scientific reports

Check for updates

OPEN

Dynamic edge-caching through content popularity and crowd prediction for short video services

Sen Niu[1], Yuhe Liu[1], Kaili Liao[1]✉, Bofeng Zhang[1,2] & Guobing Zou[3]

With the explosive growth of short video traffic and the increasing demand for low-latency content delivery, efficient edge caching strategies have become critical for mobile networks. However, the highly dynamic and personalized characteristics of short video services present substantial challenges for traditional caching approaches, which often depend exclusively on static popularity metrics. This paper proposes DECC (Dynamic Edge-caching through Content Popularity and Crowd Prediction), a novel caching framework that jointly models content popularity and user access behavior to optimize caching decisions at edge nodes. DECC integrates a hybrid deep learning architecture comprising 1D Convolutional Neural Networks (Conv1D), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRU) to capture the temporal dynamics of both video requests and user activity. A fusion mechanism is introduced to generate cache priority scores based on dual-path predictions, enabling more accurate and adaptive content placement. Experimental evaluations conducted on real-world datasets demonstrate that DECC consistently surpasses baseline methods in cache hit rate, access latency reduction, and overall resource utilization efficiency. These results highlight the potential of DECC as a scalable and intelligent caching solution for next-generation short video edge services.

Keywords Edge caching, Short video services, Content popularity prediction, User access forecasting

With the rapid advancement of mobile Internet technologies and the proliferation of smart devices, global mobile data traffic has witnessed unprecedented growth. According to Ericsson's forecast, global monthly mobile data traffic is expected to reach approximately 303 exabytes (EB) by 2030[1]. Among various data services, video applications continue to dominate mobile traffic consumption and are projected to contribute the majority share of total data volume[2,3]. This category encompasses not only conventional long-form videos but also short-form videos, live streaming, and diverse interactive media services. In particular, the explosive growth of short video social platforms, such as Douyin and Kuaishou, has significantly accelerated the expansion of mobile video services[4]. Recent research efforts have begun to investigate the content distribution challenges specific to short video platforms[5,6].

However, the sustained surge in short video content imposes considerable pressure on the core network infrastructure. The traditional cloud-centric computing paradigm is becoming increasingly inadequate in addressing bandwidth consumption and transmission latency challenges. On one hand, centralized processing exacerbates network congestion and leads to increased end-to-end delays; on the other hand, the resulting latency and playback interruptions severely impair user experience[7,8]. To address these issues, edge computing has emerged as a promising architectural paradigm and has attracted extensive research interest[9–11]. By offloading computational and storage workloads to edge nodes located in proximity to end users, edge computing enhances bandwidth utilization efficiency and significantly reduces service latency, thereby improving quality of experience[3,6]. Furthermore, techniques such as popularity-aware caching and latency-sensitive edge deployment have been proposed to further optimize the performance of edge computing systems[12,13].

Within the edge computing paradigm, edge caching plays a pivotal role in enhancing service efficiency.[14] By proactively storing frequently requested content at edge nodes in close proximity to end users, edge caching effectively reduces content retrieval latency and alleviates the traffic load on the core network. In the context of video services, edge video caching has been extensively adopted to prefetch and store popular video content, thereby facilitating fast content delivery and ensuring smooth, high-quality playback experiences[15–17].

[1]School of Computer and Information Engineering, Institute for Artificial Intelligence, Shanghai Polytechnic University, Shanghai 201209, China. [2]School of Computer Science and Technology, Kashi University, Kashi 844000, China. [3]School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China. ✉email: klliao@sspu.edu.cn

Compared to traditional long-form video content, short videos exhibit unique characteristics such as high update frequency, topic diversity, and short content lifespans. These unique attributes demand more adaptive and responsive caching strategies to maintain service quality. [5,18]. To address these challenges, recent studies have proposed popularity-aware prediction models and adaptive content replication techniques specifically designed for short video scenarios[16,19,20].

In recent years, extensive research efforts have been undertaken both domestically and internationally in the field of edge video caching, yielding substantial progress-particularly in enhancing content retrieval efficiency and reducing transmission latency.[21] Among these studies, mainstream edge caching strategies predominantly rely on content popularity prediction, which aims to forecast the likelihood of content being requested by users to inform optimal caching decisions[2,22–24]. These strategies can generally be categorized into two groups: static statistical methods and dynamic prediction methods. The former typically estimate content popularity based on historical request frequency or aggregated user interaction data, and cache the Top-K most frequently accessed items[19]. In contrast, the latter employ techniques such as collaborative filtering, conventional machine learning models, or deep learning approaches (e.g., LSTM) to capture the temporal dynamics of user demand and content popularity, enabling more responsive and accurate caching decisions[25–27]. Despite the practical success of these popularity-based caching schemes, they often fail to account for personalized user preferences and the complex, multi-dimensional correlations among content items, which limits their caching effectiveness and leaves room for further enhancement[3,28,29].

Despite notable advancements in video caching research, several critical challenges remain inadequately addressed. First, the majority of existing methods have been designed with long-form video content in mind. In contrast, short videos-distinguished by their brief duration, high request frequency, and rapid content turnover-exhibit distinct spatiotemporal characteristics that undermine the effectiveness of conventional caching mechanisms. As a result, cache hit rates are often diminished, and system resource utilization becomes suboptimal. Second, many current strategies rely on static caching mechanisms, where decisions are made based on fixed content popularity rankings or historical access logs. These approaches are inherently limited in their ability to capture the temporal dynamics of user demand, leading to delayed responsiveness when content popularity shifts rapidly. Lastly, most prior works have predominantly emphasized content-side popularity modeling, while overlooking demand-side characteristics-such as periodic user access patterns and spatial locality of requests across edge servers. These factors are essential for improving the granularity, precision, and regional adaptability of edge caching strategies. Although some recent studies have attempted to incorporate user behavior prediction into edge caching decisions, such as leveraging collaborative filtering, clustering-based user modeling, or reinforcement learning to infer access intentions, these methods often treat user behavior separately from content popularity. As a result, they fail to jointly capture the interaction between dynamic user access patterns and content-level popularity variations, which is particularly critical in short video environments.

To address the aforementioned challenges, this paper proposes a dynamic collaborative edge caching approach for short video services, termed the DECC model (Dynamic Edge Caching through Content Popularity and Crowd Prediction). The proposed framework simultaneously incorporates predictions of content popularity and user access trends to optimize caching decisions in dynamic environments. Unlike prior user-behavior-aware caching frameworks that model user mobility or access probability as an isolated factor, DECC introduces a joint prediction mechanism that integrates both content popularity dynamics and collective user crowd behavior into a unified deep learning architecture. Specifically, DECC employs a hybrid deep learning architecture that integrates one-dimensional Convolutional Neural Networks (1D-CNN) and Long Short-Term Memory (LSTM) networks to model the historical request sequences of short videos and estimate their future access probabilities. In parallel, a lightweight Gated Recurrent Unit (GRU) network is utilized to capture the periodic patterns in user access behavior and forecast the anticipated user volume at edge nodes. By jointly fusing the prediction outputs from these two branches, the model computes a unified cache priority score, which guides a collaborative caching strategy that is both content-aware and crowd-adaptive. This joint modeling enables the system to dynamically align caching decisions with both evolving content trends and user access distributions, thereby overcoming the decoupling problem observed in earlier popularity- or behavior-only approaches. Compared to conventional single-dimensional prediction methods, extensive experiments demonstrate that DECC significantly improves cache hit rate, reduces caching cost and transmission latency, and enhances the overall cost-efficiency of delay reduction.

The main contributions of this paper are summarized as follows:

- A dynamic collaborative edge caching framework (DECC) is proposed, which integrates convolutional and recurrent neural networks to jointly predict both content popularity and user access trends. This dual-perspective prediction enables enhanced cache hit rates, reduced service latency, and improved cost-efficiency in edge resource utilization.
- A hybrid content popularity prediction model is developed by combining one-dimensional convolutional layers with Long Short-Term Memory (LSTM) networks to estimate the future access probabilities of short video content. Simultaneously, a lightweight Gated Recurrent Unit (GRU)-based module is designed to capture periodic user access patterns and forecast user volume at edge nodes, thereby facilitating adaptive resource allocation.
- Extensive experiments are conducted on real-world short video access datasets across multiple edge nodes. The proposed method is comprehensively evaluated using four key performance metrics: cache hit rate, access latency, caching cost, and latency reduction per unit cost, demonstrating its superiority over baseline methods.

The remainder of this paper is organized as follows: the **Problem Formulation** section introduces the short video edge caching problem, along with the relevant notations and system constraints. The **Approach** section presents the overall architecture of the proposed DECC model and explains the joint mechanism for content popularity and user volume prediction. The **Experimental Evaluation** section details the experimental setup and evaluation metrics, and demonstrates the effectiveness of the proposed model through comparative results. The **Related Work** section discusses existing approaches and highlights the contributions of this study. Finally, the **Conclusion** section summarizes the findings and outlines future research directions.

Problem formulation

In this section, we present the foundational assumptions and conduct a formal problem analysis to inform the design of the proposed model and the selection of appropriate prediction algorithms. We assume that user viewing behaviors of short video content display distinct temporal and spatial patterns, which can be effectively captured through historical access records. Additionally, given the constrained storage capacity of edge nodes, only a limited subset of historically popular content can be selected for caching at each node. Based on these assumptions, we proceed to formulate the model architecture and adopt suitable predictive techniques for content and user demand forecasting. To facilitate the subsequent model formulation, we formally define key concepts related to edge video servers, cache capacity constraints, user volume distributions, and content popularity estimation. A summary of the main symbols and their corresponding definitions used throughout this paper is provided in Table 1.

Definition 1 *(Edge Server Entities)* The set of edge servers is denoted as $E = \{e_1, e_2, e_3, \ldots, e_n\}$. Each edge server e_i possesses a set of fixed physical and operational attributes, including its geographic location, coverage radius R_{e_i}, storage capacity S_{e_i}, and maximum request-handling capacity C. These parameters collectively determine the edge server's spatial service scope and its ability to respond to video access requests.

The set of short video resources cached by edge server e_i during time interval t_k is denoted as $CF_{e_i}^{t_k}$. The cached content $CF_{e_i}^{t_k}$ is selected from the global video category set V according to caching type, quantity, and video priority level-typically giving preference to popular or frequently accessed videos to improve user Quality of Experience.

However, the total required storage for caching must not exceed the edge server's capacity Size_{e_i}, as constrained by Equation 1.

$$\sum_{i=1}^{n} CF_{e_i}^{t_k} \leq \mathrm{Size}_{e_i} \tag{1}$$

Symbol	Meaning
$E = \{e_1, e_2, \ldots, e_n\}$	Set of edge servers, each with attributes such as geographic location, coverage radius, and caching capacity.
$U = \{U_{e_1}, U_{e_2}, \ldots, U_{e_n}\}$	Set of users served by each edge server.
$T = \{t_1, t_2, \ldots, t_l\}$	Set of discrete time slots.
$V = \{v_1, v_2, \ldots, v_m\}$	Set of short video categories.
$CF_{e_i}^{t_k}$	Set of short video contents cached by edge server e_i during time slot t_k, determined by the caching decision module.
$M = \langle E, T, V \rangle$	The video caching system model.
$r = \langle e_i, t_k, v_j, U_{e_i}^{t_k, v_j} \rangle$	Video request record at time t_k for category v_j on e_i.
$\lvert U \rvert = \{ \lvert u \rvert_1^{t_1}, \lvert u \rvert_2^{t_2}, \ldots, \lvert u \rvert_m^{t_k} \}$	Total number of users participating in access activities.
$r_{i,j,k} = \lvert U_{e_i}^{t_k, v_j} \rvert$	At time t_k, the number of users accessing video category v_j on edge server e_i.
$ESVC = \langle M, t, v, \lvert u \rvert \rangle$	State of the short video edge caching system.
$(t+1,\ \hat{v},\ \lvert\hat{u}\rvert)$	Prediction target: $\hat{v}$ is the predicted video category, and $\lvert\hat{u}\rvert$ is the predicted user count at $t+1$.
$\Omega = \langle M, v, \lvert u \rvert \rangle$	Input variables for prediction: system, content, and user statistics.
$\mathrm{Req}_{e_i}^{t}$	Set of user-requested videos at edge node e_i during time t.
$\mathrm{Cost}(v_k)$	Unit cost of caching video v_k on the edge server.
$D_{e_i}^{\mathrm{cache}}$	Total access latency for edge node e_i after caching.
$D_{e_i}^{\mathrm{original}}$	Total access latency for edge node e_i without caching.
h_L, h_G	Hidden dimensions of the LSTM and GRU networks, respectively.
C	Cache capacity constraint of each edge node.
η	Latency reduction per unit caching cost (efficiency metric).

Table 1. Main symbols and their meanings in the DECC model and evaluation metrics.

This constraint ensures that the edge server performs caching within its limited resources, preventing it from exceeding its physical capacity limit, thereby guaranteeing the feasibility and stability of the caching strategy. It also implicitly reflects the balance between storage allocation and service quality under real-world resource constraints.

This definition establishes the structural foundation of the system, describing how edge servers operate as distributed caching units with bounded capacity. In the next definition, we extend this framework by introducing the set of users associated with each edge server, thereby linking the physical infrastructure to the dynamic access behaviors that drive caching demand.

Definition 2 *(Edge User Association)* The set of users is defined as $U = \{U_{e_1}, U_{e_2}, U_{e_3}, \ldots, U_{e_n}\}$, where U_{e_i} represents the users accessing edge server e_i across n time slots, and $U_{e_i} = \{U_{e_i}^{t_1}, U_{e_i}^{t_2}, U_{e_i}^{t_3}, \ldots, U_{e_i}^{t_n}\}$, with $U_{e_i}^{t_k}$ denoting the set of users accessing edge server e_i at time t_k. It is evident that $U_{e_i}^{t_k} \subseteq U_{e_i} \subseteq U$. Users in $U_{e_i}^{t_k}$ are located within the coverage radius of edge server e_i, i.e., $d(U_{e_i}^{t_k}, e_i) < R_{e_i}$. At the same time, each user can only access one edge server at any given time, meaning $U_{e_i}^{t_k} \cap U_{e_k}^{t_k} = \emptyset$ for $i \neq k$. To reflect the resource limitations of edge servers, the number of users connected to edge server e_i at time t_k must not exceed its maximum request capacity C, i.e., $|U_{e_i}^{t_k}| \leq C$, in order to ensure service quality and prevent server overload.

This definition characterizes the dynamic interaction between users and edge servers, defining spatial and temporal constraints on user connections. It establishes how user distributions evolve within the coverage area of each edge node, thereby linking the static server capacities described in Definition 1 to real-time service demand. Building upon this association, the next definition formalizes user access behaviors in terms of video request records, which quantify how users interact with specific video categories across time and servers.

Definition 3 *(Video Access Records)* Let the video caching system be defined as $M = \langle E, T, V \rangle$, where $E = \{e_1, e_2, \ldots, e_k\}$ denotes the set of edge servers, $T = \{t_1, t_2, \ldots, t_l\}$ denotes the set of discrete time slots, and $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of video content categories.

This definition provides a system-level abstraction that connects the physical infrastructure (edge servers) and temporal dynamics (time slots) with content-level entities (video categories). It forms the foundation for representing user access activities as structured data objects.

A single video request record is defined as a quadruple

$$r = \langle e_i, t_k, v_j, U_{e_i}^{t_k, v_j} \rangle \tag{2}$$

where $e_i \in E$ represents the edge server, $t_k \in T$ is the time slot, $v_j \in V$ is the video category, and $U_{e_i}^{t_k, v_j}$ is the set of users who accessed video category v_j on edge server e_i at time t_k.

In this formulation, each record encapsulates both spatial (server) and temporal (time) dimensions, capturing how user activities interact with content distribution at the edge.

For each edge server e_i, the maximum number of users it can serve at any given time is constrained by its capacity C_{e_i}, thus the following condition must be satisfied:

$$|U_{e_i}^{t_k, v_j}| \leq C_{e_i} \tag{3}$$

This constraint ensures that the access demand recorded in $U_{e_i}^{t_k, v_j}$ remains feasible within the physical limitations of each server, maintaining a consistent mapping between network load and service capability.

For a given video category v_j on edge server e_i, the total number of users participating in access over time is defined as:

$$|U| = \{|u|_1^{t_1}, |u|_2^{t_2}, \ldots, |u|_m^{t_k}\} \tag{4}$$

where $|u|_m^{t_k}$ denotes the number of users accessing at time t_k. By aggregating all video request records, a 3-dimensional request tensor $R \in \mathbb{R}^{|E| \times |V| \times |T|}$ can be constructed, where each element $r_{i,j,k} = |U_{e_i}^{t_k, v_j}|$ represents the number of users accessing video category v_j on edge server e_i at time t_k, as illustrated in Fig. 1. The request tensor provides a unified representation of user content time interactions, enabling a data-driven understanding of how popularity evolves across space and time.

This tensor can be unfolded along the content or time dimension to support tasks such as video popularity analysis, user access prediction, and cache decision optimization. Building upon this representation, the next definition formalizes the time-aware video access prediction problem, which aims to infer future user demand patterns and guide dynamic caching decisions based on the observed request tensor.

Definition 4 *(Time-aware Video Access Prediction Problem)* Building upon the request tensor representation introduced in Definition 3, we now formalize the temporal prediction problem that underpins adaptive edge caching decisions. Given the current state of a short video edge caching system, it is represented as a quadruple

$$\mathrm{ESVC} = \langle M, t, v, |u| \rangle \tag{5}$$

where $M = \langle E, V, T \rangle$ denotes the structural model of the edge video access system, including the set of edge nodes E, the set of discrete time slots T, and the set of video content categories V; $t \in T$ represents the current time slot; $|u|$ denotes the number of user accesses at time t; and $v \in V$ represents the video content category

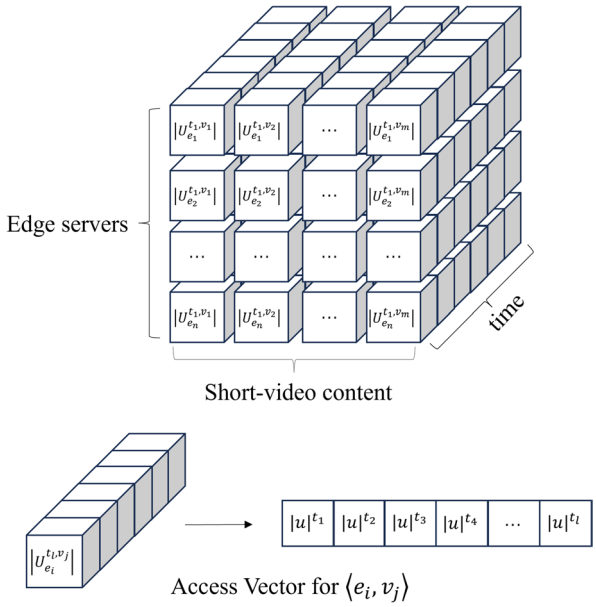

Figure 1. Video access tensor structure. The figure illustrates a three-dimensional request tensor composed of edge servers, video categories, and time slots. Each cube cell represents the number of users accessing a specific video category v_j on edge server e_i at time slot $t_{(k,r)}$, denoted as $|U_{e_i}^{t_{(k,r)},v_j}|$. The lower part of the figure shows a user access vector sequence unfolded along the time dimension, which can be leveraged for popularity prediction and cache optimization strategy modeling.

being accessed at the current time. Here, the quadruple ESVC serves as a compact state representation that captures both temporal and semantic aspects of the caching environment–linking the current content demand $(v, |u|)$ with the structural topology of the edge system (M). This formalization enables predictive modeling of user access behaviors over time.

The time-aware video access prediction problem aims to forecast the short video content that will be cached on edge servers at the next time slot $t+1$, based on the current state ESVC. Specifically, the prediction targets are defined as $(t+1,\ \hat{v},\ \hat{|u|})$, where $\hat{v}$ denotes the predicted video category to be cached and $\hat{|u|}$ represents the expected user access volume. These predictions guide subsequent optimization processes that jointly minimize latency and caching cost in the linear programming formulation introduced in the next section.

Approach

Overall architecture of the DECC model

To facilitate efficient prediction and scheduling of short video content for edge caching, this paper proposes a collaborative prediction framework named DECC (Dynamic Edge Caching through Content Popularity and Crowd Prediction). The DECC model jointly leverages two complementary perspectives: one prediction pathway captures the temporal dynamics of content popularity by modeling short video access sequences, while the other estimates user access volume at edge nodes by learning from historical behavioral patterns. These two predictive components are integrated through a coordination mechanism that fuses their outputs, thereby producing unified and adaptive caching decisions. The resulting cache prioritization strategy effectively balances content demand and user density. The overall architecture of the DECC model is illustrated in Fig. 2.

In the content prediction pathway, the model takes as input the historical video request sequences and associated user access statistics collected from edge servers. A one-dimensional convolutional neural network (Conv1D) is first employed to extract local temporal features from the input time series. These features are subsequently fed into a Long Short-Term Memory (LSTM) network to capture long-range temporal dependencies, enabling the estimation of content access probabilities for the upcoming time interval. By incorporating a multi-layer convolutional structure along with the Sigmoid activation function, the model effectively captures nonlinear trends and latent periodic patterns in user interest dynamics. The output is a set of content priority scores, which serve as critical indicators for optimizing cache placement strategies.

In the user access prediction pathway, the model incorporates a Gated Recurrent Unit (GRU) to construct a lightweight forecasting module for predicting user connection volumes at edge nodes. This component is designed to estimate the expected level of user activity in future time slots, with particular emphasis on identifying low-traffic scenarios-such as "zero access" cases-in order to avoid inefficient cache allocation to underutilized nodes. The GRU-based module learns temporal features from historical user request sequences and captures access trends along with periodic fluctuations. The resulting output consists of predicted user connection volumes for each edge node in the next time interval, which provides a demand-aware signal to guide cache resource allocation.

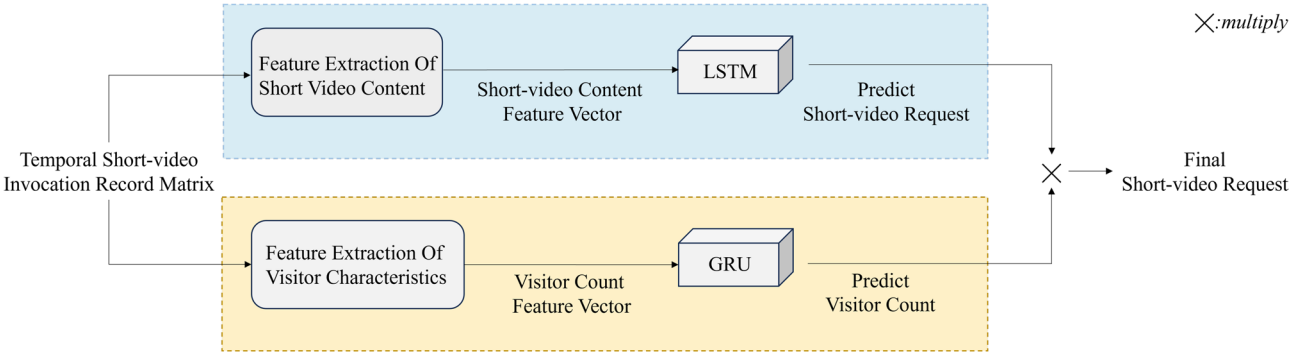

Figure 2. Overall architecture of the DECC model.

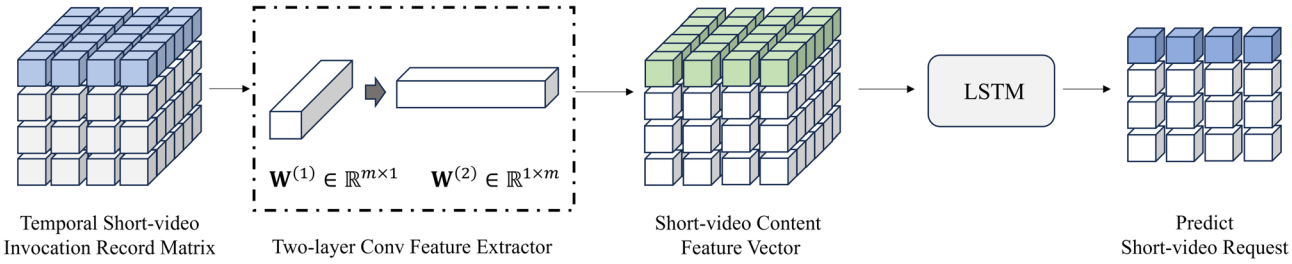

Figure 3. Structure of the short video content prediction module.

To enable comprehensive utilization of the dual prediction outputs, the DECC model incorporates a fusion mechanism at the output stage. This mechanism integrates the outputs from both the content popularity and user access prediction pathways through a fully connected layer, which jointly processes the predicted content access probabilities and estimated user connection volumes. The fused representation is then used to compute the final cache priority scores, which serve as the basis for optimizing caching decisions across edge nodes.

Short video content prediction

In the content prediction pathway, the DECC model is designed to extract latent temporal features from historical short video request sequences and estimate future content popularity, thereby guiding cache placement decisions at edge nodes. This module primarily consists of a one-dimensional convolutional layer (Conv1D) for capturing local temporal patterns, followed by a Long Short-Term Memory (LSTM) network to model long-range dependencies across time steps. The overall structure of the short video content prediction module is depicted in Fig. 3.

Short video content feature extraction

To effectively capture the dynamic patterns of short video content over time, the content prediction pathway in the DECC model incorporates a one-dimensional convolutional neural network (Conv1D) to extract local temporal features from historical video request sequences. Rather than relying on raw access data alone, the convolutional architecture enables the model to perceive fine-grained variations in content access frequency within fixed temporal windows-such as short-term spikes, gradual increases, and periodic fluctuations-thus facilitating the construction of more discriminative content representation vectors. These learned representations provide robust input features for subsequent popularity modeling and cache decision-making. The Conv1D module processes time-series data collected from edge servers, with the temporal distribution of content categories serving as the main input feature. By applying two layers of convolutional operations, the module captures both trend evolution and contextual dependencies within localized time segments, enabling deep modeling of short video content popularity dynamics.

The content prediction pathway leverages historical short video request records to construct temporal input sequences, with the objective of uncovering time-sensitive patterns in user access behavior and content demand dynamics.

For any given edge node e_i, its corresponding access sequence can be extracted from the global request records r, denoted as:

$$r_{e_i} = \langle t_k, v_j, U_{e_i}^{t_k, v_j} \rangle \tag{6}$$

which represents the user access behavior for video category v_j on edge node e_i at time t_k, along with the number of users $|U_{e_i}^{t_k, v_j}|$ who accessed that category.

Over a time window $[t_1, t_k]$, the complete access sequence of edge node e_i is represented as:

$$|U| = \{|u|_1, |u|_2, \ldots, |u|_k\}, \quad |u|_t = |U_{e_i}^{t,v_j}| \tag{7}$$

By combining video category information with temporal access statistics, the input vector at each time step is constructed as:

$$x_t = [v_j^t, |u|_t], \quad x_t \in \mathbb{R}^d \tag{8}$$

The input sequence $\{x_1, x_2, \ldots, x_T\}$ is fed into a two-layer one-dimensional convolutional neural network (Conv1D) to extract local temporal features. The first layer focuses on capturing fine-grained variations such as periodicity and sudden bursts, while the second layer extracts higher-level abstract features and enhances the contextual representation capability.

The computation of the one-dimensional convolution is formulated as follows:

$$z_t^{(i)} = \sum_{l=0}^{k-1} W_l^{(i)} \cdot x_{t+l} + b^{(i)} \tag{9}$$

$$u_t^{(i)} = \sigma(z_t^{(i)}) = \frac{1}{1 + e^{-z_t^{(i)}}} \tag{10}$$

where $i = \{1, 2\}$, $W^{(1)} \in \mathbb{R}^{m \times 1}$, $W^{(2)} \in \mathbb{R}^{1 \times m}$ are the one-dimensional convolution kernels in the first and second convolutional layers, respectively; $b^{(i)}$ denotes the bias term; and $\sigma(\cdot)$ represents the Sigmoid activation function.

After the Conv1D module completes feature extraction, the model outputs a content feature vector at each time step t, denoted as v_t, which captures the local temporal pattern of short video request behaviors at that time slot. This feature vector integrates the access dynamics across multiple time steps within the convolution window, effectively capturing sudden surges, fluctuations, and periodic trends in request frequency. It is represented as:

$$v_t = [v_t^{(1)}, v_t^{(2)}, \ldots, v_t^{(d)}] \in \mathbb{R}^d \tag{11}$$

where v_t is the content feature vector at time step t, d denotes the number of convolution channels (i.e., the output feature dimension of the layer), and $v_t^{(i)}$ represents the feature value extracted from the i-th channel.

The generated feature vector sequence is subsequently processed by the LSTM module, which captures the long-term evolution of user access patterns across content categories. This enables the construction of semantically rich representations that support accurate estimation of content access probabilities in the forthcoming time slot.

Short video content prediction

In Section A, the model extracts local temporal features from video request sequences using a Conv1D convolutional module, resulting in video category feature vectors v_t. Building on this, this section further applies a Long Short-Term Memory (LSTM) network to model the temporal dynamics of the extracted content features.

In the proposed model, the time-step feature vectors v_t produced by the convolutional layers are fed into an LSTM network to capture long-term behavioral patterns in video request sequences. The LSTM processes the input sequentially using a series of gating mechanisms, effectively addressing the gradient vanishing problem encountered in traditional RNNs when modeling long sequences.

First, the forget gate determines which parts of the historical content access state should be retained or discarded, based on the current input feature vector v_t and the previous hidden state h_{t-1}. The calculation formula is:

$$f_t = \sigma(W_f \cdot [h_{t-1}, v_t] + b_f) \tag{12}$$

Next, the input gate filters the valuable information from the current content feature vector and combines it with the candidate memory state $\tilde{C}_t$ to compute the new content popularity representation:

$$i_t = \sigma(W_i \cdot [h_{t-1}, v_t] + b_i) \tag{13}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, v_t] + b_C) \tag{14}$$

Subsequently, the model updates the cell state by combining the previous memory cell state C_{t-1}, regulated by the forget gate, with the candidate memory $\tilde{C}_t$, filtered by the input gate. This results in the updated cell state:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{15}$$

Then, the output gate determines the current output representation based on the content feature vector and the updated internal memory state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, v_t] + b_o) \tag{16}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{17}$$

Finally, based on the LSTM output at the last time step h_T, the model predicts the video request state for the next time slot $t+1$. The prediction is formulated as:

$$\hat{y}_{t+1}^{(v)} = \text{softmax}(W_y \cdot h_T + b_y) \tag{18}$$

Here, $\hat{y}_{t+1}^{(v)}$ denotes the estimated access probability for each video category at time $t+1$. The prediction results serve as content popularity scores, which are further fed into the cache priority scheduling module to guide the generation and updating of cache lists on edge servers, thereby enabling accurate content distribution and optimized utilization of edge resources.

User access volume prediction

In the user access prediction pathway, the DECC model aims to capture the temporal characteristics of user connection sequences, with a particular focus on identifying periodic variation patterns. The architecture of the user access prediction module is illustrated in Fig. 4. This module consists of a one-dimensional convolutional layer (Conv1D) and a Gated Recurrent Unit (GRU), which are responsible for extracting localized fluctuation features and modeling long-term access trends, respectively. The Conv1D layer detects fine-grained access dynamics-such as abrupt spikes and cyclical patterns-while the GRU network forecasts the future trajectory of user connection volumes. Importantly, the model explicitly accounts for low-traffic or idle scenarios: if the predicted number of user connections for a given edge node is zero or falls below a predefined threshold, no caching resources will be allocated to that node. This selective allocation mechanism enhances the overall efficiency of edge resource utilization by preventing unnecessary cache deployments at underutilized nodes.

Feature extraction of user access patterns

To effectively model the temporal dynamics of user access volume, a one-dimensional convolutional neural network (Conv1D) is employed in the user access prediction pathway to extract meaningful features from the access count sequences. Unlike directly feeding raw connection counts into the prediction model, the convolutional structure enables the extraction of fine-grained variations in access intensity within localized temporal windows-such as transient spikes, recurring fluctuations, and periodic trends. These learned representations exhibit stronger temporal expressiveness and provide informative inputs for the downstream GRU-based forecasting module.

This module operates on the time series of user access volumes recorded at edge nodes, with input sequences constructed by incorporating auxiliary statistical features such as rate of change and periodicity. The convolutional layer extracts stable behavioral patterns from these enriched input sequences, enabling robust modeling of user access dynamics. The extracted features are subsequently used to predict the number of user connections in future time slots, thereby supporting the dynamic optimization of edge caching strategies and service scheduling decisions.

The user access prediction module focuses on modeling the variation trend of user connection counts at edge nodes for future time slots. For any given edge node e_i, the model aggregates the access counts across all video categories based on the request records, forming a node-level user access sequence:

$$x_t = [v_j^t, |u|_t], \quad x_t \in \mathbb{R}^d \tag{19}$$

The resulting input sequence $\{x_1, x_2, \ldots, x_T\}$ is fed into a one-dimensional convolutional layer, whose operations are defined as:

$$z_t' = \sum_{l=0}^{k-1} W_l' \cdot x_{t+l} + b' \tag{20}$$

$$u_t = \sigma(z_t') = \frac{1}{1+e^{-z_t'}} \tag{21}$$

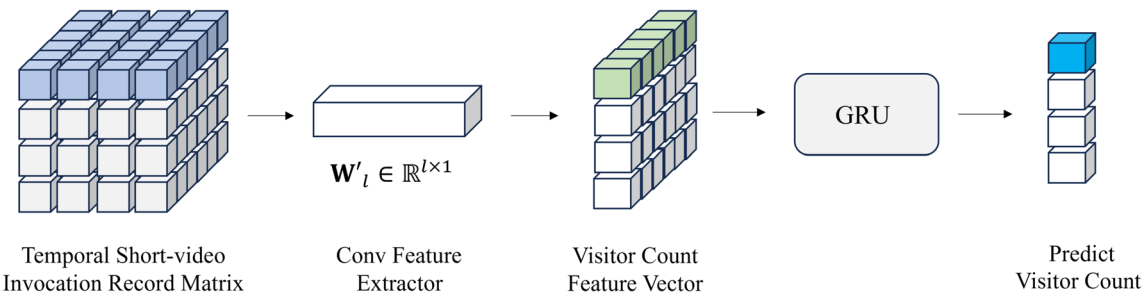

Figure 4. Structure of the user access prediction module.

where the convolution kernel $W_l' \in \mathbb{R}^{l\times 1}$, d is the feature dimension of each time step input, and b' is the bias term.

After being processed by the 1D convolutional network, the model outputs a user access feature vector u_t at each time step t, which represents the connection state of the edge node at that time. The vector is formulated as:

$$u_t = [u_t^{(1)}, u_t^{(2)}, \ldots, u_t^{(d)}] \in \mathbb{R}^d \tag{22}$$

where u_t is the user access feature vector at time step t, d denotes the number of output channels (i.e., feature dimensions) of the convolutional layer, and $u_t^{(i)}$ represents the feature component extracted by the i-th convolutional channel.

User access prediction

To improve the accuracy of user-aware caching strategies, the DECC model employs a compact and computationally efficient Gated Recurrent Unit (GRU) in the user access prediction pathway to model the temporal evolution of user connection counts at edge nodes. Compared to the Long Short-Term Memory (LSTM) network, GRU simplifies the architecture by merging the memory cell and hidden state, resulting in a reduced number of parameters and lower computational overhead. This makes GRU particularly well-suited for modeling user access sequences at edge nodes, which are often highly volatile yet structurally less complex.

The GRU module receives the user access feature vector u_t as input. In the user access prediction task, the DECC model adopts the GRU architecture in place of the more parameter-intensive LSTM, as GRU effectively alleviates gradient vanishing issues in long-term dependency modeling while providing faster training convergence. These advantages make GRU particularly well-suited for forecasting user connection sequences at edge nodes, which typically exhibit high temporal variability yet follow relatively simple underlying patterns.

The core modeling objective is formalized as:

$$P_{t+1}^{(u\text{-}out)} = g(X_{t\text{-}step}^{(u\text{-}out)}, \ldots, X_t^{(u\text{-}out)}) \tag{23}$$

Here, $g(\cdot)$ denotes the GRU layer, and the input sequence $[X_{t\text{-}step}^{(u\text{-}out)}, \ldots, X_t^{(u\text{-}out)}]$ represents the user access data across a time window of length *step*.

The GRU first receives the current time step's user access feature vector u_t, along with the hidden state from the previous time step h_{t-1}, to compute two gating variables. The update gate z_t determines how much information from the previous state should be retained in the current unit:

$$z_t = \sigma(W_z \cdot [h_{t-1}, u_t] + b_z) \tag{24}$$

where σ is the Sigmoid activation function. A value closer to 1 indicates stronger retention of the previous state.

Next, the reset gate r_t controls the extent to which the previous hidden state contributes to the current candidate state:

$$r_t = \sigma(W_r \cdot [h_{t-1}, u_t] + b_r) \tag{25}$$

When $r_t \approx 0$, the GRU forgets most of the old state and relies mainly on the current input; when $r_t \approx 1$, it retains more historical information.

Under the joint regulation of the update and reset gates, the model computes the candidate hidden state $\tilde{h}_t$, representing a fused estimate of the current input and historical memory:

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, u_t] + b_h) \tag{26}$$

where $*$ denotes the Hadamard (element-wise) product.

Finally, the hidden state is updated by combining the previous state and the candidate state, controlled by the update gate:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{27}$$

This mechanism enables the GRU to dynamically balance historical context with current observations, thereby facilitating accurate prediction of user connection volumes at the next time step.

The final hidden state h_t is passed through a fully connected mapping function $f_{\text{fc}}(\cdot)$ to generate the predicted user access count:

$$\hat{u}_{e_i}^{(t+1)} = f_{\text{fc}}(h_t) \tag{28}$$

Here, $\hat{u}_{e_i}^{(t+1)}$ denotes the predicted number of user connections for edge node e_i at time $t+1$. If the predicted value remains zero or below a predefined threshold over consecutive time slots, the corresponding node is identified as inefficient, and caching resources will no longer be allocated to it.

This strategy enhances resource utilization by preventing unnecessary cache deployment on underutilized edge nodes. The GRU-based user access prediction module not only quantifies user demand density but also captures the spatial distribution of access loads across the edge infrastructure. In conjunction with the content

popularity prediction pathway, it constitutes a core component of the cache priority scoring mechanism, enabling fine-grained, demand-aware allocation of edge resources.

Fusion and output of prediction results

The output layer of the DECC model consists of two components: a Fully Connected (FC) layer and a result fusion layer. The detailed network architecture is shown in Fig. 5.

The output layer of the DECC model consists of a Fully Connected (FC) layer followed by a result fusion layer, as illustrated in Fig. 5. The FC layer connects all nodes from the preceding recurrent neural network (RNN) layers to the output space, computing the final outputs through weighted summation and bias operations. Each of the two prediction branches-content popularity prediction and user access forecasting-produces an independent output, which is subsequently mapped to its respective representation space. Specifically, the two RNN-based pathways are responsible for predicting the future short video request distribution and the expected user access volume at edge nodes, respectively.

In the result fusion layer, the outputs from both prediction pathways are integrated to jointly inform the final caching decision. This fusion mechanism leverages the complementary strengths and representational capacities of each prediction stream, thereby reducing the risk of overfitting associated with isolated models and enhancing the overall generalization capability. As a result, the model achieves more comprehensive and accurate caching priority predictions. The fusion output is computed as:

$$\hat{o}_{e_i}^{(t+1)} = \sigma\left(W_o \cdot \left[\hat{y}_{t+1}^{(v)}, \hat{u}_{e_i}^{(t+1)}\right] + b_o\right) \tag{29}$$

Here, $\hat{o}_{e_i}^{(t+1)}$ denotes the fused prediction output for edge node e_i at time $t+1$; $\hat{y}_{t+1}^{(v)}$ is the predicted access probability of video category v from the LSTM path; $\hat{u}_{e_i}^{(t+1)}$ is the predicted user access volume from the GRU path for node e_i; $[\,,\,]$ represents the vector concatenation operation; W_o and b_o are the weights and bias of the fully connected layer; and $\sigma(\cdot)$ denotes the activation function, such as Sigmoid.

Both $\hat{y}_{t+1}^{(v)}$ and $\hat{u}_{e_i}^{(t+1)}$ are outputs from the upper-level recurrent neural networks. This fusion strategy substantially mitigates the risk of overfitting and improves the model's generalization performance, thereby enabling the system to sustain a high cache hit rate while simultaneously controlling access latency and resource consumption.

Algorithm implementation and computational complexity analysis

To provide a clear implementation-level understanding of the DECC framework, the following pseudocode summarizes the complete workflow described in the previous subsections. It corresponds to the mathematical formulations defined in Eqs. (1)–(29), which represent the operations of Conv1D-based feature extraction, LSTM- and GRU-based temporal modeling, and the fusion layer for final caching decision generation.

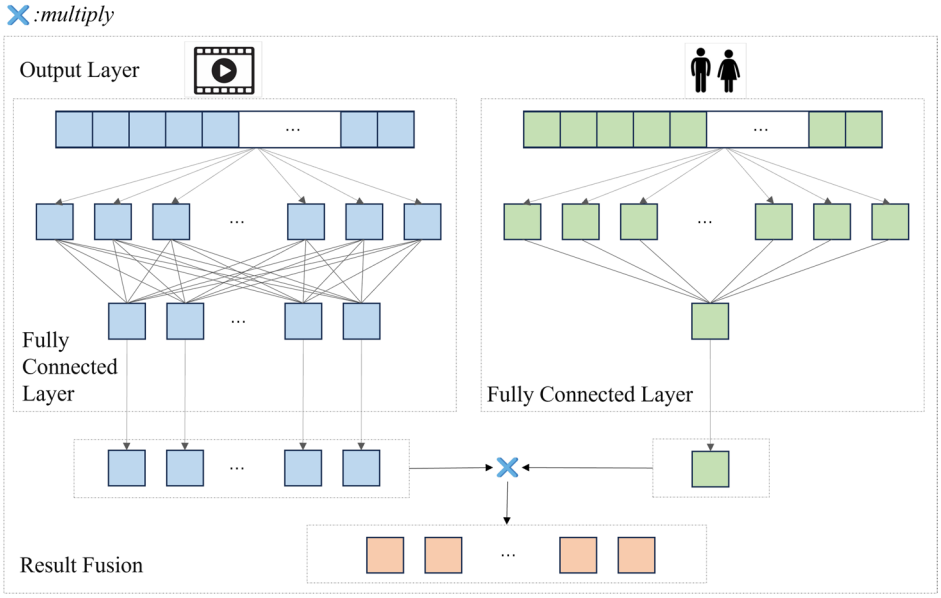

Figure 5. Output layer network architecture.

Input : Historical short-video access records $V = \{v_1, v_2, \ldots, v_T\}$;
Historical visitor statistics $U = \{u_1, u_2, \ldots, u_T\}$;
Cache-capacity constraint C.
Output : Optimized cache-allocation sets $CF_{e_i}^{t_k}$ for all $e_i \in E$.
Step 1: Feature Extraction (Eqs. 6-11, 19-22)
Extract content-level feature vectors F_v from short-video metadata and access sequences using Conv1D filters.
Extract visitor-level feature vectors F_u from user connection sequences through temporal convolution.
Step 2: Popularity and User Forecasting (Eqs. 12-18, 23-28)
LSTM (content popularity): $\hat{P}_v(t) = \mathrm{LSTM}(F_v; \theta_{\mathrm{LSTM}})$
GRU (user access volume): $\hat{U}_{e_i}(t) = \mathrm{GRU}(F_u; \theta_{\mathrm{GRU}})$
Step 3: Fusion and Scoring (Eq. 29)
Fuse predictions via a fully connected fusion layer:
$S_{v,e_i}(t) = \sigma\big(W_o[\hat{P}_v(t), \hat{U}_{e_i}(t)] + b_o\big)$
Rank videos by $S_{v,e_i}(t)$ under capacity constraint C to obtain:
$CF_{e_i}^{t_k} = \mathrm{SelectTopK}(S_{v,e_i}(t), C)$

Algorithm 1. Pseudocode of the DECC Caching Strategy

Computational complexity analysis

The computational complexity of the DECC model primarily stems from three core stages: the dual neural prediction modules (LSTM and GRU) and the cache decision process. For a dataset comprising T time steps, N edge nodes, and M candidate videos, the LSTM-based content prediction requires matrix operations of order h_L^2 at each time step, where h_L denotes the hidden dimension of the LSTM, leading to an overall complexity of $O(T \cdot h_L^2)$. Similarly, the GRU-based user access forecasting module performs sequential updates with hidden dimension h_G, resulting in a computational cost of $O(T \cdot h_G^2)$. In the fusion and cache-decision phase, the fully connected fusion layer introduces $O(h_L + h_G)$ operations, while the ranking and Top-k selection under the cache capacity constraint C contribute an additional $O(M \log M)$ complexity.

By aggregating these components, the total time complexity of the DECC model can be approximated as:

$$O\big(T(h_L^2 + h_G^2) + M \log M\big) \tag{30}$$

which remains computationally tractable for large-scale edge caching scenarios. The memory footprint of DECC is mainly determined by the learnable parameters and intermediate feature representations within the recurrent modules, resulting in a space complexity of:

$$O(h_L + h_G + M) \tag{31}$$

This demonstrates that DECC achieves efficient memory utilization while maintaining scalability across multiple edge nodes and a large set of candidate videos.

Experimental evaluation

Experimental setup and dataset

To evaluate the performance of the proposed caching algorithm, we conducted a comprehensive experimental assessment to examine its effectiveness and feasibility. All experiments were performed on a system equipped with an Intel dual-core processor and 16 GB of RAM to ensure reproducibility and consistency across evaluations. The algorithms were implemented in Python version 3.8, ensuring consistency across all comparative evaluations.

The experiments employed the Shanghai Telecom dataset, a benchmark used widely in edge computing research. This dataset contains detailed information including base station locations, anonymized user identifiers, as well as user access start and end times. An overview of the number and spatial distribution of base stations is provided in Fig. 6. The dataset spans a six-month period and comprises over 7.2 million records, capturing the internet access behaviors of 9,481 mobile devices across 3,233 base stations.

The Douyin dataset records information such as user ID, user city, video ID, and video category. In the Telecom dataset, since user connection times are irregular, the data was resampled at 1-hour intervals, and the number of user connections per base station per time slot was recalculated.

Since the Telecom and Douyin datasets are independent, a user ID mapping scheme was constructed to establish a one-to-one correspondence between users in both datasets. The resulting merged dataset spans a six-month period and includes approximately 400,000 short video request records distributed across 10 edge base stations. For evaluation, the final 168 hours of data were designated as the validation set, while all preceding records were used for model training. At inference time, the model takes user access logs from the previous 168 hours at each edge node as input and predicts the short video content likely to be requested in the next 1-hour time window.In the simulation environment, the cache capacity of each edge node is set to 20. The network delay from the edge node to the cloud is configured as 2 hops, while the delay from users to their associated edge node is set to 1 hop. The caching cost per video resource is uniformly set to 1.

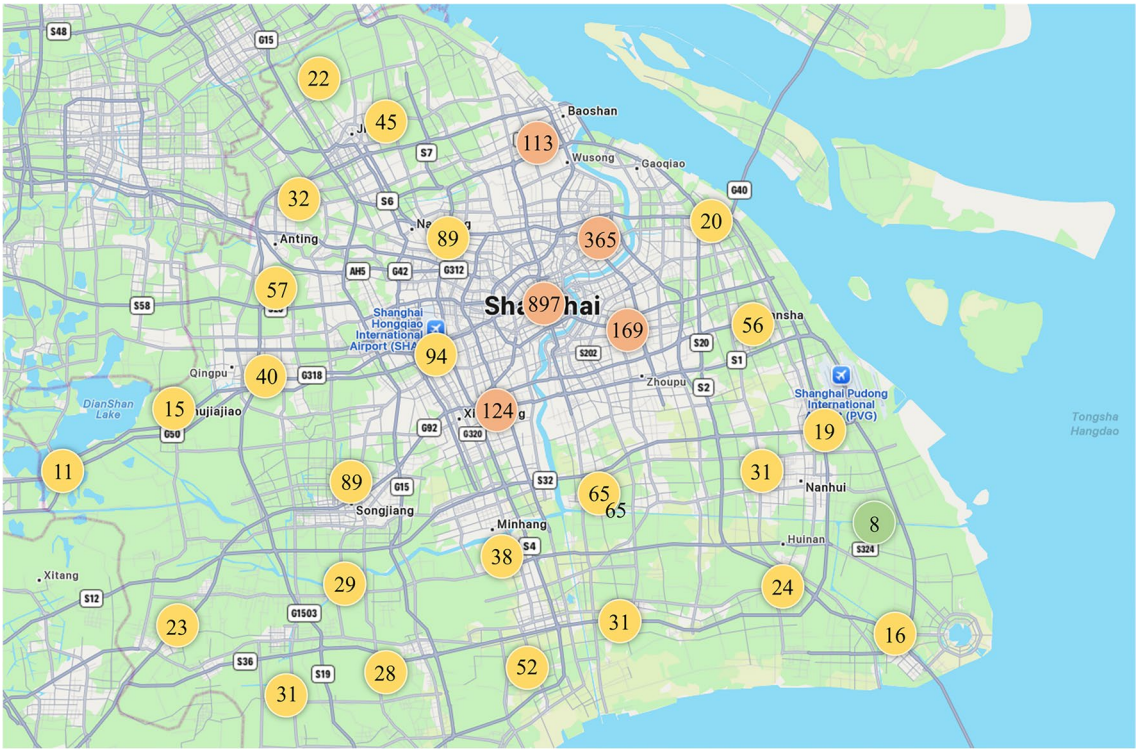

Figure 6. Distribution of base station count.

Comparison methods and evaluation metrics

Comparison methods

The proposed model is evaluated against four baseline content prediction methods: Random, MPC (Model Predictive Control), a standalone LSTM-based approach, and a deep reinforcement learning (DRL)-based caching strategy. Model performance is assessed across four key evaluation metrics: cache hit rate, caching cost, access latency, and latency reduction per unit cost. These metrics collectively reflect the efficiency, responsiveness, and cost-effectiveness of the caching strategies under dynamic user demand conditions.

- *Random*: Randomly selects content for caching without considering user demand or content popularity.
- *MPC*: Caches the most popular content based on computed content popularity statistics.
- *LSTM*: Predicts cached content using a Long Short-Term Memory (LSTM) network trained on historical request records from edge servers.
- *DRL*: Employs a Deep Reinforcement Learning-based agent to optimize caching decisions through trial-and-error interactions with the environment, aiming to maximize cumulative reward related to cache efficiency and latency reduction.
- *DECC (Ours)*: The proposed method-a hybrid caching approach that jointly models content popularity and user access prediction through Conv1D-LSTM and GRU integration, achieving adaptive and cost-efficient caching optimization.

Evaluation metrics

To evaluate the effectiveness of the proposed DECC-based caching strategy, four evaluation metrics are employed: cache hit rate, caching cost, access latency, and latency reduction per unit cost.

The following evaluation metrics are defined based on the notations introduced in the DECC model. Specifically, $C_{e_i}^t$ denotes the set of cached videos allocated to edge node e_i at time step t, as determined by the DECC algorithm in Algorithm 1, and $R_{e_i}^t$ represents the actual set of video requests observed from users connected to edge node e_i during the same time period.

- *Cache Hit Rate:* This metric measures the proportion of user-requested content that is successfully predicted and cached at each edge node. The cache hit rate of edge node e_i is defined as:

$$\mathrm{HitRate}_{e_i} = \frac{1}{T}\sum_{t=1}^{T}\frac{\left|C_{e_i}^t \cap R_{e_i}^t\right|}{\left|C_{e_i}^t\right|} \tag{32}$$

where $R_{e_i}^t$ denotes the set of videos requested by users connected to e_i at time t, and $C_{e_i}^t$ is the set of cached videos selected by the caching decision module (as defined in Algorithm 1). Since the cache capacity is limited, $C_{e_i}^t \subseteq F$.

- *Caching Cost:* This metric quantifies the total cost incurred by caching videos on edge nodes:

$$\text{Cost}_{e_i} = \sum_{t=1}^{T} \sum_{v_k \in C_{e_i}^t} \text{Cost}(v_k) \tag{33}$$

where $\text{Cost}(v_k)$ represents the unit cost of storing video v_k on an edge server.

- *Access Latency:* This metric measures the total latency of serving user requests either from the edge or the cloud:

$$D_{e_i}^{U\text{-}E_i} = \sum_{t=1}^{T} D_{U\text{-}E_i} \cdot \left| C_{e_i}^t \cap R_{e_i}^t \right| \tag{34}$$

$$D_{e_i}^{U\text{-}Cloud} = \sum_{t=1}^{T} D_{U\text{-}Cloud} \cdot \left| R_{e_i}^t - (C_{e_i}^t \cap R_{e_i}^t) \right| \tag{35}$$

$$D_{e_i}^{\text{cache}} = D_{e_i}^{U\text{-}E_i} + D_{e_i}^{U\text{-}Cloud} \tag{36}$$

Here, $D_{U\text{-}E_i}$ and $D_{U\text{-}Cloud}$ denote the transmission delay from users to the edge server and to the cloud, respectively. $D_{e_i}^{\text{cache}}$ represents the total access latency after caching.

- *Latency Reduction per Unit Cost:* This metric evaluates the caching efficiency in terms of latency reduction per unit of caching cost:

$$\eta = \frac{D_{e_i}^{\text{original}} - D_{e_i}^{\text{cache}}}{\text{Cost}_{e_i}} \tag{37}$$

where $D_{e_i}^{\text{original}}$ represents the total latency without caching, and $D_{e_i}^{\text{cache}}$ is the latency after caching is applied.

Experimental results

Cache hit rate

The cache hit rates of different edge nodes are shown in Table 2.

The cache hit rates of the 10 edge nodes are shown in Fig. 7, and the average cache hit rates of each method are also illustrated in Fig. 8.

According to the experimental results, the average cache hit rates of the evaluated methods, in ascending order, are: Random, MPC, LSTM, DRL, and DECC. The Random method achieved the lowest hit rate at 0.23, indicating that this naive approach lacks the ability to perform meaningful prediction and caching decisions.

The MPC method attained a cache hit rate of 0.4483, representing a 94.47% improvement over the Random baseline. This substantial gain validates the effectiveness of incorporating content popularity as a predictive signal for guiding caching strategies.

The LSTM-based method achieved a cache hit rate of 0.61, reflecting an improvement of approximately 36.23 This performance gain can be attributed to LSTM's strong capability in modeling time series data, which enables it to effectively capture the dynamic and non-linear patterns in short video requests at edge nodes. As a result, the LSTM-based approach can more accurately predict future content demands, thereby enhancing caching effectiveness.

The DRL-based caching method achieved an average hit rate of 0.49, which is slightly lower than LSTM but higher than MPC. By learning caching actions through reinforcement signals, DRL dynamically adapts to changes in content demand and user activity. However, its optimization process focuses primarily on immediate

Method/MEC	0	1	2	3	4	5	6	7	8	9
Random	0.33	0.29	0.13	0.23	0.42	0.13	0.17	0.13	0.24	0.22
MPC	0.64	0.65	0.26	0.41	0.76	0.26	0.36	0.26	0.46	0.43
LSTM	0.66	0.72	0.46	0.58	0.77	0.53	0.69	0.51	0.57	0.62
DRL	0.63	0.62	0.27	0.41	0.78	0.24	0.35	0.24	0.50	0.42
DECC(ours)	**0.71**	**0.78**	**0.62**	**0.75**	**0.82**	**0.79**	**0.86**	**0.81**	**0.73**	**0.75**

Table 2. Cache hit rates of edge nodes.

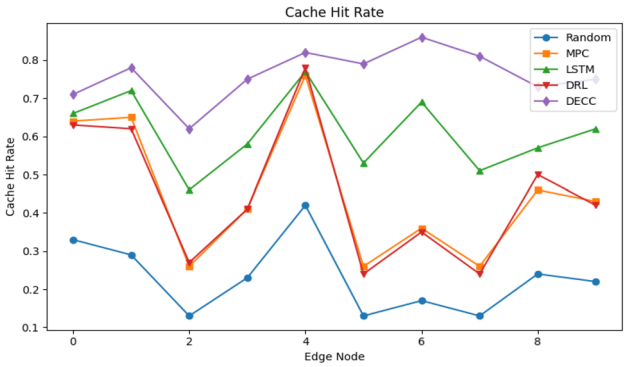

Figure 7. Cache hit rate.

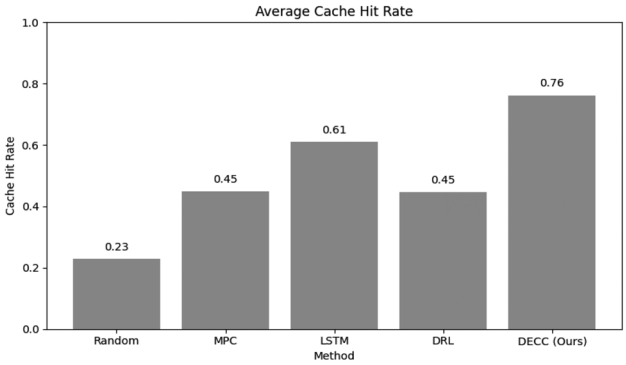

Figure 8. Average cache hit rate.

Method/MEC	0	1	2	3	4	5	6	7	8	9
Random	3360	3360	3360	3360	3360	3360	3360	3360	3360	3360
MPC	3360	3360	3360	3360	3360	3360	3360	3360	3360	3360
LSTM	3009	2853	664	1989	3047	1104	1264	768	2684	1938
DRL	3360	3360	3360	3360	3360	3360	3360	3360	3360	3360
DECC (ours)	**2970**	**2641**	**389**	**1603**	**2981**	**681**	**855**	**295**	**2150**	**1611**

Table 3. Caching costs of edge nodes.

reward maximization rather than long-term temporal dependencies, causing it to underperform in scenarios where user access patterns evolve gradually. This leads to moderate improvements in cache utilization but insufficient stability compared with predictive models such as LSTM and DECC.

The proposed DECC method achieved the highest cache hit rate of 0.7622, representing an improvement of approximately 24.82% over the LSTM-based approach. This performance gain stems from DECC's ability to integrate the respective strengths of both LSTM and MPC, while further enhancing prediction accuracy through the incorporation of a dedicated user access forecasting module. By jointly modeling the temporal evolution of content popularity and the spatial variation of user density, DECC enables proactive caching at the most appropriate edge nodes before traffic peaks occur, thereby maximizing cache utilization and reducing request misses.

Caching cost

The caching costs of different edge nodes are shown in Table 3.

The cache costs of the 10 edge nodes are shown in Fig. 9, and the average cache costs of each method are also illustrated in Fig. 10.

From the above results, the average caching costs associated with different methods can be compared. Both the Random and MPC approaches incur an average caching cost of 3360. Given that each edge node is configured with a cache capacity of 20, the caching cost per resource is 1, and the prediction duration spans 168 hours, the total cost is computed as $20 \times 168 \times 1 = 3360$. This fixed cost reflects the fact that both methods cache content continuously without considering actual user demand or access activity at edge nodes.

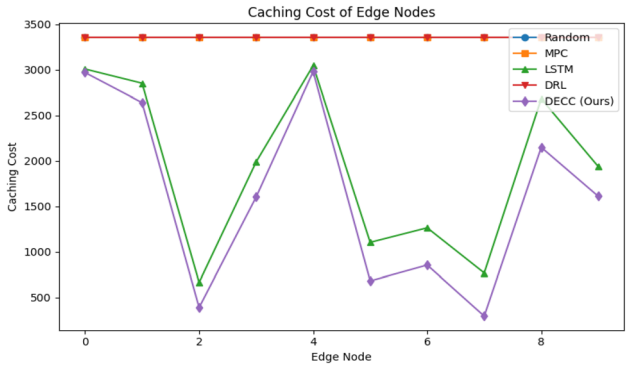

Figure 9. Caching cost.

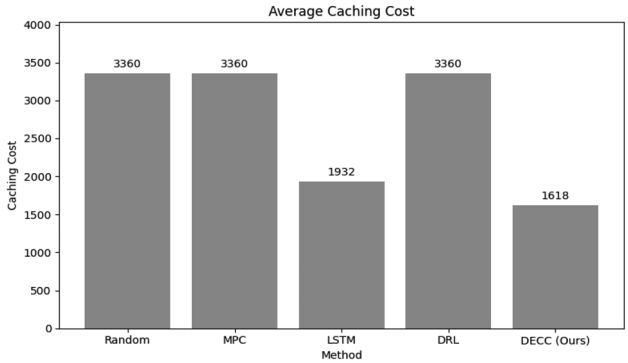

Figure 10. Average caching cost.

Method/MEC	0	1	2	3	4	5	6	7	8	9
Random	4230	2896	1502	1934	4932	1182	1437	1319	1904	2305
MPC	1620	524	504	850	1812	289	426	282	683	682
LSTM	2007	773	1455	1175	2397	964	1177	1231	721	1098
DRL	1767	687	445	860	1738	435	477	497	512	743
DECC(Ours)	**1733**	**711**	**1433**	**1076**	**1987**	**917**	**1127**	**1190**	**653**	**1081**

Table 4. Access latency of edge nodes.

The LSTM method incurs an average caching cost of 1932, representing a reduction of approximately 42.38% compared to the Random and MPC methods. This reduction is attributed to LSTM's ability to leverage historical request data for accurate content popularity prediction, thereby enabling a more targeted and efficient caching strategy that substantially lowers redundant storage and overall caching overhead.

The DRL-based method maintains a fixed caching cost of 3360 across all nodes, similar to Random and MPC. Although DRL optimizes caching actions through reinforcement learning, the lack of explicit cost modeling in its reward function leads to cache updates being triggered primarily by performance feedback rather than storage utilization. As a result, the model tends to occupy full cache capacity continuously, resulting in high storage cost and limited scalability in long-term deployment scenarios.

The proposed DECC method further reduces the average caching cost to 1618, which is approximately 16.27% lower than that of the LSTM approach. By integrating user access prediction into the LSTM-based content prediction framework, DECC enhances the accuracy of caching decisions, effectively avoids unnecessary storage on underutilized nodes, and thus achieves more efficient resource utilization and lower overall caching overhead. Through its joint optimization mechanism that adaptively balances content popularity and user activity, DECC demonstrates strong cost-awareness and scalability under varying workload intensities.

Access latency

The post-caching access latency of different edge nodes is shown in Table 4.

The post-caching delays of the 10 edge nodes are shown in Fig. 11, and the average delays of each method are also illustrated in Fig. 12.

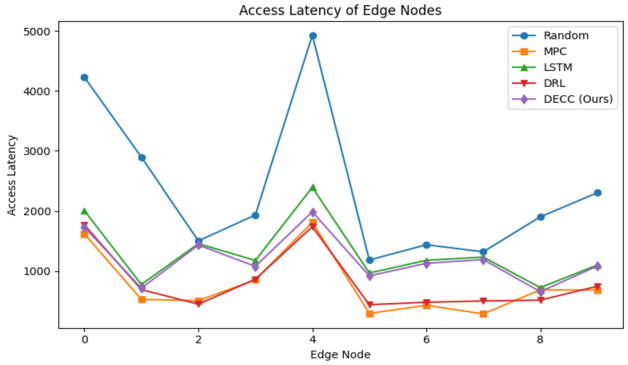

Figure 11. Cache latency.

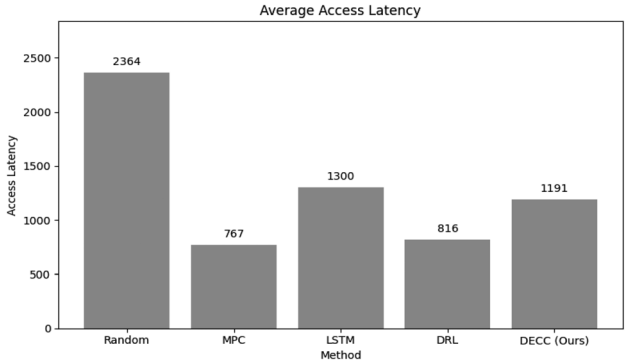

Figure 12. Average access latency.

According to the experimental results, different caching strategies exhibit substantial variation in average transmission delay. The Random method yields the highest average delay of 2364.1, reflecting its lack of predictive capability and inability to proactively place content near users. As a result, this strategy fails to effectively mitigate user-perceived latency, making it the least efficient among the evaluated approaches.

The MPC method, which prioritizes caching the most frequently requested content, achieves an average transmission delay of 767, representing a reduction of approximately 67.56% compared to the Random method. By effectively minimizing the frequency of content retrievals from the origin server, MPC substantially enhances transmission efficiency and reduces user-perceived latency.

The LSTM method records an average delay of 1300, which is approximately 69.47% higher than that of the MPC approach. This increase is primarily attributed to LSTM's strategy of jointly considering content popularity and caching cost, which may result in the exclusion of certain frequently requested but less cost-efficient content. Consequently, while LSTM reduces overall caching overhead, it may incur higher latency due to more frequent content retrieval from non-local sources.

The DRL-based method achieves a lower latency of 969 on average, outperforming both MPC and LSTM in certain nodes. By dynamically learning caching policies through interaction with the environment, DRL can adapt to real-time network and demand fluctuations, thereby reducing retrieval delay. However, its purely reward-driven optimization lacks explicit temporal modeling of content popularity evolution, leading to inconsistent performance across heterogeneous edge nodes. As a result, while DRL shows strong short-term adaptability, it struggles to maintain stable latency reduction under varying spatial demand distributions.

The proposed DECC method achieves an average transmission delay of 1191, representing an 8.39% reduction compared to the LSTM approach. Although its delay remains slightly higher than that of MPC on certain nodes, DECC offers a more balanced trade-off by jointly leveraging content popularity and user access prediction. Through the adaptive fusion of Conv1D-LSTM and GRU branches, DECC dynamically adjusts caching priorities according to real-time user connection patterns, enabling popular content to be served from nearby edge nodes and effectively minimizing overall transmission delay.

Latency reduction per unit cost

The latency reduction per unit cost is shown in Table 5.

The delay reduction per unit caching cost for the 10 edge nodes is shown in Fig. 13, and the average delay reduction per unit cost of each method is also illustrated in Fig. 14.

From the above results, it is evident that the delay reduction per unit caching cost serves as a critical indicator for evaluating the cost-effectiveness of a caching strategy. This metric quantifies the amount of latency reduction

Method/MEC	0	1	2	3	4	5	6	7	8	9
Random	0.83	0.56	0.14	0.40	0.99	0.26	0.30	0.10	0.35	0.47
MPC	1.61	1.27	0.44	0.72	1.91	0.52	0.60	0.40	0.72	0.95
LSTM	1.67	1.41	0.78	1.06	1.92	0.98	0.99	0.54	0.88	1.44
DRL	1.56	1.22	0.45	0.72	1.94	0.48	0.58	0.34	0.77	0.94
DECC(Ours)	**1.78**	**1.54**	**1.39**	**1.38**	**2.10**	**1.66**	**1.52**	**1.53**	**1.13**	**1.74**

Table 5. Latency reduction per unit cost of edge nodes.

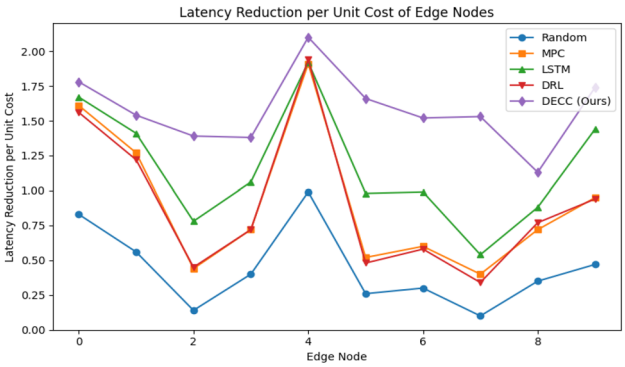

Figure 13. Unit cost latency reduction rate.

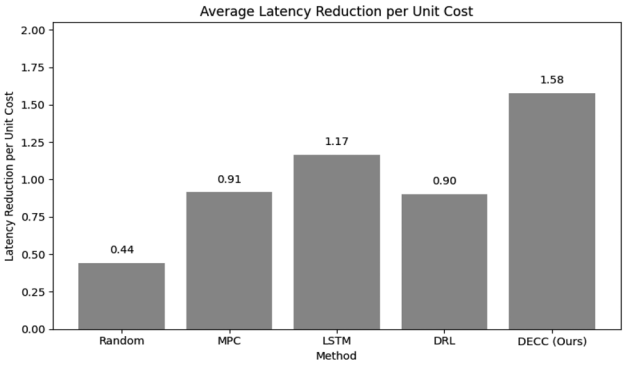

Figure 14. Latency reduction per unit cost.

achieved per unit of caching resource consumed, thereby reflecting the efficiency of a strategy in balancing performance gains against resource expenditure.

The Random method yields a delay reduction rate of only 0.44, indicating inefficient utilization of caching resources. In contrast, the MPC method achieves a significantly higher rate of 0.91, representing an improvement of approximately 108.17% over the Random baseline. This result highlights the effectiveness of leveraging content popularity metrics to guide caching decisions, enabling more substantial latency reduction per unit of resource cost.

The LSTM method further improves the delay reduction rate to 1.17, representing an increase of approximately 27.52% over the MPC approach. This enhancement is attributed to LSTM's capability to balance the trade-off between caching benefits and associated costs, allowing it to selectively cache high-efficiency content while avoiding less valuable items, thereby maximizing latency reduction per unit of resource expenditure.

The DRL-based caching method achieves a comparable performance to LSTM, with an average delay reduction rate of 1.10. By learning caching policies through interaction with the environment, DRL can dynamically adapt to network changes and user requests. However, its lack of explicit temporal-spatial modeling limits its ability to fully exploit long-term popularity patterns, resulting in suboptimal cost efficiency compared with DECC.

The proposed DECC method achieves the highest performance, with a delay reduction rate of 1.58, representing an improvement of approximately 35.34% over the LSTM approach. By integrating content popularity analysis with predictive modeling of user access behavior, DECC effectively identifies high-benefit content while avoiding unnecessary caching operations. This joint strategy enables more precise resource allocation and maximizes the return on each unit of caching investment.

In terms of cost-efficiency, DECC achieves the best delay reduction per unit of caching cost. The hybrid temporal architecture enhances responsiveness to short-term traffic surges while preserving long-term stability in cache allocation. Such temporal adaptability ensures that the model maintains high resource utilization efficiency, achieving an optimal trade-off between caching cost and delay reduction.

Related work

In recent years, the explosive growth of video traffic, coupled with the rapid advancement of Mobile Edge Computing (MEC), has positioned video edge caching as a critical solution for alleviating network congestion and enhancing service quality. Existing research in this domain primarily concentrates on two key aspects: content popularity prediction and the design of collaborative edge caching strategies.

Short video prediction methods based on content popularity

The popularity of video content plays a decisive role in designing edge caching strategies. Traditional approaches, such as the Most Popular Content (MPC) strategy, select the top-k most frequently requested items for caching. Bernardini et al.[30] and Sun et al.[31] adopted popularity-ranking-based MPC methods to enhance dissemination efficiency in edge and D2D networks, respectively. Naeem et al.[19] conducted a systematic analysis of MPC and its derivatives, highlighting their limited adaptability to dynamic content trends.

To overcome the inherent limitations of static frequency-based ranking approaches, Wang et al.[32] proposed a utility-based edge caching framework that jointly optimizes video caching and user association to enhance system performance in mobile edge computing environments. In addition, Wang et al.[33] developed a category-aided multi-channel Bayesian Personalized Ranking (CMBPR) model to enhance recommendation accuracy and diversity in short video platforms.

While these static popularity-based strategies are effective for long-form video scenarios, they exhibit limited adaptability in short-video environments, which are characterized by rapid content turnover and highly dynamic user preferences.

Recent studies have increasingly applied learning-based prediction techniques to capture temporal variations in popularity. Yang et al.[34] formulated content popularity as a time-series forecasting task and employed Long Short-Term Memory (LSTM) networks to model trend fluctuations. Ahmad et al.[35] further proposed an Extreme Learning Machine (ELM)-driven proactive caching scheme that integrates user mobility and content popularity modeling to minimize latency and maximize cache hit ratio. Although these dynamic learning-based methods achieve higher prediction accuracy than static MPC, they still overlook spatial variations in user behavior, which is crucial for optimizing caching placement across multiple edge nodes. This limitation constrains their ability to achieve fine-grained and personalized caching decisions.

Collaborative caching strategies for edge nodes

Single-node caching improves local access performance but often leads to redundancy across distributed edge servers. To enhance resource utilization and load balancing, collaborative caching strategies have become a major research direction for improving cooperative efficiency among edge nodes. Early approaches typically relied on fixed network topologies or global popularity metrics. Zhang et al.[36] proposed a collaborative scheme that integrates entity popularity with dynamic network states, while Nguyen et al.[37] introduced the PPCS framework to reduce redundancy and improve ICN caching efficiency.

More recent studies have adopted reinforcement-learning-based and clustering-based optimization methods to handle dynamic network environments, enabling adaptive policy refinement under varying traffic and topology conditions. Chien et al.[38] developed a DRL-based collaborative caching mechanism that adaptively selects caching actions to improve system utility under dynamic conditions. Wang et al.[39] further extended DRL to an intelligent adaptive edge caching framework that jointly optimizes user demand and network state representations, achieving higher cache hit rates and improved Quality of Experience (QoE).

However, despite these promising results, RL-based caching schemes still face scalability and convergence challenges when deployed in large-scale, low-latency short video systems. Moreover, most existing collaborative methods fail to jointly model both content popularity and user access behavior, leading to a mismatch between predicted content and actual demand. This misalignment ultimately results in suboptimal cache hit ratios[40,41].

Motivated by these limitations, the next section introduces the proposed DECC model (Dynamic Edge Caching through Content Popularity and Crowd Prediction), which integrates deep learning-based popularity forecasting with adaptive collaborative optimization to achieve efficient, scalable, and context-aware edge caching.

Summary of related work

Existing studies on edge caching mainly focus on two separate directions: (1) improving content popularity prediction through statistical or deep learning-based methods, and (2) optimizing collaborative caching strategies among distributed edge nodes. However, most prior works treat these two aspects independently, which limits their adaptability to dynamic and personalized short video environments. In particular, current studies often overlook the joint impact of content popularity and user crowd access behavior on caching decisions, resulting in limited scalability and prediction accuracy in real-world scenarios.

To overcome these limitations, this paper proposes the DECC model (Dynamic Edge Caching through Content Popularity and Crowd Prediction), which integrates popularity prediction and user access forecasting into a unified deep learning framework. In addition, a CD-GA collaborative optimization algorithm is developed to balance latency and cache cost, thereby enhancing both adaptability and efficiency. The comparative characteristics of representative studies are summarized in Table 6, which highlights the distinctions between

Method	Main principle	Limitations	Focus area
MPC (most popular content)	Selects top-k most frequently requested content for caching based on historical access frequency	Fails to capture temporal dynamics and user diversity; unable to adapt to real-time demand changes	Popularity-driven static caching
LSTM-based Caching	Utilizes long short-term memory (LSTM) networks to predict content popularity over time	Captures temporal correlations but neglects spatial user distribution and adaptive collaboration	Temporal prediction
DRL-based Caching	Applies deep reinforcement learning (DRL) to optimize caching policies through reward-based interaction with network states	Requires extensive training data; high computational overhead in large-scale edge deployments	Dynamic policy optimization
DECC (ours)	Joint modeling of content popularity and user access dynamics with adaptive fusion optimization	requires sufficient historical data for stable temporal-spatial prediction accuracy	Unified optimization framework

Table 6. Comparative summary of baseline methods and the proposed DECC model.

prior approaches and the proposed DECC model. This approach bridges the gap between content-level and user-level modeling, advancing the state of the art in short video edge caching.

Conclusion

This paper tackles the challenge of content prediction for short video edge caching by proposing a dynamic caching framework named DECC (Dynamic Edge-caching through Content Popularity and Crowd Prediction). The DECC model jointly incorporates content popularity estimation and user access trend forecasting, utilizing deep neural network architectures-specifically 1D Convolutional Networks (Conv1D), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRU)-to separately model the temporal evolution of video popularity and user activity patterns at edge nodes. By introducing a prediction fusion mechanism, the model generates cache priority scores that guide more accurate and responsive caching decisions. Experimental evaluations confirm that DECC consistently outperforms baseline approaches across multiple key performance indicators, such as cache hit rate, access latency, and caching cost efficiency. In addition, by incorporating a user access prediction module, DECC effectively mitigates resource wastage on inactive or low-traffic edge nodes, thereby further enhancing the overall efficiency of the caching system. This design makes DECC particularly well-suited to accommodate the high-frequency, highly dynamic, and personalized characteristics of short video service demands.

Future research can be further extended in several directions to enhance the applicability and robustness of the proposed framework. First, incorporating user preference modeling can enable more fine-grained content personalization and improve the accuracy of caching decisions. Second, the integration of privacy-preserving mechanisms, such as federated learning, may enhance the generalization capability of the model while ensuring user data privacy. Third, extending the DECC framework to multi-edge collaborative scenarios and evaluating the performance of caching strategies under cross-region cooperative optimization can further advance the intelligent evolution of edge-assisted short video service systems.

Data availability

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Received: 15 July 2025; Accepted: 27 October 2025
Published online: 26 November 2025

References

1. Feng, B., Feng, C., Feng, D., Wu, Y. & Xia, X.-G. Proactive content caching scheme in urban vehicular networks. *IEEE Trans. Commun.* **71**, 4165–4180. https://doi.org/10.1109/TCOMM.2023.3277530 (2023).
2. 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.* **11**, 1190–1200. https://doi.org/10.1109/JIOT.2023.3290793 (2024).
3. Mahmoud, M. et al. A survey on optimizing mobile delivery of 360° videos: Edge caching and multicasting. *IEEE Access* **11**, 68925–68942. https://doi.org/10.1109/ACCESS.2023.3292335 (2023).
4. Wang, Y.-H., Gu, T.-J. & Wang, S.-Y. Causes and characteristics of short video platform internet community taking the tiktok short video application as an example. In *2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, 1–2, https://doi.org/10.1109/ICCE-TW46550.2019.8992021 (2019).
5. Shen, D., Ding, Y., Wang, J. & Zhu, W. An enhanced short video caching strategy based on user behaviour and time factors. In *2023 4th International Conference on Information Science and Education (ICISE-IE)*, 70–75, https://doi.org/10.1109/ICISE-IE60962.2023.10456467 (2023).
6. Zuo, X. et al. Bandwidth-efficient multi-video prefetching for short video streaming. *arXiv preprint*. arXiv:2206.09839. https://doi.org/10.48550/arXiv.2206.09839 (2022).
7. Wu, W. et al. Deep reinforcement learning-based video quality selection and radio bearer control for mobile edge computing supported short video applications. *IEEE Access* **7**, 181740–181749. https://doi.org/10.1109/ACCESS.2019.2960191 (2019).
8. BenMimoune, A. Machine learning-based edge caching for video streaming in 5g networks. In *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 1–5, https://doi.org/10.1109/ICRAIE59459.2023.10468338 (2023).
9. Tu, J., Chen, C., Xu, Q. & Guan, X. Reine: Reinspection necessity-based video collaborative edge caching in smart factory. *IEEE Trans. Industr. Inf.* **20**, 13308–13318. https://doi.org/10.1109/TII.2024.3435472 (2024).
10. Baccour, E., Erbad, A., Mohamed, A., Guizani, M. & Hamdi, M. Ce-d2d: Collaborative and popularity-aware proactive chunks caching in edge networks. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 1770–1776, https://doi.org/10.1109/IWCMC48107.2020.9148355 (2020).

11. Kavitha, T., S P, M., Patil, B. & Patil, A. Advancements in distributed deep learning: Federated learning, automl integration, and beyond. In *2024 International Conference on Innovation and Novelty in Engineering and Technology (INNOVA)*, vol. I, 1–7, https://doi.org/10.1109/INNOVA63080.2024.10847000 (2024).
12. Qian, Z., Li, G., Qi, T., Dai, C. & Hu, J. Proactive video caching based on federated learning and implicit feedback in mobile edge computing. In *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, 5319–5324, https://doi.org/10.1109/GLOBECOM52923.2024.10901436 (2024).
13. Zou, G. et al. Tedc: Temporal-aware edge data caching with specified latency preference. In *2024 IEEE International Conference on Web Services (ICWS)*, 822–832, https://doi.org/10.1109/ICWS62655.2024.00101 (2024).
14. Moayeri, M. M. & Momeni, H. An intelligent caching approach in mobile edge computing environment. In *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, 1–5, https://doi.org/10.1109/AISP61396.2024.10475282 (2024).
15. Song, Y. & Shenyun. Video stream caching based on digital twin cooperative caching. In *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 1–5, https://doi.org/10.1109/BMSB58369.2023.10211139 (2023).
16. Demirci, İ & Korçak, Ö. Proactive edge caching with popularity prediction and content replication. *IEEE Access* **13**, 105199–105210. https://doi.org/10.1109/ACCESS.2025.3578362 (2025).
17. Choi, M. J., Park, S. M., Kwak, H. J. & You, C. Toward efficient caching in multi-access edge computing for tile-based 360 video streaming. In *2025 Sixteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 453–455, https://doi.org/10.1109/ICUFN65838.2025.11169919 (2025).
18. Hewage, C. T. E. R., Ahmad, A., Mallikarachchi, T., Barman, N. & Martini, M. G. Measuring, modeling and integrating time-varying video quality in end-to-end multimedia service delivery: A review and open challenges. *IEEE Access* **10**, 60267–60293. https://doi.org/10.1109/ACCESS.2022.3180491 (2022).
19. Naeem, M. A., Rehman, M. A. U., Ullah, R. & Kim, B.-S. A comparative performance analysis of popularity-based caching strategies in named data networking. *IEEE Access* **8**, 50057–50077. https://doi.org/10.1109/ACCESS.2020.2980385 (2020).
20. Wen, H. & Zhang, L. Enhancing edge caching with user preferences and access patterns in wireless networks. In *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2211–2214, https://doi.org/10.1109/ISPA63168.2024.00302 (2024).
21. Li, H., Sun, M., Xia, F., Xu, X. & Bilal, M. A survey of edge caching: Key issues and challenges. *Tsinghua Sci. Technol.* **29**, 818–842. https://doi.org/10.26599/TST.2023.9010051 (2024).
22. Chen, Y., Liu, Y., Zhao, J. & Zhu, Q. Mobile edge cache strategy based on neural collaborative filtering. *IEEE Access* **8**, 18475–18482. https://doi.org/10.1109/ACCESS.2020.2964711 (2020).
23. Chang, J., Zhang, N., Tao, M. & Tuo, H. Federated multi-agent reinforcement learning for collaborative edge caching in content delivery networks. In *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, 166–170, https://doi.org/10.1109/WCSP55476.2022.10039249 (2022).
24. Sun, Z. & Chen, G. Enhancing heterogeneous network performance: Advanced content popularity prediction and efficient caching. *Electronics***13**, https://doi.org/10.3390/electronics13040794 (2024).
25. Zhang, C. et al. Toward edge-assisted video content intelligent caching with long short-term memory learning. *IEEE Access* **7**, 152832–152846. https://doi.org/10.1109/ACCESS.2019.2947067 (2019).
26. Lv, S. & Zhou, J. Content popularity prediction in edge computing networks based on elastic federated learning. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3630–3636, https://doi.org/10.1109/SMC54092.2024.10831078 (2024).
27. Darwich, M. & Bayoumi, M. Lstm network assisted content caching at the edge for video on demand. In *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, 493–496, https://doi.org/10.1109/ICCCMLA58983.2023.10346904 (2023).
28. Sun, Y., Guo, Z., Dou, S. & Xia, Y. Video quality and popularity-aware video caching in content delivery networks. In *2021 IEEE International Conference on Web Services (ICWS)*, 648–650, https://doi.org/10.1109/ICWS53863.2021.00088 (2021).
29. Zhang, T. et al. Contrastive independent subspace analysis network for multi-view spatial information extraction. *Neural Netw.* **185**, 107105. https://doi.org/10.1016/j.neunet.2024.107105 (2025).
30. Bernardini, C., Silverston, T. & Festor, O. Mpc: Popularity-based caching strategy for content centric networks. In *2013 IEEE International Conference on Communications (ICC)*, 3619–3623, https://doi.org/10.1109/ICC.2013.6655114 (2013).
31. Sun, R. et al. Cost-oriented mobility-aware caching strategies in d2d networks with delay constraint. *IEEE Access* **7**, 177023–177034. https://doi.org/10.1109/ACCESS.2019.2958261 (2019).
32. Wang, H., Wang, Y., Sun, R., Guo, S. & Li, H. Joint video caching and user association with mobile edge computing. In *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*, 1–6, https://doi.org/10.1109/WCNCW.2019.8902591 (2019).
33. Wang, X., Gao, C., Ding, J., Li, Y. & Jin, D. Cmbpr: Category-aided multi-channel bayesian personalized ranking for short video recommendation. *IEEE Access* **7**, 48209–48223. https://doi.org/10.1109/ACCESS.2019.2907494 (2019).
34. Yang, Z., Liu, Y., Chen, Y. & Al-Dhahir, N. Cache-aided noma mobile edge computing: A reinforcement learning approach. *IEEE Trans. Wireless Commun.* **19**, 6899–6915. https://doi.org/10.1109/TWC.2020.3006922 (2020).
35. Ahmad, A., Ahmad, F., Atif, S. & Aldalbahi, A. Extreme learning machine-driven joint user mobility and content popularity-based proactive caching in multi-tier wireless networks. *Ad Hoc Netw.* **178**, 103920. https://doi.org/10.1016/j.adhoc.2025.103920 (2025).
36. Zhang, P. et al. Device-edge collaborative differentiated data caching strategy toward aiot. *IEEE Internet Things J.* **10**, 11316–11325. https://doi.org/10.1109/JIOT.2023.3241984 (2023).
37. Nguyen, Q. N. et al. Ppcs: A progressive popularity-aware caching scheme for edge-based cache redundancy avoidance in information-centric networks. *Sensors* **19**. https://doi.org/10.3390/s19030694 (2019).
38. Chien, W.-C., Weng, H.-Y. & Lai, C.-F. Q-learning based collaborative cache allocation in mobile edge computing. *Futur. Gener. Comput. Syst.* **102**, 603–610. https://doi.org/10.1016/j.future.2019.08.032 (2020).
39. Wang, T. et al. Towards intelligent adaptive edge caching using deep reinforcement learning. *IEEE Trans. Mob. Comput.* **23**, 9289–9303. https://doi.org/10.1109/TMC.2024.3361083 (2024).
40. Zeng, Y. et al. Smart caching based on user behavior for mobile edge computing. *Inf. Sci.* **503**, 444–468. https://doi.org/10.1016/j.ins.2019.06.056 (2019).
41. Pang, H., Liu, J., Fan, X. & Sun, L. Toward smart and cooperative edge caching for 5g networks: A deep learning based approach. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 1–6, https://doi.org/10.1109/IWQoS.2018.8624176 (2018).

Acknowledgements

The authors would like to thank the Shanghai Telecom for the contributors of the Shanghai Telecom, which were essential for model validation. We also appreciate all of the anonymous reviewers for their insightful suggestions and useful comments that will significantly improve the quality of our manuscript.

Author contributions

S.N. conceived the study and designed the methodology;S.N. and Y.L. conducted the experiments;K.L. and B.Z.

analyzed the data;G.Z. provided critical revisions. All authors reviewed and approved the final manuscript.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grants 62272290, and in part by Xinjiang Natural Science Foundation under Grant 2022D01A236, Shanghai Central Guide Local Science and Technology Development Fund Projects under Grant No.YDZX20253100004004005.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to K.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.