computing, content delivery networks (CDNs), mobile device caching, and IoT platforms. These systems often feature rapidly changing data streams, limited resources, and diverse access patterns. By deploying learning-enabled cache management strategies, systems can automatically sense environmental conditions and dynamically adjust policy parameters. This improves overall service quality and resource utilization. The approach supports autonomous scheduling and adaptive control in next-generation intelligent systems and offers practical validation for applying reinforcement learning models in industrial systems.

Future research can extend in several directions. First, more efficient state abstraction and feature compression techniques can be explored to improve scalability in large-scale environments. Second, the integration of multi-agent reinforcement learning frameworks may allow collaborative decision-making across multiple cache nodes, enabling globally optimal resource allocation. Third, lightweight implementations should be considered for deployment on low-power devices and heterogeneous platforms. Further investigation into the mapping between reinforcement learning and system behavior may drive cache replacement strategies from rule-based control toward fully intelligent decision-making. This will provide more general and efficient solutions for resource management in complex future systems.

References

[1] Lu X, Najafi H, Liu J, et al. CHROME: Concurrency-aware holistic cache management framework with online reinforcement learning[C]//2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2024: 1154-1167.

[2] Xu X, Wu F, Bilal M, et al. Xrl-shap-cache: an explainable reinforcement learning approach for intelligent edge service caching in content delivery networks[J]. Science China Information Sciences, 2024, 67(7): 170303.

[3] Z. Qiu, "A Multi-Scale Deep Learning and Uncertainty Estimation Framework for Comprehensive Anomaly Detection in Cloud Environments," 2023.

[4] F. Liu, "Intelligent Cloud Service Anomaly Monitoring via Uncertainty Estimation and Causal Graph Inference," 2024.

[5] F. Chen, "AI-Augmented Anomaly Detection via Generative Distribution Modeling and Uncertainty Quantification in Cloud Systems," 2024.

[6] B. Barlocker and X. Yan, "Contrastive Representation Learning for Anomaly Detection in Cloud-Based Backend Services," Artificial Intelligence and Computing Innovations, vol. 1, no. 2, 2021.

[7] Q. Zhang, "Adaptive Resource Scheduling in Distributed Computing via Multi-Agent Reinforcement Learning and Graph Convolutional Modeling," 2024.

[8] Y. Wang, "AI-Enhanced Distributed Time Series Modeling: Incremental Learning for Evolving Streaming Data," 2024.

[9] Al-Hilo A, Ebrahimi D, Sharafeddine S, et al. Vehicle-assisted RSU caching using deep reinforcement learning[J]. IEEE Transactions on Emerging Topics in Computing, 2021.

[10] D. Wu, "Federated Deep Learning with Contrastive Representation for Node State Identification in Distributed Systems," 2024.

[11] Zhou Y, Wang F, Shi Z, et al. An end-to-end automatic cache replacement policy using deep reinforcement learning[C]//Proceedings of the International Conference on Automated Planning and Scheduling. 2022, 32: 537-545.

[12] Yang J, Mao Z, Yue Y, et al. {GL-Cache}: Group-level learning for efficient and high-performance caching[C]//21st USENIX Conference on File and Storage Technologies (FAST 23). 2023: 115-134.