electronics

MDPI

Article

Enhancing Heterogeneous Network Performance: Advanced Content Popularity Prediction and Efficient Caching

Zhiyao Sun and Guifen Chen *

School of Electronic and Information Engineering, Changchun University of Science and Technology, Changchun 130022, China; sunzhiyao@mails.cust.edu.cn
* Correspondence: chengf0213@163.com

Abstract: With the popularity of smart devices and the growth of high-bandwidth applications, the wireless industry is facing an increased surge in data traffic. This challenge highlights the limitations of traditional edge-caching solutions, especially in terms of content-caching effectiveness and network-communication latency. To address this problem, we investigated efficient caching strategies in heterogeneous network environments. The caching decision process becomes more complex due to the heterogeneity of the network environment, as well as due to the diversity of user behaviors and content requests. To address the problem of increased system latency due to the dynamically changing nature of content popularity and limited cache capacity, we propose a novel content placement strategy, the long-short-term-memory–content-population-prediction model, to capture the correlation of request patterns between different contents and the periodicity in the time domain, in order to improve the accuracy of the prediction of content popularity. Then, to address the heterogeneity of heterogeneous network environments, we propose an efficient content delivery strategy: the multi-intelligent critical collaborative caching policy. This strategy models the edge-caching problem in heterogeneous scenarios as a Markov decision process using multi-base-station-environment information. In order to fully utilize the multi-intelligence information, we have improved the actor–critic approach by integrating the attention mechanism into a neural network. Whereas the actor network is responsible for making decisions based on local information, the critic network evaluates and enhances the actor's performance. We conducted extensive simulations, and the results showed that the Long Short Term Memory content population prediction model was more advantageous, in terms of content-popularity-prediction accuracy, with a 28.61% improvement in prediction error, compared to several other existing methods. The proposed multi-intelligence actor–critic collaborative caching policy algorithm improved the cache-hit-rate metric by up to 32.3% and reduced the system latency by 1.6%, demonstrating the feasibility and effectiveness of the algorithm.

Keywords: heterogeneous network; edge caching; content popularity prediction; content placement strategy; content delivery strategy; long and short-term memory networks; multi-intelligent actor–critic

Citation: Sun, Z.; Chen, G. Enhancing Heterogeneous Network Performance: Advanced Content Popularity Prediction and Efficient Caching. *Electronics* **2024**, *13*, 794. https://doi.org/10.3390/electronics13040794

Academic Editor: Dimitris Kanellopoulos

Received: 20 December 2023
Revised: 15 February 2024
Accepted: 16 February 2024
Published: 18 February 2024

1. Introduction

The proliferation of smart devices and the growth of high-bandwidth applications have led to an unprecedented surge in data traffic, thus imposing significant pressure on mobile network operators (MNOs) [1]. Mobile edge caching, as a pioneering technology, is widely regarded as an effective means through which to alleviate the traffic burden on MNOs [2]. By storing content at network edge nodes, Mobile Edge Computing(MEC) can help with significantly reducing data transmission latency, thereby enhancing the quality of experience (QoE) and quality of service (QoS) for users. However, traditional edge-caching strategies are increasingly revealing their limitations in addressing the growing diversity of content and changes in user behavior [3].

The emergence of heterogeneous networks (HetNets) provides a new perspective for tackling these challenges [4]. A core feature of HetNets is the diversity of base-station

types, including micro, macro, and small cells, which differ significantly in caching capacity, coverage area, and service modes. The diversity of user behavior and the heterogeneity of content requests further complicate the caching decision-making process, thereby intensifying the demand for more sophisticated and intelligent caching strategies.

To effectively reduce the load on heterogeneous networks, caching popular content is essential. However, given the limited caching resources at the edge, it is crucial to analyze the popularity of content in the library. The key challenge in caching-placement strategy lies in accurately predicting future content popularity, which is complicated by the high dynamics of user content requests and the diversity of user behavior. Machine learning (ML) techniques have been adopted as powerful tools for actively predicting content popularity based on historical request data [5]. Despite this, learning-based edge-caching strategies still face several unresolved challenges, including considerations of content correlations, optimizations of resource consumption in heterogeneous networks, and, particularly, the development of collaborative caching strategies that involve multiple base stations.

For the complex load and content delivery strategies in heterogeneous edge networks, traditional caching-replacement strategies are limited by their neglect of dynamic interactions with the environment. With the advancement of multi-agent reinforcement-learning models, researchers are exploring the use of these models to formulate adaptive edge-caching strategies, thus fully leveraging the environmental-awareness capabilities of network systems [6]. We propose a multi-agent actor–critic framework that utilizes multi-head attention technology to integrate features from multiple base stations, thereby addressing the collaborative caching issues that are present in heterogeneous networks with mixed base-station cooperation [7]. This framework aims to make effective caching decisions and performance evaluations by considering both the local base station's caching capacity and the global information from neighboring base stations.

The main contributions of this paper include the following:

- Establishing an efficient caching strategy that minimizes communication latency and maximizes cache hit rates by addressing the challenges of dynamic content popularity and resource allocation in heterogeneous networks.
- Introducing an Long Short Term Memory(LSTM)-based content-popularity-prediction model for a caching placement strategy, using low-dimensional denoising encoders to capture content-request information and LSTM modules to analyze temporal features. This is coupled with a fully connected network for feature fusion, thereby enhancing prediction accuracy and effectively addressing the diversity of user requests in heterogeneous network environments.
- Representing the heterogeneous-edge-content-delivery problem as a Markov decision process (MDP) and providing a collaborative caching solution based on multi-agent actor–critic networks. This approach utilizes multi-head attention technology to account for the variability in a diverse network environment.
- Conducting a series of tests to validate the effectiveness of the proposed solution in improving network performance and reducing resource inefficiency, thus demonstrating significant improvements in content-popularity-prediction capability, cache hit rate, and system latency reduction compared to existing solutions and other reinforcement-learning methods.

2. Related Works

With the proliferation of intelligent devices and the growth of high-bandwidth applications, heterogeneous networks have become an effective solution for addressing the increased surge in data traffic [8,9]. Heterogeneous networks consist of different types of base stations, including micro base stations, macro base stations, and small-cell base stations, each differing in cache capacity, coverage area, and service modes [10].

This diversity allows heterogeneous networks to be deployed flexibly according to different scenarios. For example, in densely populated urban centers, small-cell base

stations or micro base stations can be deployed to provide high-density network coverage; in vast suburban or rural areas, macro base stations are more suitable, due to their wide coverage range, and can effectively provide basic communication services to cover larger geographical areas. For indoor environments, such as office buildings and shopping centers, more stable and high-speed network connectivity can be provided by deploying small cellular base stations or micro base stations to meet the communication needs in dense environments. In addition, by deploying edge-computing nodes in the seismic monitoring network, immediate collection and processing of seismic data can be realized. Edge nodes can perform preliminary analysis of collected seismic-waveform data, such as magnitude estimation and epicenter localization, and then quickly transmit critical information to the central processing system or directly trigger the warning mechanism [11]. In all these scenarios, edge caching can improve overall network performance and reduce communication delay. Furthermore, mobile edge computing (MEC) improves quality of experience (QoE) and quality of service (QoS) by storing content in network edge nodes, reducing the distance and time of data transmission. However, edge-caching strategies face numerous challenges when dealing with increasingly diverse content and varying user behaviors [12]. For instance, limited caching resources must be effectively utilized to store the most popular content while also considering how to rapidly respond to changes in user behavior.

In a heterogeneous network environment, accurately predicting content popularity is crucial for efficient cache placement. Traditional content-popularity-prediction methods primarily rely on historical access data and simple statistical models, such as moving averages or exponential smoothing techniques [13]. However, these methods have limitations in dealing with dynamically changing content popularity. Certain studies [14–17] have adopted reinforcement learning (RL) techniques based on insights into content popularity to proactively manage cache decisions, thereby demonstrating a shift toward dynamic, adaptive caching strategies; however, these can still fail to capture complex data patterns.

Given the dynamic nature of content popularity, numerous studies [18–21] have employed various time-series-analysis methods (such as ARMA, ESN, segmented linear regression, Bi-LSTM, etc.) to capture and predict fluctuations in content demand over time. Mannepalli et al. introduced a model for predicting the popularity of social-media material. The authors combined textual content and user and time-series encoders with user-sentiment analysis. The model uses a long short-term memory network with weights that are fine tuned by adaptive rain optimization (SA-RO) to improve prediction accuracy [22]. HU et al. proposed a collaborative caching framework based on content-popularity prediction for multi-objective optimization for cloud-edge-end collaborative IoT networks, where there are difficulties in optimizing multiple network metrics at the same time, by integrating three prediction algorithms for predicting content popularity, and using a multi-objective evolutionary algorithm to mine user and content preferences and popularity characteristics [23]. However, these models still have limitations in capturing correlations and long-term dependencies among different types of content.

Most of the aforementioned studies have proposed proactive caching systems based on content popularity—a method that often overlooks user variability and the diversity of network structures. To address this issue, current research has predominantly employed artificial intelligence (AI) methods, particularly deep learning (DL) and deep reinforcement learning (DRL), in tackling cache-replacement problems. Deep learning is utilized to precisely investigate user preferences and predict base-station request patterns [24]. Deep reinforcement learning (DRL) is applied to optimize caching strategies, thereby enabling network entities to learn, build knowledge, and interact with the environment [25]. Due to the presence of macro base stations and multiple micro base stations in heterogeneous networks, which results in a complex structure with numerous nodes but limited resources, researchers have employed multi-agent reinforcement-learning approaches for cache decision making, to improve resource utilization, reduce content transmission delays, and better facilitate multi-base-station collaborative operation [26]. The studies of [27,28] explored the

application of multi-intelligent-body learning in network resource allocation and caching strategies. Multi-agent reinforcement-learning networks are overly complex and have a singular focus on feature analysis, which can lead to a wastage of computational resources and inadequacies in addressing user diversity and adaptability. The multi-head attention mechanism enables a model to obtain more layered information about base stations from different representation spaces, thereby enhancing the model's feature-representation capability. To better extract the underlying information of the system, this paper introduces an attention-mechanism approach, to focus on analyzing multi-base-station environmental information for global decision making.

Although current research has provided various innovative solutions for edge caching, effectively integrating multi-node information in heterogeneous network environments, responding to network dynamics in real time and balancing the contradiction between caching performance and resource consumption remain unresolved. These issues have still not been adequately addressed. Therefore, the objective of this study was to provide a highly effective collaborative caching method that relies on predicting the popularity of content across multiple base stations. This approach aims to address the challenges of cache resource allocation and content preferences in heterogeneous networks, thus seeking to achieve comprehensive improvements in caching efficiency and network performance.

3. System Model

3.1. Network Model

We considered a heterogeneous network architecture containing three layers with collaborative edge caching, as shown in Figure 1. The first layer was the core network-communication layer, which consisted of a macro base station (MBS). The second layer, the medium communication layer, contained a number of small base stations (SBSs) with limited caching capabilities, which were responsible for responding directly to user requests. The third layer was the mobile-edge-communication-node layer, which consisted of several mobile terminals (MTs). The set of MBS and SBSs were $\mathcal{M}$, where $M = 0$ denoted the MBS and the MTs were randomly distributed. The set of MTs for each SBS service was defined as $\mathcal{N}_m$. MT n communicated wirelessly with SBS m. For ease of reference, Table 1 summarizes the key notations.

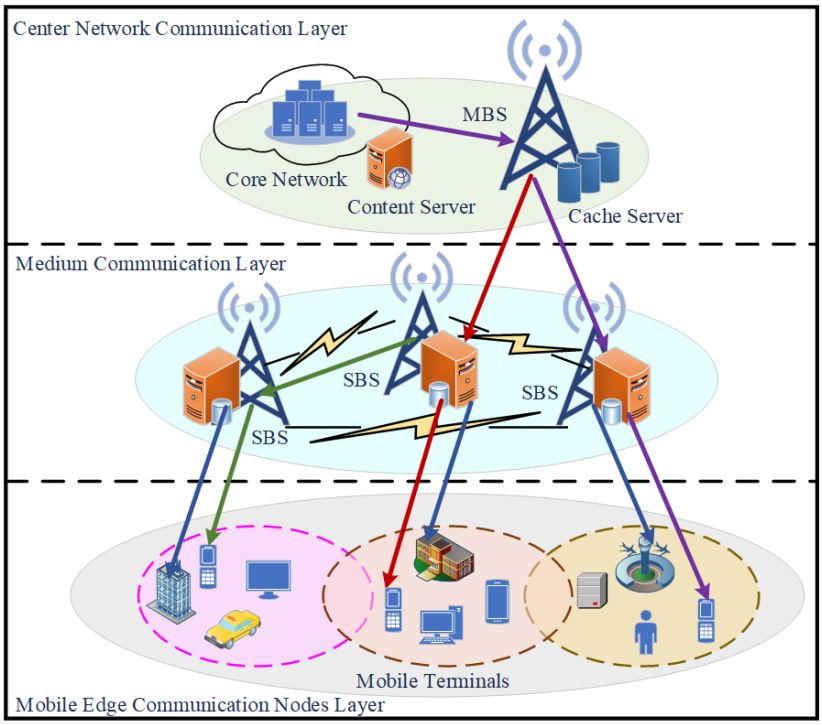


Figure 1. Three-layer heterogeneous network architecture.

Table 1. Key notations.

Symbol	Description
$\mathcal{M} = \{0, 1, ..., m, ..., M\}$	Set of MBS and SBSs
$\mathcal{N}_m = \{0, 1, ..., n, ..., N_m\}$	Set of all MTs within the coverage area of SBS m
t	Time slot t
$V_{m,n}(t)$	Data transmission rate
$G_{m,n}(t)$	Channel gain between SBS m and MT n in time slot t
P_s	Transmit power
ζ^2	Background-noise power
B_s	Bandwidth of SBSs
$x_{m,n}(t) \in (0, 1)$	Bandwidth percentage allocated by SBS m to corresponding MT n
$\mathcal{F} = \{1, 2, ..., f, ..., F\}$	Set of contents stored in the content server
$SIZE_f$	Size of each content
C_S and C_M	Cache capacity of servers equipped in SBSs and MBS
H	Historical request data of contents
τ	Length of historical-data observation window
$\mathcal{K} = \{1, 2, ..., k, ..., K\}$	Categories of content popularity for different service ranges
$\gamma_n^k \in \{0, 1\}$	Interest category of each base station
pr_f	Request probability of content f
H_f^k	Popularity ranking of content f in interest category k
ι	Reflects the skewness of the Zipf distribution
$y_{m,n}^t$	Content-caching decision of BS m in time slot t
D	Delay function
$C_m(t)$	Cache hit rate in time slot t
$c_f(t)$	Number of requests for content f in time slot t
$d_f(t)$	Number of requests for all content within the coverage area of base station m
$\mathbf{W}$	Weight matrix
$\mathbf{b}$	Bias vector
z_i^t	Input gate of the LSTM network
z_f^t	Forget gate of the LSTM network
z_o^t	Output gate of the LSTM network
C_t^{LSTM}	Cell state of the LSTM network
$\mathcal{H}_t$	Hidden-layer output variable of the LSTM network
$sim_{f,f'}$	Similarity between new content f and existing content f'
S_i	State space
a_i	Action space
$\mathbf{R}(\cdot)$	Reward function
O	Observation mechanism
$\mathcal{AF}(\cdot)$	Update rule typically involves the advantage function
$key_{d,m'}$	Key vector
$value_{d,m'}$	Value vector
Att_m	Critic network of each BS estimates the action-value function through attention mechanism
ω_k	MLP layer
μ	Decay factor
α	Balances between the maximum entrance and reward
ψ	Network-update parameter
lr	Learning rate

In addition, the edge server had the computational capability to perform offline training tasks and caching-policy tasks. The system adopted a heterogeneous network with multiple base stations for collaborative communication. The macro base station shared cache contents with the small base stations through backhaul links, and the cache contents could also be transmitted between the small base stations through cables with limited capacity.

3.2. Communication Model

The different SBSs were considered to share the same radio spectrum, and the MTs connected to the same SBS were given orthogonal spectra. As a result, the MTs experienced interference from other small cells; in addition, no interference within the small cells was taken into account.

At time slot t, the data-transmission rate of the downlink, where the SBS m sent its content to the MT n, was

$$V_{m,n}(t) = x_{m,n}(t)B_s\log_2(1 + \frac{P_s G_{m,n}(t)}{\sum_{m' \in \mathcal{M}\setminus\{0,m\}} P_s G_{m',n}(t) + \zeta^2}). \quad (1)$$

The rate depended on the channel gain $G_{m,n}(t)$ of SBS m and MT n at time slot t, the transmit power of SBS P_s, the background-noise power, and the bandwidth of SBS B_s. The variable $x_{m,n}(t) \in (0, 1)$ represented the proportion of bandwidth assigned by SBS m to the matching MT n.

The channel power gain in this model was the gain of the LOS channel, which reduced the complexity of the model and ensured the validity of the caching algorithm study. According to the LOS channel form, the power gain of the channel was $G_{m,n} = \beta_0 d^{-\alpha}\|g\|^2$ where $\|g\|^2$ represented the small-scale fading caused by a variety of factors, such as the Doppler shift and multipath propagation, which was the result of meeting the requirements of $\|g\|^2\mathcal{N}(1)$, an independent identically distributed circularly symmetric complex Gaussian vector with zero mean and variance 1. In addition to this, β_0 represented the path loss experienced by the signal-per-unit reference distance, d was the communication distance, and the form of the gain could be directly seen from the fact that it exhibited an inversely proportional function to the communication distance, with the gain decreasing as the distance increased. And α was the path-loss exponent, which was usually taken between 2 to 7.

3.3. Content-Caching Model

Every time slot was thought to have a content request from each MT, and the content of these requests could be cached by the edge server. The content servers stored a collection of contents that were denoted by $\mathcal{F}$, and each content had a size of $SIZE_f$ bits. The server cache capacity at each base station was, respectively, C_S and C_M. If an MT requested content, it could obtain it directly from the core network or from the closest edge server (i.e., an SBS or the MBS).

The collaborative edge-caching approach comprised two phases (as illustrated in Figure 2): content distribution and content placement.

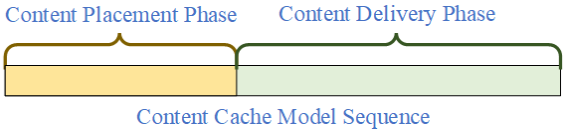


Figure 2. Collaborative edge-caching policy.

In order to forecast the content popularity, the content placement phase made use of previous request data. We categorized the content popularity into different categories, to simulate the request patterns of the content in different interest categories.

The content delivery model described the four ways for MTs to obtain requested content, each involving different transmission-delay calculations.

3.3.1. Content Placement Phase

The ratio of requests (for a specific piece of content to all requests) for the material in the MTs was known as content popularity. At the start of each time interval, the historical request data of the content $H = [h^{t-\tau}, h^{t-\tau+1}, \ldots, h^t]$ determined the content-popularity-prediction model, while the symbol τ represented the duration of the historical-data

observation window [29]. The number of requests for time slot t and content f were indicated by the symbol h_f^t. We then applied the prediction to forecast how popular the material would be during the following time slot.

We considered the variability and complexity of the network environment, and we set it so that the different MTs had different preferences for requested content. We categorized the content popularity of the different service ranges into different classes as $\mathcal{K} = \{1, 2, \ldots, k, \ldots, K\}$, where $\gamma_n^k \in \{0, 1\}$ denoted the interest class to which each base station belonged.

We utilized a homogeneous Poisson process with density $\lambda_f = \wedge pr_f$ models, as well as the requested process for a given content f [30], where $\wedge$ was the number of times the content was requested and pr_f denoted the request probability of the content f. For each base station, the popularity of the content f in the interest category to which it belonged could be modeled via the Zipf distribution with the following expression:

$$pr_{mf}^k = \frac{\left(H_f^k\right)^{-\iota}}{\sum\limits_{i \in F} \left(H_i^k\right)^{-\iota}}, \forall f \in \mathcal{F}, \forall m \in \mathcal{M}, \tag{2}$$

where H_f^k denoted the popularity ranking of content f in interest category k, ι reflected the degree of Zipf skewing, and a larger value of ι indicated a higher concentration of content popularity.

At time slot t, BS m had the ability to determine whether and where to store the requested content f for MT n. We defined $y_{m,n}^t = \{-1, 0, 1\}$ as the content-caching decision of BS m in time slot t. When $y_{m,n}^t = -1$, BS m had no content caching at time slot t. When $y_{m,n}^t = 0$, it meant that in time slot t the content f was cached in SBS. When $y_{m,n}^t = 1$, it meant that in time slot t the content f was cached in MBS, i.e., BS $m = 0$.

3.3.2. Content Delivery Phase

We identified four distinct strategies for the mobile terminals (MTs) to access requested content within a heterogeneous network, thereby leveraging collaborative edge caching.

CASE 1: SBSs to MTs.

In this case, the SBSs were directly transmitted to the MTs through a cellular link. The transmission delay for the content requested by MT n was given by

$$D_{m,n}^{sbs}(t) = \frac{SIZE_f}{V_{m,n}(t)}. \tag{3}$$

CASE 2: Neighboring SBS to MTs.

If the local SBS did not have the requested content, it was sourced from a nearby SBS that had the content cached. The content transfer latency from this neighboring SBS $m^{'}$ to MT n was expressed as

$$D_{m^{'},n}^{ns}(t) = D_{m,n}^{sbs} + D_{SBS}, \tag{4}$$

where Dm, n^{sbs} was the wireless transmission delay for acquiring the content, and D_{SBS} represented the wired transmission delay from a neighboring SBS to the local SBS.

CASE 3: MBS to MTs.

If the content requested by MT n was only cached by the MBS, it was first sent to the local SBS, which then delivered it to the MT. The content delivery delay in this scenario was

$$D_{0,n}^{mbs}(t) = D_{m,n}^{sbs}(t) + D_{MBS}, \tag{5}$$

where $DMBS$ was the transmission delay from the MBS to an SBS. It was assumed that all SBSs within the same MBS coverage area were interconnected via optical fiber, thus enabling a consistent, short transmission delay [31].

CASE 4: Core network to MTs.

When the requested content was not available by any of the above methods, i.e., not cached by any edge-cache server, then the data were transferred from the core network to the MBS by a backhaul link, which was subsequently relayed to the MTs. The latency for content transfer was given by

$$D_n^{CN}(t) = D_{0,n}^{mbs}(t) + D_{CN}, \tag{6}$$

where $D_{CN} = SIZE_f / B_n^{CN}$ was the backhaul delay, $B_n^{CN} = pr_{m,f}^k \overline{B}$ was the backhaul bandwidth of the MT n requesting the content f, and $\overline{B}$ denoted the average data transfer rate in the internet [32].

3.4. Problem Description

In the context of SBS m, the transmission delay experienced by MT n in time slot t when requesting content f is articulated as follows:

$$D_n(t) = \begin{cases} D_{m,n}^{sbs}(t), & m \neq 0, m \in \mathcal{M} \\ D_{m',n}^{ns}(t), & m' \neq m \neq 0, m' \in \mathcal{M} \\ D_{0,n}^{mbs}(t), & m = 0 \\ D_n^{CN}(t), & \text{not cached} \end{cases}. \tag{7}$$

Here, MT $n \in \mathcal{N}_m$ seeks content f sequentially from the local SBS, a neighboring SBS, the MBS, and finally, the core network.

The total content delivery latency for all the MTs in BS m is calculated as

$$D_m(t) = \sum_{n \in \mathcal{N}_m} D_n(t). \tag{8}$$

For BS k, the cache hit rate during time slot t is defined as

$$C_m(t) = \frac{c(t)}{|\mathcal{N}_m|}, \tag{9}$$

where $C_m(t) \in [0, 1]$, $c(t)$ denotes the number of MT n request contents that have been cached by the edge server, and $|\mathcal{N}_m| = N_m$ is the number of contents requested by all MTs within the BS m coverage. At time slot t, an MT requests only one content.

To minimize each BS's delivery latency while maximizing the cache hit rate, we formulated the content-caching-and-bandwidth-allocation challenge as Problem 1. This problem aimed to minimize the disparity between the content delivery latency and the cache hit rate in each small cell for any BS m. It is presented as follows:

$$\begin{aligned} \textbf{Problem 1:} & \min_{\mathbf{y}_m(t), \mathbf{x}_m(t)} D_m(t) - C_m(t), \\ \text{s.t.} & \textbf{C1:} y_{m,n}^t \in \{-1, 0, 1\}, n \in \mathcal{N}_m, m \in \mathcal{M} \backslash \{0\}, \\ & \textbf{C2:} x_{m,n}(t) \in (0, 1), n \in \mathcal{N}_m, m \in \mathcal{M} \backslash \{0\}, \\ & \textbf{C3:} \sum_{n \in \mathcal{N}_m} x_{m,n}(t) = 1, m \in \mathcal{M} \backslash \{0\}. \end{aligned} \tag{10}$$

In this model, $\mathbf{y}m(t) = ym, n^t(t) : n \in \mathcal{N}m$ and $\mathbf{x}m(t) = xm, n(t) : n \in \mathcal{N}m$ represent the content-caching decision matrix and bandwidth-allocation matrix for SBS m, respectively. Constraint C1 denotes the constraints on the caching decisions, C2 denotes the constraints on the bandwidth-allocation ratio, and C3 ensures the sum of the bandwidths allocated to associated MTs via each SBS, which equals its total bandwidth. These con-

straints add complexity to the problem, as they restrict the solution space. It is important to note that the total volume of all cached content must not exceed the edge server's total storage capacity. Problem 1 is a mixed-integer-nonlinear-programming problem, and it poses challenges in a direct polynomial-time resolution.

Next, we employed the offline training of content-popularity-prediction models. We considered heterogeneous-environment variability and organically combined content similarity. We introduced a long short-term memory–content popularity prediction (LSTM–CPP)-model content placement strategy to accurately predict the content popularity. In addition, considering the content complexity of the MT requests and the variability of the channel states, we designed a collaborative caching strategy based on multi-intelligent actor–critic networks, to solve the optimization problem.

4. LSTM-Based Content Placement Strategy

For edge-cache management, advance knowledge of content popularity plays a key role. For this purpose, we developed an innovative offline training model to predict content popularity. With historical content-request data as the input, the model improved prediction accuracy by fusing correlation and time-periodic features. It also addressed the content variability in heterogeneous network environments by proposing a method for predicting the popularity of new content based on content similarity.

4.1. Content Popularity Prediction

The requirements of MTs in heterogeneous networks are complex and variable, and there are significant correlations between content requests. By using historical request data, we aimed to explore these relationships, to improve the predictive accuracy of content popularity. At first, we introduced the LSTM model for predicting the popularity of web content. With its advantage in processing time-series data, the LSTM model can effectively capture the long-term dependencies of content requests, thus improving the accuracy of popularity prediction. The model is a regression model.

To improve the prediction accuracy further, we proposed the LSTM–CPP model, as shown in Figure 3. The model achieves accurate predictions of content popularity by analyzing the correlation between the low-dimensional feature information and time-domain features of different content-request patterns. Generating appropriate features from a historical content-request sequence H for input into a predictive model involves feature learning, where the model extracts the most representative features $\mathbf{h}$ from the content-request sequence H across all the past nodes through an encoder. This process is formally expressed as follows:

$$\mathbf{h} = f(H) = s_f(W_{AE}H + b_{AE}), \tag{11}$$

where W_{AE} and b_{AE} represent the encoder's weight matrix and bias vector, respectively, and $s_f(\cdot)$ denotes an activation function (typically a sigmoid function).

After mapping $\mathbf{h}$ to a lower-dimensional hidden layer, the data are then mapped to the output layer through a decoding function $\mathbf{g}(\cdot)$, which is expressed as

$$\widehat{\mathbf{h}} = g(\mathbf{h}) = s_g(W_{AE}\mathbf{h} + b_{AE}). \tag{12}$$

The aim of auto-encoding is to replicate the original data; thus, the difference before and after reconstruction is used to obtain the error value. If the error is significant, it indicates that the hidden layer has not extracted effective features from H. Therefore, it is necessary to continuously adjust the generation of W_{AE} and b_{AE} during the encoding and decoding process to minimize the reconstruction loss function $\mathcal{L}_{AE}$, which is specifically defined as follows:

$$\mathcal{L}_{AE} = \min_{W_{AE}, b_{AE}} \sum_{i=1}^{m} (\widehat{\mathbf{h}} - \mathbf{h})^2. \tag{13}$$

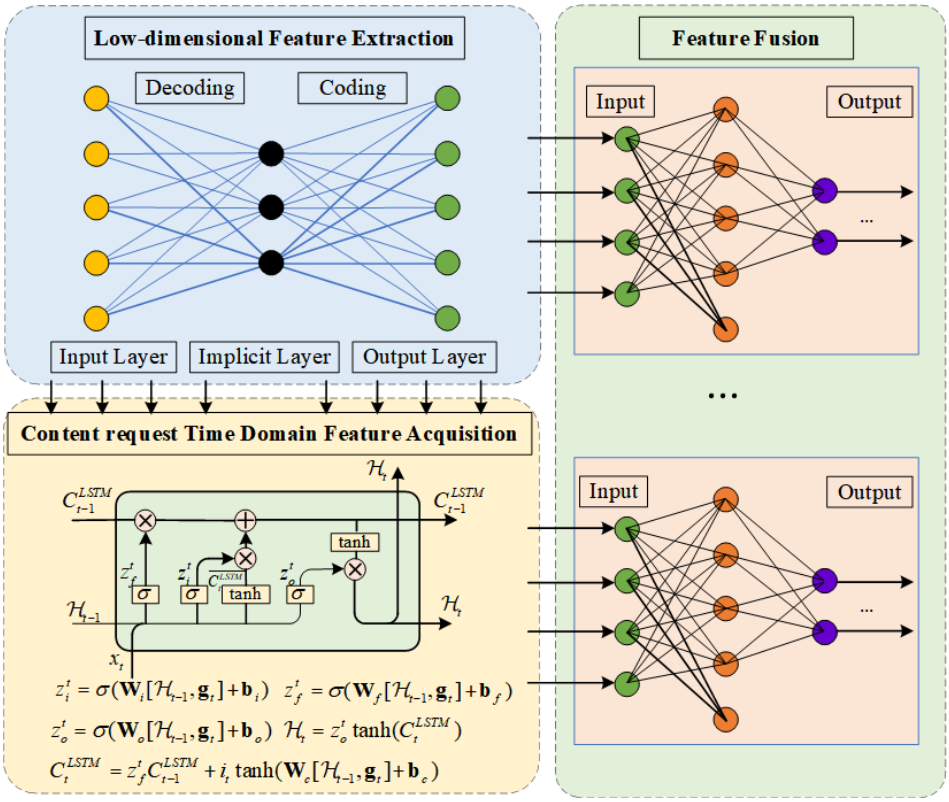


Figure 3. Long-short-term-memory–content-popularity-prediction model.

Additionally, long short-term memory (LSTM) networks are used to capture the temporal features of a content request sequence. LSTM networks control the features of content-request patterns over long and short timescales through three control gates, i.e., the input gate z_i^t, the forget gate z_f^t, and the output gate z_o^t, as follows:

$$z_i^t = \sigma(\mathbf{W}_i[\mathcal{H}_{t-1}, \mathbf{g}_t] + \mathbf{b}_i), \tag{14}$$

$$z_f^t = \sigma(\mathbf{W}_f[\mathcal{H}_{t-1}, \mathbf{g}_t] + \mathbf{b}_f), \tag{15}$$

$$z_o^t = \sigma(\mathbf{W}_o[\mathcal{H}_{t-1}, \mathbf{g}_t] + \mathbf{b}_o), \tag{16}$$

$$C_t^{LSTM} = z_f^t C_{t-1}^{LSTM} + i_t \tanh(\mathbf{W}_c[\mathcal{H}_{t-1}, \mathbf{g}_t] + \mathbf{b}_c), \tag{17}$$

$$\mathcal{H}_t = z_o^t \tanh(C_t^{LSTM}). \tag{18}$$

In the LSTM network, the cell state C_t^{LSTM} and the hidden-layer output variable $\mathcal{H}_t$ are crucial components that store and transfer information throughout the sequence processing. The dynamics of these components are governed by several parameters and operations, primarily involving weight matrices and bias terms that correspond to different gates and cell interactions, as well as to the nonlinear activation functions σ and $\tanh()$. Furthermore, $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o$, and $\mathbf{W}_c$ are the weight matrices between the input gates, oblivion gates, output gates, and the cells, respectively. In addition, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, and $\mathbf{b}_c$ are the to-be-learned bias terms between the LSTM input gates, oblivion gates, output gates, and cells, respectively.

We converged the relevance and temporal multi-dimensional characterizations of the content requests over a fully connected network as follows:

$$\mathbf{O} = \mathbf{O}_1 \oplus \mathbf{O}_2 \cdots \oplus \mathbf{O}_t, \tag{19}$$

where $\mathbf{O}_t$ represented the content-request-pattern features extracted for each time slot. After three layers of fully connected networks and activation operations, the model output the content popularity $\hat{\mathbf{Y}} = \delta(\mathbf{O})$ for the next moment.

The purpose is to perform dimensionality reduction on the input data, due to the dimension of the auto-encoder's hidden layer being smaller than the input layer. However, this does not prevent the opposite situation, wherein the dimensionality of the hidden layer is larger than that of the input layer, thus resulting in an increase in data dimensionality. In the heterogeneous network's changing environment, content constantly changes. According to YouTube data analysis, including short videos, popular news updates every 2 to 3 h, while new movies are released almost weekly. When new content emerges, there is not enough of an accumulation of historical data. Therefore, when the frequency of user content requests is low, or when new content appears, it may lead to sparsity in the historical request sequence itself or the data being affected by noise, thereby increasing the reconstruction error. A denoising auto-encoder was thus adopted in this context, as illustrated in Figure 4.

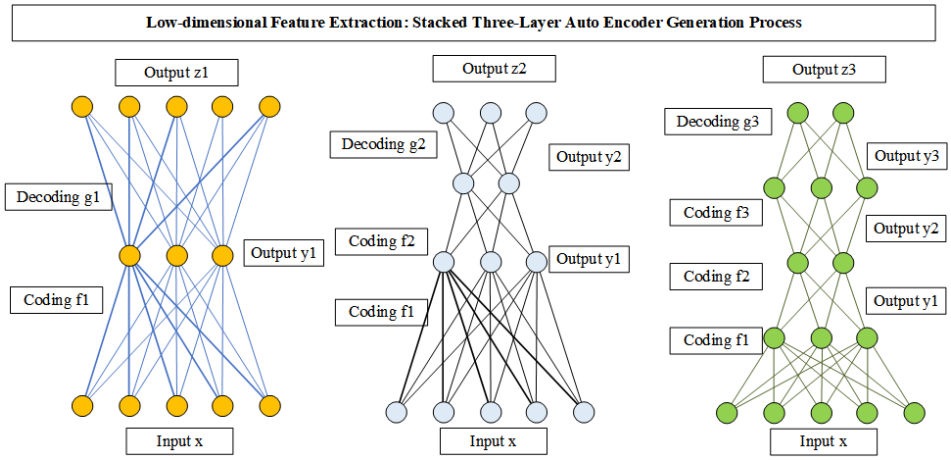


Figure 4. Low-dimensional feature extraction.

Denoising auto-encoders consider that features capable of restoring original data are not necessarily optimal. Only those features extracted from "contaminated" data that can still replicate the initial data can express the hidden information of the data. They are encoded and decoded to obtain data $\mathbf{g}(f_\theta(\widetilde{H}))$. However, in practice, a single layer of a denoising auto-encoder is insufficient for extracting features from historical content-request data. Therefore, it is feasible to cascade multiple denoising auto-encoders to form a stacked denoising auto-encoder (SDAE) for network training, and this is performed to complete data feature extraction, with the extracted features serving as inputs to the classifier. SDAE, as a hierarchical coding model, takes the output of one layer as the input of the next. In addition, after multiple layers of training, the deep features of the data can be obtained. This process does not require the addition of labels during input sample processing; hence, it is referred to as unsupervised pre-training. After pre-training, fine tuning is required. Firstly, all hidden-layer parameters are initialized, thereby forming a matrix containing all hidden-layer bias vectors learned during pre-training; then, the weight matrix and bias matrix are randomly initialized; and, finally, the entire model's parameters are adjusted by the backpropagation algorithm, which is specifically expressed as follows:

$$J_B = \frac{1}{2TS'}\|\mathbf{h} - \widetilde{\mathbf{h}}\|, \tag{20}$$

where TS' represents the training samples. The model parameters are fine tuned by comparing the label inputs with the model outputs. The Adam algorithm updates all weight matrices and bias terms in the model, to minimize the loss between predicted and actual values. The collective loss function expression is

$$\mathcal{L}^{LSTM} = \min_{\mathbf{W},\mathbf{b}} \|\hat{\mathbf{Y}} - \mathbf{Y}\|. \tag{21}$$

4.2. Content-Popularity-Prediction Methods in Varying Environments

In the dynamic environment of heterogeneous networks, content flows change in real time. The lack of historical data for new content makes it difficult to predict its popularity. To address this challenge, we propose the LSTM–CPP strategy. It uses the request data of new content at the current moment to compute the similarity to the existing content. The similarity between the new content f and the existing content $f^{'}$ is calculated by the following formula:

$$sim_{f,f'} = \frac{1}{\left| h_f^t - h_{f'}^t \right| + 1}. \tag{22}$$

Here, h_f^t and $h_{f'}^t$ represent the number of requests for content f and $f^{'}$ at time t, respectively. The similarity metric $sim_{f,f'}$ falls within the range $(0, 1]$.

The next step involves estimating the number of requests for new content based on its similarity to existing content. This estimation is pivotal in predicting the popularity of the new content. The formula for this estimation is as follows:

$$\widehat{h}_f^{t+1} = \frac{\sum_{f' \in \mathcal{F}_{ed}} sim_{f,f'} \widehat{h}_{f'}^{t+1}}{\sum_{f' \in \mathcal{F}_{ed}} sim_{f,f'}}. \tag{23}$$

In this expression, $\mathcal{F}ed$ denotes the set of existing similar content and $\widehat{h}f^{'t+1}$ is the predicted number of requests for existing content $f^{'}$ at time $t+1$, which is based on the content-popularity-prediction model. The popularity of content f at moment $t+1$ can be expressed as

$$\hat{p}r_{mf}^{k,t+1} = \frac{\widehat{h}_f^{t+1}}{\sum_i^F \widehat{h}_i^{t+1}}. \tag{24}$$

The LSTM–CPP model utilizes these methods in a multi-variable environment, to predict the popularity of new content until there is sufficient accumulated historical request data to effectively train a content-popularity-prediction model. Subsequently, the model predicts the popularity of existing content using the trained model, and it then iterates this process.

This entire workflow of the cache-placement policy, as proposed and based on the LSTM–CPP model, is encapsulated in Algorithm 1.

The complexity of Algorithm 1 is influenced by multiple factors, including the number of existing contents $|F_{\text{ed}}|$, the number of new contents $|F_{\text{new}}|$, the length of the time series T, and the structure of the SAE and LSTM networks. Specifically, the complexity of the SAE encoding step is approximately $O(|F_{\text{ed}}| \cdot L_{SAE} \cdot N_{SAE}^2)$. The complexity associated with the LSTM processing step is approximately $O(|F_{\text{ed}}| \cdot T \cdot N_{LSTM}^2)$, whereas the complexity for feature fusion and FC network prediction is approximately $O(T \cdot L_{FC} \cdot N_{FC}^2)$. The complexity for estimating the popularity of new content is $O(|F_{\text{new}}| \cdot |F_{\text{ed}}|)$. Collectively, these steps result in relatively high complexity when the algorithm processes a large volume of content and lengthy time series. Consequently, it is imperative to meticulously select the network architecture and parameters to achieve an optimal balance between prediction performance and computational efficiency, especially considering potential constraints on computational resources. However, as this algorithm is designed for offline pre-training models, the high time complexity can be offset by the resultant improvement in prediction accuracy.

Algorithm 1 Long short-term memory–content popularity prediction (LSTM–CPP) algorithm.

Require:
H: Historical content request data, $H = [h_1, \ldots, h_f, \ldots, h_F]$
F_{ed}: Set of existing contents
τ: Historical data observation-window length
Ensure:
$\hat{Y}$: Predicted popularity for each content
Initialize the stacked auto-encoder (SAE)
for each content f in F_{ed} **do**
    Extract the low-dimensional representation g_f using an SAE trained on H
    $g_f = \text{Encode\_SAE}(H[f])$
end for
Initialize the LSTM network
for each representation g_f **do**
    Process g_f with LSTM to capture the time-dimension features H_t
    $H_t = \text{LSTM}(g_f)$
end for
Initialize the feature-fusion module
for each time slot t **do**
    Fuse multi-dimensional features g_f and H_t to form O_t
    $O_t = g_f \oplus H_t$
    Predict content popularity $\hat{Y}_t$ using a fully connected (FC) network on O_t
    $\hat{Y}_t = \text{FC}(O_t)$
end for
for new content f_{new} in each time slot **do**
    Calculate similarity $sim_{f,f'}$ between f_{new} and each content in F_{ed}
    $sim_{f,f'} = \frac{1}{|h^t_{f_{\text{new}}} - h^t_{f'}| + 1}$
    Estimate the popularity of f_{new} based on weighted similarity
    $\hat{Y}_{f_{\text{new}}} = \frac{\sum_{f' \in F_{\text{ed}}} sim_{f,f'} \hat{Y}_{f'}}{\sum_{f' \in F_{\text{ed}}} sim_{f,f'}}$
end for
return $\hat{Y}$

5. Collaborative Caching Strategies Based on Multi-Intelligent Actor–Critic Networks

The multi-intelligence actor–critic collaborative caching policy (MACP) algorithm provides an effective method through which to manage and optimize complex heterogeneous network systems, as shown in Figure 5.

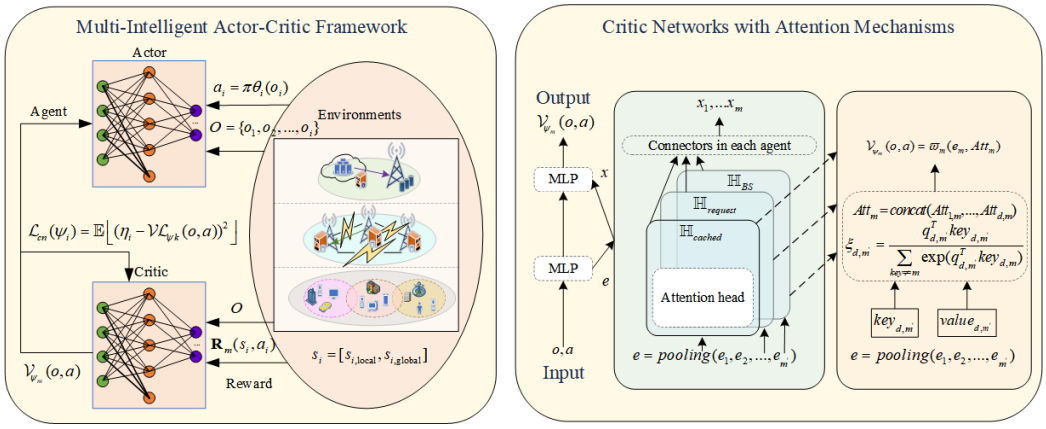


Figure 5. Multi-intelligence actor–critic collaborative caching policy (MACP) algorithm.

The LSTM–CPP model provides MTs with the foresight of future content demand by modeling content popularity analysis. The integrated actor–critic network with a multi-head

attention mechanism ensures optimal decision making for the overall network performance. We suggest employing a cooperative caching approach for delivering content, thereby utilizing multi-intelligent actor–critic networks as a solution to the edge-caching issue.

5.1. Multi-Intelligent Actor–Critic Framework

We constructed a networked system consisting of multiple BSs, each of which interacted with the information as an independent agent of intelligence. This system achieved efficient cache management in a heterogeneous network environment by combining actor networks and critic networks to accommodate the needs and preferences of different MTs.

- State Space:
  We defined the state space $S = [s_1, s_2, \ldots, s_i, \ldots, s_m]$, where $s_i = [s_{i,\text{local}}, s_{i,\text{global}}]$ contained a composition of the local and global information for each agent. The local information $s_{i,\text{local}}$ indicated the state of the cached list, and the global information $s_{i,\text{global}}$ focused on the content requests of the MTs, including historical and current data.
- Action Space:
  The action space was denoted as $A = [a_1, a_2, \ldots, a_i, \ldots, a_m]$, where $a_i = [a_{i,\text{cache}}, a_{i,\text{bw}}]$ and $a_{i,\text{cache}}$ denoted the process where the caching decision is the agent's decision on which content is to be stored or removed. The decision is based on the predicted content popularity and the current cache state. Moreover, $a_{i,\text{bw}}$ denoted the bandwidth-allocation decision, which involves how the agent allocates the available bandwidth resources among the MTs it serves. Each element indicates the proportion of bandwidth allocated to the corresponding MTs or tasks.
- Reward Mechanism:
  We designed the reward function $\mathbf{R}_m(s_i, a_i)$ to evaluate the effect of an agent's action according to the definition of the system model. The reward mechanism includes aspects such as reducing network latency, increasing cache hit rate, and improving the efficiency of bandwidth allocation. For behaviors that are effective in reducing network latency and improving cache hit rate, the agent will be positively rewarded. At the same time, we also considered the optimization of the bandwidth-allocation efficiency. For behaviors that reduce network latency, the agent should receive positive rewards:

$$\mathbf{R}_{delay}(s_i, a_i) = w_1 \cdot \Delta\text{delay}(s_i, a_i), \tag{25}$$

  where $\Delta\text{delay}(s_i, a_i)$ denotes the reduction in delay due to taking action a_i.
  An increase in cache hit rate should also be rewarded, especially if it matches the popular content predicted by the LSTM–CPP algorithm:

$$\mathbf{R}_{\text{hit\_rate}}(s_i, a_i) = w_2 \cdot \Delta\text{hit\_rate}(s_i, a_i), \tag{26}$$

  where $\Delta\text{hit\_rate}(s_i, a_i)$ denotes the increase in cache hit rate.
  Efficient bandwidth allocation should be equally rewarded:

$$\mathbf{R}_{eff}(s_i, a_i) = w_3 \cdot \Delta\text{eff}(s_i, a_i), \tag{27}$$

  where $\Delta\text{eff}(s_i, a_i)$ denotes the efficiency of resource utilization. In addition, w_1, w_2, and w_3 are the weighting coefficients. In summary, the total reward function can be expressed as follows:

$$\mathbf{R}_m(s_i, a_i) = \mathbf{R}_{delay}(s_i, a_i) + \mathbf{R}_{\text{hit\_rate}}(s_i, a_i) + \mathbf{R}_{eff}(s_i, a_i). \tag{28}$$

- Observation Mechanism:
  The observation mechanism $O = \{o_1, o_2, \ldots, o_i\}$ enables each agent to obtain the state of the local cache in its service area, as well as a sense of the overall network state. In addition, the agents can obtain the state information of neighboring SBSs through the established communication links.

- Actor Networks:
  An actor network is a parameterized policy network that defines the probability of choosing a particular action given an observation. Each agent observes the local state s_i of its service region, and it then makes a caching decision based on the current observation of its local state. For agent i, the actor network can be represented by the function $\pi\theta_i$, which outputs the probability distribution of taking action a_i under a given observation o_i as follows:
  $$a_i = \pi\theta_i(o_i). \tag{29}$$
  Here, θi denotes the parameters of the actor network.
- Critic Network:
  The determination to cache actions is made by the actor network implemented in each agent, and it relies on local knowledge. After the action selection in the actor network, the critic network selects actions based on the states $S = [s_1, s_2, \ldots, s_i, \ldots, s_m]$ of all the BSs and actions $A = [a_1, a_2, \ldots, a_i, \ldots, a_m]$, which guides the update of the actor network by evaluating the expected returns of the strategies through a value function. In addition, we introduced an attention mechanism to more accurately model the influence of other agents, thus making the evaluation process more accurate and efficient.

5.2. Critic Networks with Attention Mechanisms

We introduced a state-of-the-art critic network that employs an enhanced attention mechanism for the purposes of in-depth analysis and optimization of the collaborative caching strategies among BSs. This network structure not only considers the local information of each agent, but also integrates the insight of the global network state.

It achieves efficient learning and decision making in complex heterogeneous network environments.

5.2.1. Optimization of the Embedding Layer

We used a multi-layer perceptron (MLP) to process the observed features and to adapt to the diverse needs of different BSs in providing services to MTs. Due to variations in the caching capability of the BSs, content sizes, and content quantities at each base station, the length of the related coded vector list was not set. It was not found to be suitable for use as an input in a fully linked network that can only process inputs of a defined length. To represent the type, we employed one-hot or multi-hot encoding techniques [33]. We embedded the sparse embedding vectors $e_{m'}$ of the cached content in the BSs with embedding techniques. We used pooling to convert it into dense embedding vectors $e = pooling(e_1, e_2, \ldots, e_m, \ldots, e_{m'})$.

5.2.2. Attention Mechanism Refinement

The attention layer utilizes a fine-grained, multi-head attention mechanism. The attention mechanism processes a set of multidimensional information $Q = q1, q2, \ldots, q_d$, where d represents the number of different types of features. Each attention head operates in parallel, thereby focusing on distinct aspects of the input message. This parallel processing enables the network to concurrently consider multiple informative elements, which enhances the richness and depth of the analysis.

Each attention header focuses on a specific subspace of features, such as the size, type, request frequency, and cache capacity of the cached content, thus providing a more comprehensive understanding of the network state. The key $key_{d,m'}$ and the value $value_{d,m'}$ are the information of neighboring SBSs on feature d. We obtained the attention on feature d for the neighboring node $m^{'}$, which was defined as the attention weight, as follows:

$$\xi_{d,m'} = \frac{q_{d,m'}^T key_{d,m'}}{\sum\limits_{key \neq m} \exp(q_{d,m'}^T key_{d,m})}. \tag{30}$$

We computed the cooperative unit $Att_{d,m}$ by focusing on the weighted sum of the weights $\xi_{d,m'}$ and the values $value_{d,m'}$. We obtained the computed weights $key_{d,m'}$ and captured features $value_{d,m'}$ through multiple linear embedding layers $Att_{d,m}$, which is defined as

$$Att_{d,m} = \sum_{m' \neq m} \xi_{d,m'} value_{d,m'}. \tag{31}$$

This mechanism is particularly well suited for capturing and responding to the subtle changes in a network state, so as to support caching decisions.

5.2.3. Integration of the Output Layer

After the attention layer, we integrated the outputs of each attention head through a concatenate operation. Then, we generated the final output via linear transformation. The concatenation function, which aggregates the attention from multiple heads, is defined as

$$Att_m = concat(Att_{1,m}, ..., Att_{d,m}). \tag{32}$$

This process facilitates the efficient estimation of the behavior value function performed by the critic network for each BS as follows:

$$\mathcal{V}_{\psi_m}(o, a) = \varpi_m(e_m, Att_m), \tag{33}$$

where ϖ_m is an MLP layer and ψ_m denotes the parameter set of the target critic network associated with the agent.

5.3. Multi-Intelligence Actor–Critic Collaborative Caching Policy Algorithm

The multi-intelligence actor–critic collaborative caching policy algorithm that we implemented is based on a reinforcement-learning framework. This algorithm is particularly designed to evaluate and iteratively update the policy, thereby taking into consideration the intricate dynamics of the network and the unique roles played by different agents within it. Its primary objective is to enhance the efficiency and precision of caching decisions across the network, whereby it factors in the complex interplay of various elements and agent-specific functions.

5.3.1. Strategy Evaluation and Iterative Optimization

We utilized the value function Q to estimate the total payoff that an agent may receive in the future under a particular state S and action a. This function reflects the long-term payoff effect of implementing a particular action in a particular context as follows:

$$\mathcal{V}_i(o, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{+\infty} \mu^t r_{t+1} | o_0 = o, a_0 = a \right]. \tag{34}$$

To stabilize the learning process and enhance the algorithm's performance, we introduced a target network. In our approach, $\mathcal{V}_{\psi_m}(o, a)$ was utilized to approximate $\mathcal{V}_i(o, a)$, thereby facilitating more accurate predictions and efficient learning.

5.3.2. Utilization of Dominance Functions

We recognized the advantage function $A_i^a(o, a)$ as a function that can aid in determining the advantage of the current action over the average action level. Accordingly, we guided the gradient update of the strategy. The advantage function considers the additional benefits in multi-environment interactions. Importantly, while it provides an additional perspective for evaluation, it does not alter the fundamental decision-making process of the agent, as is expressed in the following:

$$A_i^a(o, a) = \mathcal{V}_{\psi_m}(o, a) - \sum_{a_i' \in A_i} \pi\theta_i(a_i' | o_i) \mathcal{V}_{\psi_m}(o, (a_i' | a_i)). \tag{35}$$

This equation introduces a baseline for evaluating additional benefits in multi-environment interactions without impacting the agent's core decision-making process. The policy's gradient update is calculated, whereby the integration of the advantage function in our approach is ensured, as follows:

$$\nabla\theta_i J(\pi\theta_i) = \mathbb{E}_{\pi\theta_i}[\nabla\theta \log(\pi\theta_i(a_i|o_i))A_i(o, a, \psi_i)]. \tag{36}$$

5.3.3. Update Mechanisms for the Critic and Actor Networks

The critic network's update relied on a specifically defined loss function:

$$\mathcal{L}_{cn}(\psi_i) = \mathbb{E}\Big[(\eta_i - \mathcal{V}\mathcal{L}_{\psi k}(o, a))^2\Big], \tag{37}$$

where $\eta_i = r_i + \mu\mathbb{E}_{\pi\theta_i}\Big[\mathcal{V}_{\psi k}(o', a')\mathcal{V}_i^{\overline{\psi}}(o', a') - \alpha\log(\pi\overline{\theta}(a'|o'))\Big]$, μ was the decay factor, the terms $\overline{\psi}$ and $\overline{\theta}$ represented the network parameters prior to updating, and α determined the balance between the maximum entrance and return [34].

We focused on minimizing the deviation between the predicted and actual values. Meanwhile, the actor-network updates were focused on maximizing the expected return of the strategy. In order to post each iteration, we updated the parameters $\overline{\psi}$ of the critic network and $\overline{\theta}$ of the actor network as follows:

$$\overline{\psi} = \psi - lr_{cr}\nabla\mathcal{L}_{cn}(\psi_i), \tag{38}$$

$$\overline{\theta} = \theta - lr_{ac}\frac{\nabla\theta_i J(\pi\theta_i)}{\nabla\theta_i}, \tag{39}$$

where lr_{cr} and lr_{ac} were the learning rates, which ensured the stability and efficiency of the algorithm. Notably, the above equations were instrumental in introducing randomness into our strategy, as they effectively distributed the probability of each action, especially in scenarios where the agent might favor a single action. The specific process of this strategy is delineated in Algorithm 2.

5.4. Algorithm Complexity Analysis

Time complexity focuses on how quickly the time required for algorithm execution grows with the size of the input. In each iteration, the algorithm traverses all agents and performs a series of computations on each agent. These computations include observing states, selecting actions, executing actions, and observing new states with rewards. The updating of the actor network and critic network are the main computational steps for each agent. The updating of the actor network involves the computation of the gradient and the dominance function, while the updating of the critic network includes the computation of the loss function and the gradient descent of the parameters. Assume that the number of agents is N, where the dimensions of the state and action space for each agent are S and A, respectively. The operation of each agent in each iteration takes roughly $\mathbf{O}(S + A)$ time. Therefore, the overall time complexity of the algorithm can be estimated as $\mathbf{O}(N \cdot (S + A))$ [35].

This synthesis approach is particularly effective in dynamic and complex network environments, and it can significantly improve the efficiency of resource allocation and user satisfaction.

Algorithm 2 Multi-intelligence actor–critic collaborative caching policy (MACP) algorithm.

Input:
Set of SBSs, each as an intelligent agent
Initial set of states S
Predicted content popularity as determined via the LSTM model
Observation mechanism O
Set of parameters Θ for the actor networks
Set of parameters Ψ for the critic networks

Output:
Updated set of parameters Θ for the actor networks
Updated set of parameters Ψ for the critic networks

Initialize parameters Θ for the actor networks and Ψ for the critic networks
for each iteration **do**
  for each agent i **do**
    Observe current state s_i
    Choose action $a_i = \pi_{\theta_i}(o_i)$
    Execute action a_i
    Observe new state s_i' and reward r_i
  end for
  Update state set S
  for each agent i **do**
    Compute advantage function $A_i^a(o,a) = V_{\psi_i}(o,a) - \sum \pi_{\theta_i}(a_i'|o_i) V_{\psi_i}(o,(a_i'|a_i))$
    Update parameters θ_i of the actor network: $\theta_i = \theta_i + \alpha_{\text{an}} \nabla_{\theta_i} J(\pi_{\theta_i})$
  end for
  for each agent i **do**
    Compute loss function for the critic network $L_{\text{cr}}(\psi_i) = \mathbb{E}[(\eta_i - V_{\psi_k}(o,a))^2]$
    Update parameters ψ_i of the critic network: $\psi_i = \psi_i - \alpha_{\text{cn}} \nabla L_{\text{cr}}(\psi_i)$
  end for
end for
return Updated parameters Θ and Ψ

6. Simulation Analysis

We conducted extensive simulations to evaluate the system performance through Python and MATLAB. The simulation parameter configurations are first given, then we will analyze the performance of the algorithms and the comparative method analysis for the LSTM–CPP content placement strategy and MACP content delivery strategy, respectively. The experimental dataset is derived from the real-world dataset of YouTube in April 2018 and is used to evaluate the proposed content popularity prediction method. This dataset has records of the count observations (views, comments, likes, dislikes, etc.) of 1500 videos being posted in real time. After filtering out some of the missing and erroneous data, 50 files were selected as the experimental dataset.

The low-dimensional feature extraction for the cache-placement-strategy model was performed using a two-layer stacked denoising encoder with 20 neurons in total. The time-domain-feature acquisition was conducted using an LSTM network with 24 neurons. The feature-fusion part consisted of a three-layer fully connected network, where the number of neurons per layer was 12, 15, and 50. Furthermore, the learning rate was set to 0.0007 and the batch size to 32.

The network for a cache-delivery strategy was set up with a total of four micro base stations with storage sizes of 300, 340, 380, and 420. The number of mobile terminals served in the coverage area of each base station was 10, 7, 5, and 8, and the initial state of the terminals was randomly generated. The ReLU activation function was used for the actor and critic networks. The size of the experience pool was [20,000, 25,000, 15,000, and 20,000], and the batch size was [128, 128, 256, and 256].

6.1. Simulation Parameter Configuration

We utilized a dataset from the content-popularity-prediction method of [36]. Table 2 shows the important parameters of the experiment conducted in [37].

Table 2. Simulation parameter configuration.

Symbol	Setting	Parameter
$\mathcal{M}$	4	Number of base stations
$\mathcal{N}_m$	40	Number of mobile terminals
W_s	10 MHz	Size of bandwidth
P_s	1 w	Transmission power of SBS
ζ^2	10^{-14}	Power of background noise
D_{SBS}	10 ms	Delay of content delivery from neighboring SBS to SBS
D_{MBS}	10 ms	Delay of content delivery from MBS to SBS
$\overline{V}$	100 Mb/s	Average data rate of the core network
λ_f	0.003	SBS density
C_S	10 Mb	Capacity of SBS-equipped server cache
C_M	20 Mb	Capacity of MBS-equipped server cache
$\mathcal{F}$	1000	Number of contents
$SIZE_f$	1 Mb	Size of each content
τ	7	Step size of LSTM

6.2. Content-Popularity-Prediction Method

We analyze the content-popularity-prediction method by comparing it to the following different methods:

- LSTM: The content-popularity-prediction model of an LSTM uses a K-mean clustering algorithm to group different contents [38]. Moreover, the content popularity of each group was predicted using a long short-term memory network.
- DNN-LSTM: An end-to-end network from [39]. A two-layer fully connected network with [20, 10] neurons, as well as two-layer long and short-term memory network with [24, 24] neurons.
- CNN-LSTM: This method is different from DNN-LSTM. This model uses a one-dimensional convolutional neural network instead of a fully connected network [40].

We employed the LSTM–CPP model to forecast the content popularity for the upcoming time slot, thereby informing our cache placement policy. As illustrated in Figure 6, the LSTM–CPP model significantly reduced the prediction error, especially the root mean square error (RMSE), by 28.61% compared to the standard LSTM model. The LSTM model's relatively inferior performance was attributed to its inability to account for the correlation between different content request patterns. The reason for its especially large execution time was the popularity-prediction model that corresponded to one for each content of the method. LSTM uses a clustering method to process the data before prediction. If clustering correctly divides the data into clusters with similar patterns of prevalence during training, then the LSTM is likely to make more accurate predictions in most cases, resulting in lower MAEs; however, even a small percentage of inaccurate clustering may result in larger errors in individual cluster predictions, which can increase the RMSE value.

In our analysis, we compared the LSTM–CPP model to LSTM-DNN and LSTM-CNN. As can be seen in Figure 6, our proposed LSTM–CPP model achieved a better prediction performance than LSTM-DNN and LSTM-CNN. This enhanced accuracy can be attributed to the LSTM–CPP model's integration of unsupervised pre-training, which effectively initializes the network parameters. This approach enables the model to establish more suitable initial values for the parameters, thereby optimizing its predictive capability.

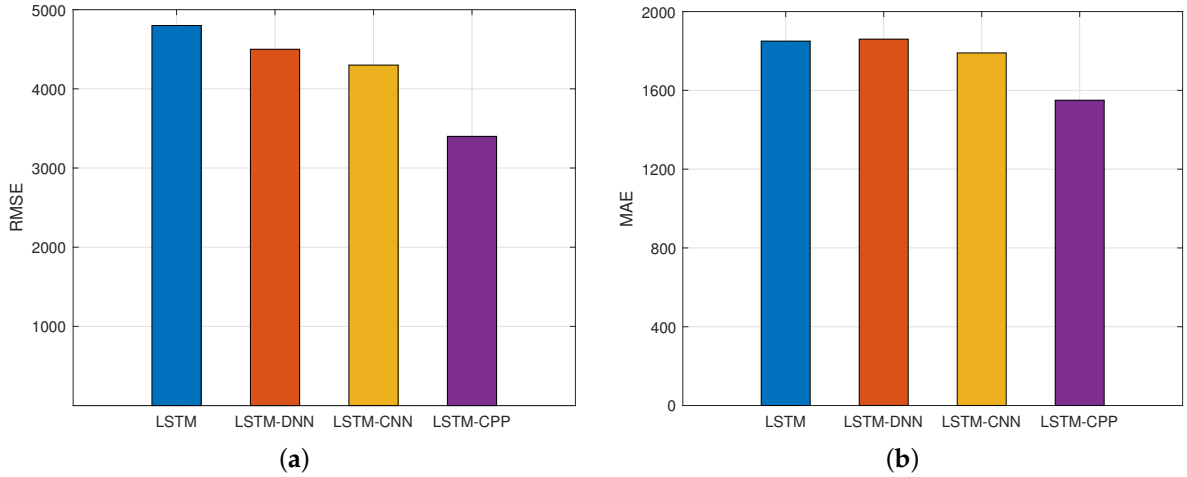


Figure 6. Comparison of the performance indicators: (**a**) MAE. (**b**) RMSE.

We plotted the training loss curves, as shown in Figure 7. The illustration clearly demonstrates that the LSTM–CPP model exhibited a faster convergence rate and superior prediction accuracy compared to LSTM, DNN-LSTM, and CNN-LSTM. Therefore, stacked auto-encoders for unsupervised pre-training can accelerate the model convergence speed and improve the model prediction performance. The dataset has multidimensional features and contains unlabeled data. The LSTM model only uses clustering algorithms to discover inherent groupings or patterns in the video data. However, our focus is on how to use a large number of video features to predict future trends, making an unsupervised pre-trained LSTM model potentially more effective.

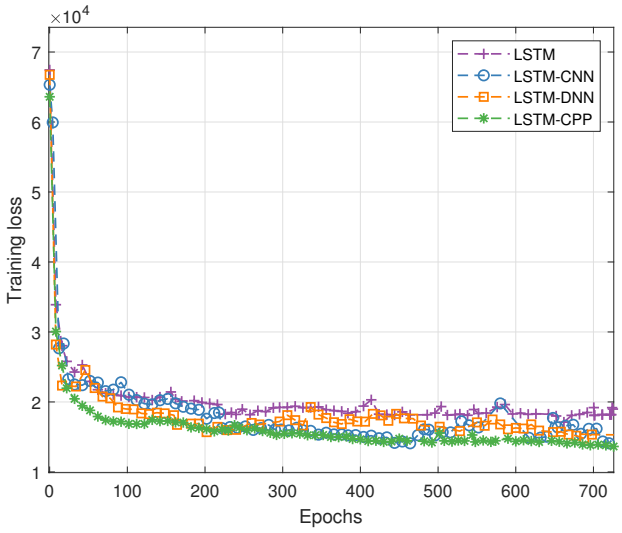


Figure 7. Training loss.

We compared the predicted values of the LSTM–CPP model to the true values, to analyze its prediction level, the results of which are shown in Figure 8. Our proposed LSTM–CPP method achieved better content-popularity-prediction performance and could roughly fit the true distribution. To improve the accuracy of content popularity prediction, we not only considered the correlation and time-domain characteristics between content requests, but also considered dynamic content changes and introduced the similarity between new and original content.

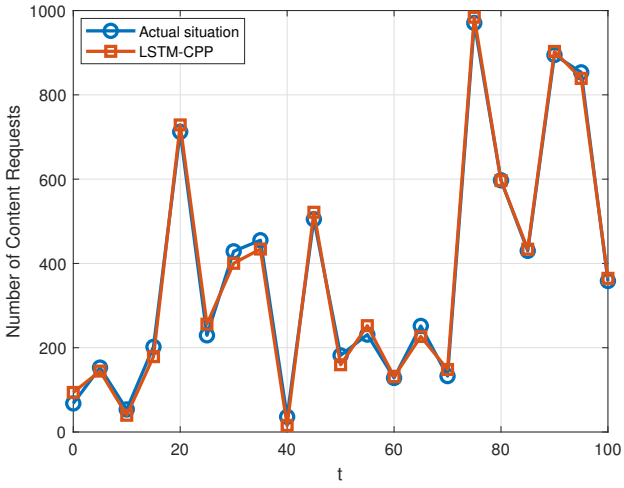


Figure 8. Prediction error.

Figure 9 depicts the cache-hit-rate performance of the system when under a content placement policy with and without prediction; as expected, the performance was found to be better under prediction. We used the LSTM–CPP model, which considers not only query relevance and user preferences, but also the similarity between old and new content, in the scenario of dynamic variability of content popularity in heterogeneous environments. This aspect is evident in Figure 9, which demonstrates the decrease in the cache hit rate as the numbers of content increased. However, it is worth noting that the LSTM–CPP scheme with content similarity had an 18.4%-higher cache hit rate than the scheme without content similarity (NCSC). In addition, it was also found to be much higher than the no projected programs (NPPs) approach.

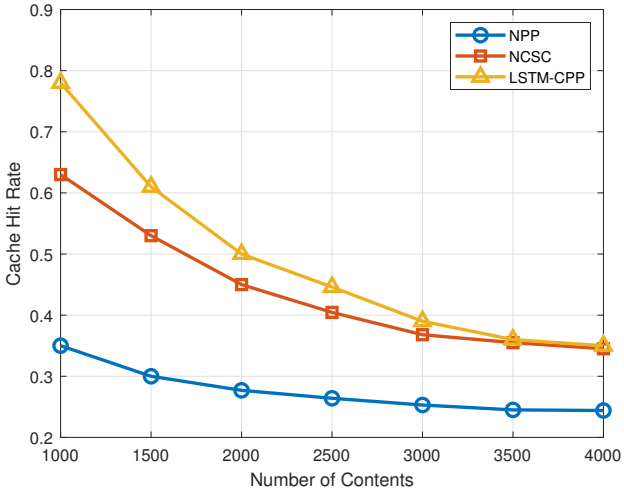
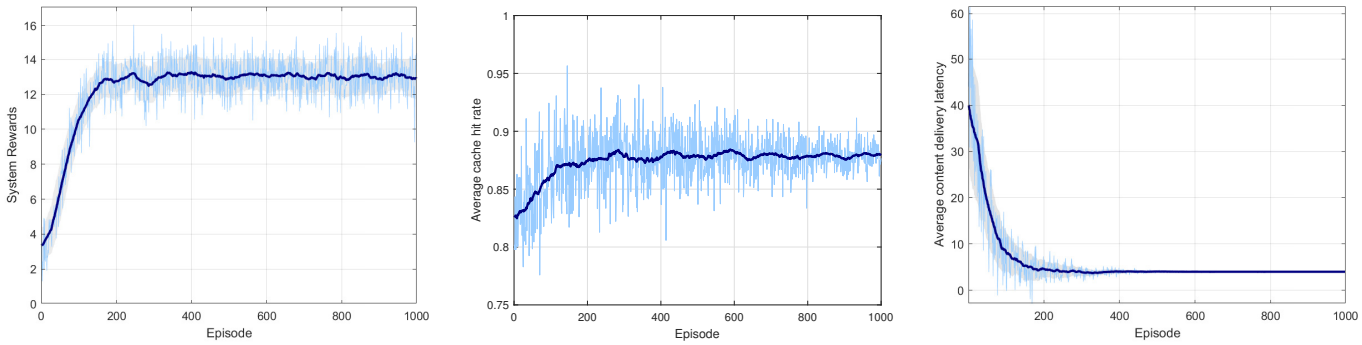


Figure 9. Effect of the content popularity prediction on the cache hit rate.

6.3. Efficient Caching Strategy

Figure 10 showcases the convergence performance of the MACP algorithm. In the figure, the following three key metrics are focused upon: system reward, average content delivery latency, and average cache hit rate. As the number of episodes increased, we observed a corresponding rise in the system reward. Notably, the growth rate of the system reward began to slow significantly and reached a steady state around episode = 200.

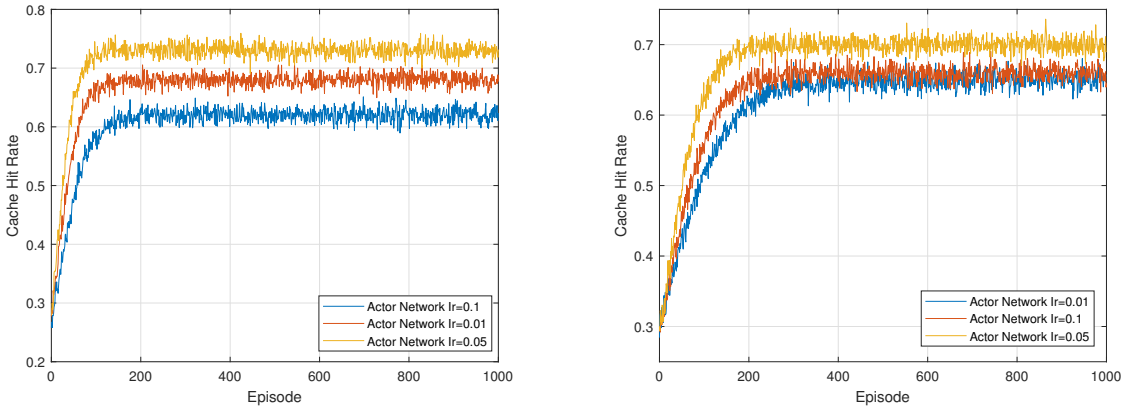


(**a**) (**b**) (**c**)

Figure 10. Algorithm convergence analysis: (**a**) System rewards. (**b**) Average cache hit rate. (**c**) Average content delivery latency.

During the training phase, the exploration noise was progressively reduced as the episode count increased. This adjustment was critical for the learning process. The reward function in our model is primarily composed of two factors: content delivery latency and cache hit rate. As we progressed through more episodes, there was a noticeable decrease in the content delivery delay; meanwhile, the cache hit rate experienced a gradual increase. This trend underscored the effectiveness of the MACP algorithm in optimizing both the timeliness of content delivery and the efficiency of cache utilization.

Figure 11 examines the influence of the learning rate on the cache hit rate, and this was achieved using the number of episodes as the horizontal axis to illustrate the effects of the varying parameters. We initiated our analysis by comparing the performance of the actor networks that were operating at different learning rates. To ensure the stability and efficiency of our algorithm, we set the learning rate for the actor network at 0.05.



(**a**) (**b**)

Figure 11. Learning-rate analysis: (**a**) Actor network. (**b**) Critic network.

Subsequently, we evaluated the cache hit rates achieved by the critic network at varying learning rates. The results indicated that our proposed algorithm attains the highest cache hit rate when the learning rate for the critic network is set to 0.05. This observation highlights the significance of optimizing the learning rate to achieve superior performance in terms of cache hit rate, thereby demonstrating the algorithm's efficacy in handling network caching tasks.

The multi-head self-attention mechanism employed in the MACP model effectively leveraged multiple sets of keys, $key_{d,m'}$, and values, $value_{d,m'}$. Here, $key_{d,m'}$ was utilized for the computing weights, while $value_{d,m'}$ was instrumental in capturing features. Figure 12 presents the experimental outcomes of incrementally increasing the number of heads in the multi-head self-attention mechanism from one to eight.

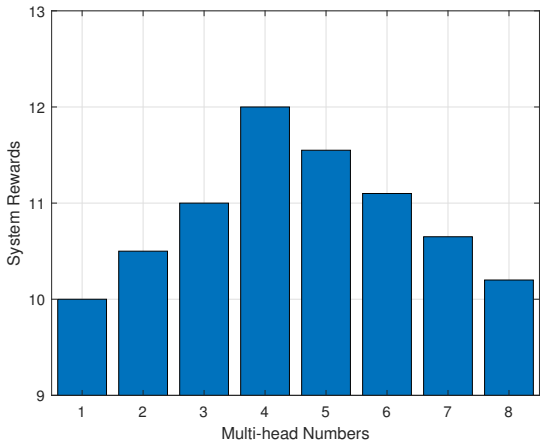


Figure 12. Analysis of the multiple attention mechanisms.

The experimental results indicated that, when the number of heads is relatively low, increasing their count can enhance the model's ability to capture user interest from a broader range of perspectives, thereby improving the model's performance. Optimal results were observed when the number of heads was set to four. Beyond this point, further increasing the number of heads led to a decline in performance. This trend underscores the importance of balancing the number of attention heads to achieve the best experimental outcomes.

In assessing our approach, we benchmarked it against the following state-of-the-art baseline algorithms:

- Actor–Critic (AC): This method employs a distinct memory structure that explicitly represents the policy, and this is true independent of the value function. Such a setup allows for a more nuanced policy development that is not directly tied to the value estimations [41].
- Deep Q Networks (DQN): DQN differs not only in its network structure, but also in its approach to feature handling. This method is grounded in local features and does not incorporate the influences from neighboring nodes' environments. This limitation could impact the algorithm's effectiveness in more interconnected or dynamic settings [42].

The performance of the hits is shown in Figure 13. MACP can be seen as clearly the highest. The aforementioned approach improved by 15.3% (AC) and 32.3% (DQN) when compared to the several commonly used conventional methods with different parameters. Heterogeneous networks are characterized by multi-server heterogeneity, and the AC and DQN methods formulate caching policies that are based on local BSs that do not consider the correlation of neighboring BSs. By contrast, the MACP algorithm we adopted formulates caching policies based on multi-BS information, and it is able to collaborate caching policies across multiple BSs, which greatly improves the system performance.

As shown in Figure 14, the MACP method reduces the system delay by 1.6% compared to the AC method. Although the performance of the MACP facet was slightly lower than the DQN, it was still found to be better in terms of hit rate. From the figure, it can be seen that the MACP approach improved with an increase in the cache capacity when compared to DQN. DQN, due to being a value-function-based algorithm, struggles to identify the maximum q-value among a large set of values, especially in scenarios with extensive action spaces. As the cache capacity expands, which leads to an even larger action space, the limitations inherent in DQN's algorithmic approach become more pronounced. This aspect underscores the advantage of the MACP approach in environments with growing cache capacities.

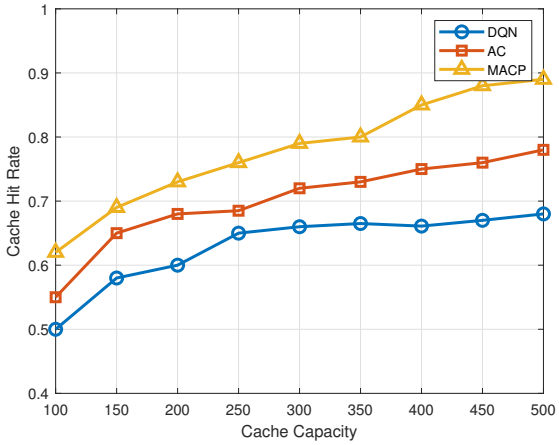


Figure 13. Analysis of the the hit ratio performance.

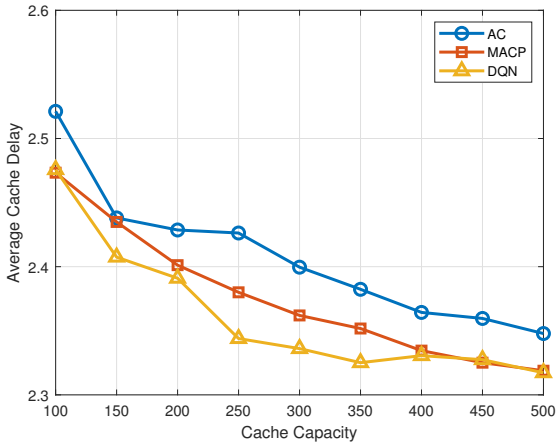


Figure 14. Analysis of the delay performance.

7. Conclusions

This work investigated precise content popularity prediction and efficient hybrid collaborative caching strategies to enhance heterogeneous network performance. Initially, based on the macro–micro structure of heterogeneous networks, we established a problem framework aimed at minimizing system latency and maximizing cache hit rates. Then, recognizing the crucial role of content popularity prediction in edge-cache placement, we proposed a novel content-popularity method that executes rapidly and effectively improves prediction accuracy. Lastly, considering multi-base station information in heterogeneous networks, we proposed a hybrid collaborative caching delivery strategy. The actor–critic method was employed to solve complex edge-caching issues within this strategy. We utilized a multi-head attention mechanism to quantify the influence of multiple base stations on the current node, thereby making effective caching decisions after collecting multidimensional global and local information, as well as significantly reducing backhaul traffic and enhancing the QoS/QoE. We conducted extensive simulations, and the results show that the LSTM–content-population-prediction model is more advantageous in terms of content-popularity-prediction accuracy, with a 28.61% improvement in prediction error, compared to several other existing methods. The proposed MACP algorithm improves the cache-hit-rate metric by up to 32.3% and reduces the system latency by 1.6%, demonstrating the feasibility and effectiveness of the algorithm. Building on this work, we will focus on user privacy issues in the future and will continuously refine our algorithms and framework according to real-world scenarios.

Author Contributions: Conceptualization, Z.S.; methodology, Z.S.; validation, Z.S.; data curation, Z.S.; writing—original draft preparation, Z.S.; writing—review and editing, G.C.; visualization, Z.S.; project administration, Z.S.; funding acquisition, G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the "Thirteenth Five-Year Plan" Science and Technology Research Project of Jilin Provincial Department of Education, Research on Large-scale D2D Access and Traffic Balancing Technology for Heterogeneous Wireless Networks JJKH20181130KJ, Special Project on Industrial Technology Research and Development of Jilin Province, Research on Selforganizing Network System of Unmanned Platform for Optoelectronic Composite Communication, 2022C047-8.

Data Availability Statement: Data are contained within the article.

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

References

1. Reiss-Mirzaei, M.; Ghobaei-Arani, M.; Esmaeili, L. A review on the edge caching mechanisms in the mobile edge computing: A social-aware perspective. *Internet Things* **2023**, *22*, 100690. [CrossRef]
2. Somesula, M.K.; Rout, R.R.; Somayajulu, D. Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks. *Comput. Netw.* **2022**, *209*, 108876. [CrossRef]
3. Wang, Q.; Chen, S.; Wu, M. Incentive-Aware Blockchain-Assisted Intelligent Edge Caching and Computation Offloading for IoT. *Engineering* **2023**, *in press*. [CrossRef]
4. Zyrianoff, I.; Gigli, L.; Montori, F.; Sciullo, L.; Kamienski, C.; Felice, M.D. Cache-it: A distributed architecture for proactive edge caching in heterogeneous iot scenarios. *Ad Hoc Netw.* **2024**, *156*, 103413. [CrossRef]
5. Liu, W.-X.; Zhang, J.; Liang, Z.-W.; Peng, L.-X.; Cai, J. Content popularity prediction and caching for icn: A deep learning approach with sdn. *IEEE Access* **2017**, *6*, 5075–5089. [CrossRef]
6. Zhou, H.; Jiang, K.; He, S.; Min, G.; Wu, J. Distributed deep multi-agent reinforcement learning for cooperative edge caching in internet-of-vehicles. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 9595–9609. [CrossRef]
7. Lee, D.D.; Pham, P.; Largman, Y.; Ng, A. Advances in neural information processing systems 22. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012, Lake Tahoe, NV, USA, 3–6 December 2009.
8. Sultan, M.T.; Sayed, H.E. QoE-aware analysis and management of multimedia services in 5 g and beyond heterogeneous networks. *IEEE Access* **2023**, *11*, 77679–77688. [CrossRef]
9. Salim, M.M.; Elsayed, H.A.; Elaziz, M.A.; Fouda, M.M.; Abdalzaher, M.S. An optimal balanced energy harvesting algorithm for maximizing two-way relaying d2d communication data rate. *IEEE Access* **2022**, *10*, 114,178–114.191. [CrossRef]
10. Fang, S.; Tang, R.; Guo, X. An adaptive adjusting density scheme of small cell base stations in heterogeneous cell networks. In Proceedings of the 5th International Conference on Communication and Information Processing, Chongqing, China, 15–17 November 2019; pp. 221–225.
11. Abdalzaher, M.S.; Moustafa, S.S.; Hafiez, H.A.; Ahmed, W.F. An optimized learning model augment analyst decisions for seismic source discrimination. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–12. [CrossRef]
12. Choi, Y.; Lim, Y. Deep reinforcement learning-based edge caching in heterogeneous networks. *J. Inf. Process. Syst.* **2022**, 18, 803.
13. Li, Y.; Ma, H.; Wang, L.; Mao, S.; Wang, G. Optimized content caching and user association for edge computing in densely deployed heterogeneous networks. *IEEE Trans. Mob. Comput.* **2020**, *21*, 2130–2142. [CrossRef]
14. Tang, J.; Tang, H.; Zhang, X.; Cumanan, K.; Chen, G.; Wong, K.-K.; Chambers, J.A. Energy minimization in d2d-assisted cache-enabled internet of things: A deep reinforcement learning approach. *IEEE Trans. Ind. Inform.* **2019**, *16*, 5412–5423. [CrossRef]
15. Tang, J.; Tang, H.; Zhao, N.; Cumanan, K.; Zhang, S.; Zhou, Y. A reinforcement learning approach for d2d-assisted cache-enabled hetnets. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
16. Sadeghi, A.; Wang, G.; Giannakis, G.B. Deep reinforcement learning for adaptive caching in hierarchical content delivery networks. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1024–1033. [CrossRef]
17. Hou, J.; Xia, H.; Lu, H.; Nayak, A. A graph neural network approach for caching performance optimization in ndn networks. *IEEE Access* **2022**, *10*, 112657–112668. [CrossRef]
18. Hassine, N.B.; Milocco, R.; Minet, P. ARMA based popularity prediction for caching in content delivery networks. In Proceedings of the 2017 Wireless Days, Porto, Portugal, 29–31 March 2017; pp. 113–120.
19. Ale, L.; Zhang, N.; Wu, H.; Chen, D.; Han, T. Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network. *IEEE Internet Things J.* **2019**, *6*, 5520–5530. [CrossRef]
20. Jiang, Y.; Feng, H.; Zheng, F.-C.; Niyato, D.; You, X. Deep learning-based edge caching in fog radio access networks. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 8442–8454. [CrossRef]
21. Lin, Z.; Sun, X.; Ji, Y. Landslide displacement prediction model using time series analysis method and modified lstm model. *Electronics* **2022**, *11*, 1519. [CrossRef]

22. Mannepalli, K.; Singh, S.P.; Kolli, C.S.; Raj, S.; Bojja, G.R.; Rajakumar, B.; Binu, D. Popularity prediction model with context, time and user sentiment information: An optimization assisted deep learning technique. *Int. J. Uncertain. Fuzziness-Knowl.-Based Syst.* **2023**, *31*, 283–302. [CrossRef]
23. Hu, Z.; Fang, C.; Wang, Z.; Tseng, S.-M.; Dong, M. Many-objective optimization based-content popularity prediction for cache-assisted cloud-edge-end collaborative iot networks. *IEEE Internet Things J.* **2023**, *11*, 1190–1200. [CrossRef]
24. Shekhar, S.; Singh, A.; Gupta, A.K. A deep neural network (dnn) approach for recommendation systems. In *Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2021*; Springer: Singapore, 2022; pp. 385–396.
25. Zhang, R.; Yu, F.R.; Liu, J.; Huang, T.; Liu, Y. Deep reinforcement learning (drl)-based device-to-device (d2d) caching with blockchain and mobile edge computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6469–6485. [CrossRef]
26. Zhong, C.; Gursoy, M.C.; Velipasalar, S. Deep multi-agent reinforcement learning based cooperative edge caching in wireless networks. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
27. Yan, H.; Xu, X.; Bilal, M.; Xia, X.; Dou, W.; Wang, H. Customer centric service caching for intelligent cyber-physical transportation systems with cloud-edge computing leveraging digital twins. *IEEE Trans. Consum. Electron.* **2023**. [CrossRef]
28. Lu, Y.; Zhang, P.; Duan, Y.; Guizani, M.; Wang, J.; Li, S. Dynamic scheduling of iov edge cloud service functions under nfv: A multi-agent reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2023**. [CrossRef]
29. Li, D.; Zhang, H.; Ding, H.; Li, T.; Liang, D.; Yuan, D. User preference learning-based proactive edge caching for d2d-assisted wireless networks. *IEEE Internet Things J.* **2023**, *10*, 11922–11937. [CrossRef]
30. Garetto, M.; Leonardi, E.; Traverso, S. Efficient analysis of caching strategies under dynamic content popularity. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 2263–2271.
31. Li, D.; Han, Y.; Wang, C.; Shi, G.; Wang, X.; Li, X.; Leung, V.C. Deep reinforcement learning for cooperative edge caching in future mobile networks. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–6.
32. Wang, C.; Liang, C.; Yu, F.R.; Chen, Q.; Tang, L. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 4924–4938. [CrossRef]
33. Al-Shehari, T.; Alsowail, R.A. An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques. *Entropy* **2021**, *23*, 1258. [CrossRef]
34. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*; PMLR: London, UK, 2018; pp. 1861–1870.
35. Wang, X.; Wang, C.; Li, X.; Leung, V.C.; Taleb, T. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet Things J.* **2020**, *7*, 9441–9455. [CrossRef]
36. ElSawy, H.; Sultan-Salem, A.; Alouini, M.-S.; Win, M.Z. Modeling and analysis of cellular networks using stochastic geometry: A tutorial. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 167–203. [CrossRef]
37. Li, D.; Zhang, H.; Yuan, D.; Zhang, M. Learning-based hierarchical edge caching for cloud-aided heterogeneous networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 1648–1663. [CrossRef]
38. Xu, H.; Sun, Y.; Gao, J.; Guo, J. Intelligent edge content caching: A deep recurrent reinforcement learning method. *Peer-to-Peer Netw. Appl.* **2022**, *15*, 2619–2632. [CrossRef]
39. Zheng, H.; Lin, F.; Feng, X.; Chen, Y. A hybrid deep learning model with attention-based conv-lstm networks for short-term traffic flow prediction. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 6910–6920. [CrossRef]
40. Zhang, L.; Cai, Y.; Huang, H.; Li, A.; Yang, L.; Zhou, C. A cnn-lstm model for soil organic carbon content prediction with long time series of modis-based phenological variables. *Remote Sens.* **2022**, *14*, 4441. [CrossRef]
41. Peters, J.; Schaal, S. Natural actor-critic. *Neurocomputing* **2008**, *71*, 1180–1190. [CrossRef]
42. Li, R.; Zhao, Y.; Wang, C.; Wang, X.; Leung, V.C.; Li, X.; Taleb, T. Edge caching replacement optimization for d2d wireless networks via weighted distributed dqn. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Republic of Korea, 25–28 May 2020; pp. 1–6.