Proactive Spatio-Temporal Request Prediction for Replica Placement in Edge-Cloud Computing

Hao Zheng, *Graduate Student Member, IEEE*, Zhigang Hu, Liu Yang, Aikun Xu, Meiguang Zheng, Hui Xiao, and Keqin Li, *Fellow, IEEE*

***Abstract*—User requests in edge computing environments are inherently decentralized and dynamic, posing significant challenges for efficient and adaptive service replica placement. To address this, we formulate the service replica placement problem in an edge-cloud collaborative environment, explicitly incorporating the spatio-temporal distribution of user requests. By capturing spatial and temporal correlations, we predict future request patterns to enable forward-looking replica placement. Given the NP-hard nature of the optimization problem, we design a deep reinforcement learning (DRL) algorithm that optimizes replica placement decisions based on predictive modeling. To validate our approach, we conduct extensive experiments on real-world datasets across two typical application scenarios—grid-based and graph-based request distributions. Experimental results show our method reduces average response latency by up to 59.6% and boosts service provider profitability by 4.85% compared to reactive and temporal-only baselines. The proposed framework provides a novel and effective solution for proactive service provisioning in edge computing environments.**

***Index Terms*—Deep reinforcement learning (DRL), edge computing, replica placement, spatio-temporal prediction.**

I. Introduction

MOBILE edge computing (MECs) has emerged as a promising paradigm to bring cloud resources closer to end users by deploying micro data centers at the network's periphery [1], [2], [3]. This proximity enables rapid service responses and reduces the load on core networks [4], [5]. However, the deployment of MEC also introduces new challenges, particularly in managing limited edge resources and addressing the highly dynamic and decentralized nature of user requests.

As illustrated in Fig. 1, user requests in edge environments exhibit complex spatio-temporal characteristics in practice [6], [7]. In a grid-based scenario [Fig. 1(a)], the service region is partitioned into uniform grids, each experiencing varying levels of request intensity (from light to heavy).

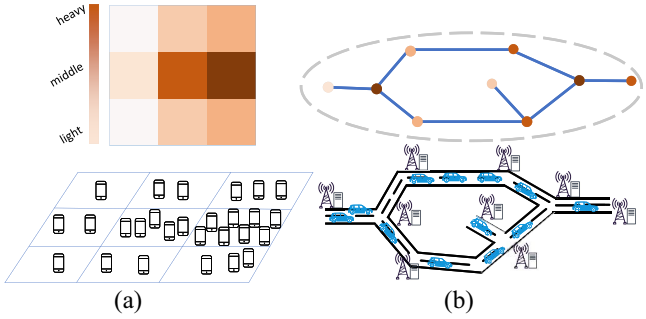

Fig. 1. Visualization of spatio-temporal user request distributions in grid-based and graph-based scenarios. (a) Grid-based scenario, where the coverage area is divided into uniform grids with varying request intensities. (b) Graph-based scenario, where user requests follow a road network topology, and each node experiences dynamic demand over time.

Meanwhile, in a graph-based scenario [Fig. 1(b)], requests predominantly follow road networks, where each node [e.g., a base station or roadside unit (RSU)] may experience fluctuating demand over time. These spatial variations, compounded by temporal dynamics, pose significant challenges for resource allocation and service provisioning. Traditional service replica placement methods primarily focus on reactive strategies, which adjust replica deployment by designing predictive models or heuristics based on historical request patterns [8], [9]. Such reactive methods often fail to cope with sudden changes in demand due to their inherent response delay [10], [11], [12], [13]. Moreover, many existing predictive models only consider temporal dynamics, overlooking the inherent spatial correlations that are critical in geographically distributed settings [14], [15], [16]. Consequently, designing an efficient and proactive replica placement strategy must account for both where and when demand emerges, ensuring that edge resources are deployed to best serve the evolving distribution of user requests.

However, the inherent complexity and nonfixed nature of request patterns, especially mapping high-dimensional, uncertain predictions to multiple conflicting goals, poses a challenge for designing such proactive strategies [17], [18]. Fundamentally, proactive placement in such dynamic environments constitutes a sequential decision-making problem, where the placement of replicas at the current time affects future network costs and system states. Deep reinforcement learnings (DRLs) is well-suited for this scenario, as it excels at modeling sequential decision processes and optimizing long-term cumulative rewards, aligning perfectly with the dynamic

Received 3 March 2025; revised 23 May 2025; accepted 10 June 2025. Date of publication 13 June 2025; date of current version 25 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62172442 and Grant 62172451; in part by the China Scholarship Council under Grant 202306370179; and in part by the High Performance Computing Center of Central South University. *(Corresponding author: Liu Yang.)*

Hao Zheng, Zhigang Hu, Liu Yang, Aikun Xu, Meiguang Zheng, and Hui Xiao are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: zhenghao@csu.edu.cn; zghu@csu.edu.cn; yangliu@csu.edu.cn; aikunxu@csu.edu.cn; zhengmeiguang@csu.edu.cn; huixiao@csu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/JIOT.2025.3579618

nature of replica placement [19], [20], [21], [22]. Specifically, DRL learns optimal policies through trial-and-error interaction with the environment, automatically uncovering complex relationships among spatial locations, temporal patterns, and placement strategies to maximize a reward signal that captures both response latency and operational cost.

Based on the aforementioned challenges, this article investigates the service replica placement problem from the perspective of service providers operating in an edge-cloud collaborative environment. We rigorously formulate the replica placement problem by incorporating both the spatial and temporal distribution of user requests. By capturing the interdependencies between neighboring regions and the evolution of request patterns over time, we propose a proactive and forward-looking replica placement strategy that anticipates future demand rather than merely reacting to past trends.

To tackle the NP-hard nature of the optimization problem, our approach integrates spatio-temporal predictive learning with DRL. The predictive module extracts key features from historical data to accurately forecast future request distributions, while the DRL-based solution leverages these predictions to determine an optimal replica placement scheme that minimizes response latency and balances resource utilization with operational costs.

We conduct extensive trace-driven experiments on both grid-based and graph-based real scenario datasets, and the results show that our proposed method significantly reduces overall response latency and enhances service provider profitability compared to conventional reactive and temporal prediction-based approaches. These results not only validate the effectiveness of our approach but also underscore its potential for practical deployment in IoT and edge computing applications. The main contributions of this article are as follows.

1) We formulate the replica placement problem in an edge-cloud collaborative environment by explicitly modeling the spatio-temporal dynamics of user requests, addressing the limitations of prior temporal-only or reactive approaches.
2) We propose a novel framework that combines spatio-temporal request prediction with DRL, enabling proactive and adaptive replica placement.
3) We validate our method through extensive experiments on real-world datasets, showing substantial improvements in both response latency and service provider profitability.

The remainder of this article is organized as follows. Section II reviews the related work. Section III describes the system architecture and formally defines the replica placement problem. In Section IV, we present our approach for spatio-temporal prediction of user requests, and Section V details the proactive replica placement strategy. Section VI outlines the experimental results and evaluation. Finally, Section VII concludes this article and discusses directions for future work.

II. Related Work

The literature on replica placement in edge computing can be broadly categorized into reactive approaches, predictive methods based on time-series analysis, recent deep learning (DL)–driven solutions, and the latest reinforcement learning-based methods [29].

A. Reactive Replica Placement Strategies

Early research focused on reactive replica placement strategies [11], [24], where placement decisions were adjusted based on historical observations of user requests. Chang and Wang [11] proposed an adaptive replica placement algorithm that monitors request patterns and periodically relocates replicas to minimize access latency. Similarly, Li et al. [24] developed a dynamic placement scheme that responds to workload changes by migrating replicas across edge nodes. Wang et al. [23] introduced a decentralized replica placement framework that uses local observations to make placement decisions, reducing coordination overhead. Xu et al. [30] leveraged game-theoretic approaches to model the competitive behavior among edge service providers. While these methods dynamically adapt to changing workloads, their reliance on past data often results in a lag in response to sudden demand fluctuations. Such reactive schemes are particularly inadequate in edge environments where the spatio and temporal variability of user requests can be pronounced.

B. Traditional Predictive Methods

To overcome the limitations of reactive strategies, predictive methods have been proposed [31]. Traditional statistical approaches, such as ARIMA [25] and exponential weighted moving averages [26], model the temporal evolution of request rates. Maia et al. [25] demonstrated that ARIMA-based forecasting can improve replica placement effectiveness in content delivery networks. Li et al. [26] incorporated user mobility patterns into exponential smoothing models to predict future request locations. Some research has also explored optimization algorithms like swarm intelligence (e.g., variants of particle swarm optimization [32] or ant colony optimization [33]) to solve the placement problem formulated with predicted future states, aiming for better resource utilization based on forecasted demand. However, these methods typically neglect the spatio dependencies inherent in user request distributions. For instance, while [13] and [24] demonstrate that prediction-based replica placement can improve performance, their models generally assume that requests are independent of geographic location, thereby limiting their effectiveness in scenarios where spatio correlations play a crucial role.

C. Deep Learning-Based Methods

More recently, DLs techniques have emerged as powerful tools for capturing the complex spatio-temporal dynamics of user requests. recurrent neural networks (RNNs), particularly long–short-term memorys (LSTMs) [12], [28] and gated recurrent units (GRUs) models, have shown promise in forecasting temporal trends in service demand. Wang et al. [12] employed LSTM networks to predict request patterns and integrated these predictions into a replica placement optimization framework. Zaman et al. [28] developed a hybrid LSTM-CNN architecture for multistep ahead forecasting of edge computing

TABLE I
OVERVIEW OF REPRESENTATIVE REPLICA PLACEMENT TECHNIQUES

Source	Proactive placement	Prediction technique	Temporal	Spatial
Chang et al. [11]	×	-	-	-
Wang et al. [23]	×	-	-	-
Li et al. [24]	×	-	-	-
Fahadi et al. [13]	✓	Assume perfect prediction	✓	×
Maia et al. [25]	✓	ARIMA	✓	×
Li et al. [26]	✓	Multiple linear regression	✓	×
Li et al. [27]	✓	Gray Markov chain	✓	×
Zaman et al. [28]	✓	LSTM (Predict VNF demand)	✓	×
Wang et al. [12]	✓	LSTM (Predict data popularity)	✓	×
Our work	✓	Spatio-Temporal predictive learning	✓	✓

workloads. However, many early DL applications in this domain still focused predominantly on the time dimension, without robustly integrating spatio information. To address spatio dependencies, Zhang et al. [34] proposed a CNN-based approach for modeling spatio correlations in MEC environments. Moreover, graph neural networks (GNNs) have emerged as particularly suitable for modeling complex spatio relationships, with recent works [35], [36] demonstrating their effectiveness in network resource allocation scenarios. However, most of these approaches still focus primarily on either the temporal or spatio dimension in isolation, without effectively integrating both aspects. Although some works have applied spatio-temporal methods in the context of network resource allocation and base station management [37], [38], they are primarily targeted at infrastructure providers rather than service providers who require proactive replica placement strategies.

D. Reinforcement Learning-Based Methods

The application of reinforcement learning to edge computing problems has gained significant momentum [39]. Traditional RL approaches [40], [41] have been applied to resource allocation and task scheduling problems. Liang et al. [40] used deep Q-networks (DQNs) for dynamic resource allocation in edge computing environments. Yang et al. [41] proposed a multiagent reinforcement learning framework for collaborative edge computing. More sophisticated approaches have emerged that combine RL with other techniques. Actor–Critic methods [42], [43] have shown particular promise for continuous control problems in edge computing. Gao et al. [42] developed an actor–critic framework for joint computation offloading and resource allocation. Zhang et al. [43] employed Proximal policy optimizations (PPOs) for dynamic service placement in edge-cloud environments. Liu et al. [44] leveraged a parameterized DQN framework to jointly optimize service placement and computational resource allocation, with the goal of minimizing long-term task latency under stochastic workloads and migration delays. Zheng et al. [45] formulated the replica placement problem as a multiobjective problem and used a DRL strategy based on dual DQNs to improve latency, reliability, and load balancing. However, most existing RL-based approaches for replica placement do not adequately incorporate predictive components or do not consider the complex spatio-temporal nature of user requests.

In contrast, our work aims to bridge these gaps by integrating both spatio and temporal predictive learning within a unified framework for proactive replica placement. As summarized in Table I, our approach uniquely addresses the limitations of existing methods by simultaneously considering the spatial distribution and temporal evolution of user requests. This enables the design of a DRL-based placement strategy that not only anticipates future demand but also optimizes resource allocation from the service provider's perspective.

III. SYSTEM ARCHITECTURE AND PROBLEM FORMULATION

In this section, we present the overall system architecture and formally define the replica placement problem in an edge-cloud collaborative environment. Then, we describe the architecture and introduce two representative scenarios, followed by the definition of the request model, replica placement, scheduling variables, and problem formulation.

A. System Architecture

Fig. 2 illustrates the three-layer architecture specifically as.

Edge Cloud Layer is a typical wireless access network that consists of n geographically distributed edge clouds, denoted as $S = S_1, S_2, \ldots, S_n$. Each edge cloud has a built-in edge server and network access pointss (APs) (e.g., base station or WiFi hotspot) that provides network access to end users within its coverage area. Due to limited computational and storage resources, these edge clouds must efficiently deploy service replicas.

Regional Central Server Layer orchestrates replica placement and request scheduling across all edge clouds. It collects request data from each edge cloud and makes system-wide decisions based on global resource availability and network conditions.

Central Cloud Layer is denoted by S_0 and the set of all clouds is denoted as $\mathcal{S}_+ = \mathcal{S} \cup \{S_0\}$. It provides virtually unlimited resources but is located farther from end users. Although it guarantees service availability, the higher latency compared to the edge clouds makes it less desirable for delay-sensitive applications.

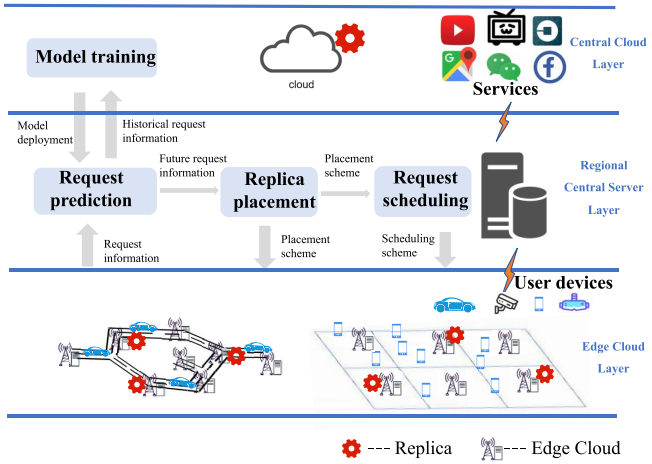

Fig. 2. System of proactive replica placement in edge-cloud collaboration, illustrating the three-layer framework (central cloud, regional central server, and edge cloud layers) with integrated processes for model training, request prediction, replica placement, and request scheduling to optimize service delivery in IoT applications.

TABLE II
NOTATION LIST

Notation	Meaning
$\mathcal{T} = \{0, 1, 2, ...T\}$	Time slots
$\mathcal{S} = \{S_1, S_2, ...S_n\}$	Set of edge clouds
$\mathcal{S}_+ = \mathcal{S} \cup S_0$	Set of clouds
$\mathcal{C} = \{C_1, C_2, ...C_n\}$	Set of grid areas or road sections
$\mathcal{R} = \{r_1, r_2, ...r_n\}$	Set of area requests arrival rate
Cap	Capacity of each replica in edge cloud
μ	Wireless link cost, a constant
d_{ij}	Communication delay between S_i and S_j
$x_i^t \in \{0, 1\}$	Placement variable for slot t
$y_{ij}^t \in [0, 1]$	Shceduling variable for slot t
T_{in}	Prediction input time windows length
SD	Spatial distribution of user requests

To capture diverse application requirements, we consider two typical scenarios.

Grid-Based Scenario: The first typical service scenario targets applications, such as cloud gaming, real-time video streaming and conferencing [46], [47], where requests are dispersed throughout the region. The service region is partitioned into a uniform $n \times n$ grid. Each cell is managed by a base station that monitors local request intensity. This scenario is representative of urban IoT applications where user requests are distributed over a city.

Graph-Based Scenario: The second typical service scenario targets traffic-based applications, such as in-vehicle navigation and assisted driving [48], [49], [50], where requests are sent along roads. Their spatial distribution is usually related to the structure of the road network. The road network is modeled as an undirected graph $G = (V, E, A)$, where V is the set of road segments (nodes), E is the set of connections based on the road topology, and A is the adjacency matrix. In this case, each road segment deploys a RSUs that is responsible for managing and forwarding requests sent from that road segment. This scenario is particularly applicable to vehicular IoT applications.

In both scenarios, grid areas and road sections are utilized to manage user requests, and for simplicity, we denote them uniformly as $C = \{C_1, C_2, \ldots, C_n\}$, where n represents the number of regions. Each region is equipped with an edge server to form an edge cloud. The main notations used in this article are summarized in Table II.

B. Replica Placement and Request Scheduling

To ensure a high-Quality of Services (QoSs) in an edge-cloud collaborative environment, the replica placement scheme must proactively adapt to the dynamic spatio-temporal distribution of user requests. This section introduces the key mechanisms of replica placement and request scheduling, defining their roles, mathematical representations, and underlying assumptions.

Time-Slotted Framework: We model the system dynamics using a discrete-time framework, where the timeline is divided into equal-length time slots $t \in \mathcal{T} = \{0, 1, 2, \ldots, T\}$. Each time slot represents a decision epoch during which the service provider evaluates and adjusts the replica placement based on predicted request patterns. The duration of a time slot (e.g., one hour) is a configurable parameter determined by the service provider, balancing the need for timely adaptation to request fluctuations with the overhead of frequent redeployment. For instance, shorter slots enable rapid responses to sudden demand spikes, while longer slots reduce operational costs.

Replica Placement: Replica placement involves determining the optimal locations for deploying service replicas across the edge clouds and the central cloud. Under the rental rules and cost constraints, we assume that each edge cloud S_i (for $i = 1, \ldots, n$) can deploy at most one service replica, while the central cloud S_0 always has a replica deployed [11]. Each replica deployed at an edge cloud has a fixed maximum processing capacity Cap (i.e., the maximum number of requests processed per second), which is determined by the rented computing, storage, and communication resources.

The replica placement decision at time slot t is represented by a binary vector $X^t = [x_0^t, x_1^t, x_2^t, \ldots, x_n^t]$, where $x_i^t \in \{0, 1\}$ for $i = 0, 1, \ldots, n$. If $x_i^t = 1$, a replica is deployed on edge cloud S_0; otherwise, $x_i^t = 0$ indicates no replica is deployed on that edge cloud. We assume that the central cloud S_0 always has a replica deployed, i.e., $x_0^t \equiv 1$ for all $t \in \mathcal{T}$, serving as a fallback option for request processing.

The placement decision X^t is made proactively at the beginning of each time slot based on predicted request distributions (detailed in Section IV), aiming to minimize response latency and operational costs.

Request Scheduling: Once the replica placement X^t is determined, incoming user requests must be efficiently scheduled to available replicas. Users are distributed across the coverage area and access services by sending requests to their nearest base stations. When a user requests a service that is not deployed on the local edge server, the system must determine where to process this request—either at another edge server hosting a replica or at the central cloud. This decision is made according to scheduling rules designed to optimize service response time while respecting capacity constraints.

Rather than modeling individual request routing, we focus on the statistical distribution of requests across space and

time from a service provider's perspective. Let $\mathcal{R}^t = \{r_1^t, r_2^t, \ldots, r_n^t\}$ denote the set of request arrival rates from a specific grid area or road section C_i at time slot t, where r_i^t is the predicted number of requests originating from C_i. The complete scheduling decision is represented by a matrix $Y^t = \{y_{ij}^t\}$ subject to the constraints $\sum_{j=0}^{n} y_{ij}^t = 1$ for all i and $y_{ij}^t \leq x_j^t$ for all i, j, ensuring that requests are only scheduled to clouds with deployed replicas.

Through this formulation, we can systematically address the joint optimization of replica placement and request scheduling decisions to minimize service latency while respecting capacity constraints *Cap* and managing deployment costs effectively. This joint optimization problem is NP-hard (see Section III-D), making it computationally intractable for large-scale systems. To overcome this challenge, we decompose the problem into manageable subproblems that are solved sequentially in our proposed approach (Section V), striking a balance between solution quality and computational efficiency.

C. Response Latency Model

In edge computing systems, QoS is primarily measured by the request response latency, which consists of two major components: 1) computation latency and 2) communication latency. This section presents our model for quantifying these latency components in the context of replica placement and request scheduling.

Communication latency in our system has two key components: 1) wireless access latency between user devices and their nearest base stations, and 2) request routing latency between different edge clouds or from edge clouds to the central cloud. The wireless access component is largely independent of our replica placement decisions, as requests must first traverse the wireless link to reach the nearest base station regardless of where service replicas are deployed. Therefore, we model this wireless access latency as a constant μ for all requests.

For the request routing latency between clouds, we adopt a distance-based model where the latency between edge clouds S_i and S_j is proportional to the number of network hops between them [23], [51]. This is represented by d_{ij}, which corresponds to the geographical distance between the regions containing these edge clouds. This simplification is widely used in edge computing research as it captures the essential characteristics of network transmission delays while maintaining model tractability.

Regarding computation latency, we assume that the load on each replica does not exceed its maximum capacity *Cap*, ensuring no queuing delays. Under this condition, the processing time for each request can be approximated as constant for a given service type. This allows us to focus on the impact of replica placement and request scheduling on network latency without the additional complexity of variable computation times. Consequently, we can express the overall system response latency at time slot t as

$$\text{Latency}^t = \sum_{i=1}^{n} r_i^t \sum_{j=0}^{n} y_{ij}^t d_{ij} + \sum_{i=1}^{n} \mu r_i^t \tag{1}$$

where r_i^t represents the number of requests originating from area C_i during time slot t. The first term captures the total request routing latency across all network paths, weighted by the number of requests and the fraction of requests assigned to each path. The second term represents the aggregate wireless access latency for all requests, which remains constant for a given request distribution and is included for completeness.

D. Problem Formulation

Building on the previous sections, we now formulate the optimization problem. Our objective is twofold: 1) to minimize the overall response latency experienced by users; and 2) to minimize the number of replicas deployed, which directly impacts deployment costs. This dual-objective optimization problem for time slot t can be mathematically formulated as

$$\mathcal{P}1: \min \quad \{\sum_{i=1}^{n} r_i^t \sum_{j=0}^{n} y_{ij}^t d_{ij} + \sum_{i=1}^{n} \mu r_i^t, \sum_{j=1}^{n} x_j^t\} \tag{2}$$

$$\text{s.t.} \quad \sum_{j=0}^{n} y_{ij}^t = 1 \quad \forall C_i \in \mathcal{C} \tag{3}$$

$$y_{ij}^t \leq x_j^t \quad \forall C_i \in \mathcal{C} \forall S_j \in \mathcal{S}_+ \tag{4}$$

$$\sum_{i=1}^{n} y_{ij}^t r_i^t \leq \text{Cap} \quad \forall S_j \in \mathcal{S} \tag{5}$$

$$x_i^t \in \{0, 1\}, y_{ij}^t \in [0, 1] \quad \forall C_i \in \mathcal{C} \quad \forall S_j \in \mathcal{S}_+ \tag{6}$$

where the first term represents the total routing latency across all requests, excluding the constant wireless access latency since it is independent of placement decisions. The second term penalizes the number of replicas deployed at edge clouds, capturing the tradeoff between QoS and resource costs.

The problem $\mathcal{P}1$ is formulated as a constrained mixed-integer linear programmings (MILPs) problem, which is NP-hard due to the binary placement variables and their coupling with the continuous scheduling variables. The complexity of this problem grows exponentially with the number of edge clouds, making exact solutions computationally infeasible for large-scale systems. This computational challenge motivates our approach in Section V, where we decompose the problem into manageable subproblems that can be solved efficiently while maintaining solution quality.

IV. Spatio-Temporal Prediction of User Requests

To enable proactive replica placement, this section develops a predictive model for user request distributions by capturing their spatio-temporal dynamics. We define the spatio-temporal distribution of requests and propose tailored prediction methods for the grid-based and graph-based scenarios, leveraging historical data to forecast future demand.

A. Spatio-Temporal Distribution of User Requests

As outlined in Section III-A, the service area is partitioned into region set $\mathcal{C}$ (grid cells or road segments), and the timeline is discretized into time slot set $\mathcal{T}$. Request patterns in the service area exhibit two fundamental types of correlation.

1) *Temporal Correlation:* The request arrival rate at a specific location (grid area or road section) shows strong correlation with its historical values from neighboring time slots. This correlation captures daily and weekly patterns, trends, and seasonal variations in user behavior.
2) *Spatial Correlation:* Geographically proximate locations tend to exhibit similar request patterns. For instance, neighboring grid cells in urban areas or connected road sections in transportation networks often experience correlated demand due to similar user activities and movement patterns.

When a user sends a service request, it first reaches a nearby base station or RSU. These AP serve as data collection nodes that record request volumes without capturing sensitive user information, thus preserving privacy while providing valuable aggregate statistics. For each region C_g at time slot t, the request volume D_g^t is calculated as the number of requests received by its associated access point

$$D_g^t = \{|\text{req} \in \text{Req}_t || g_{\text{req}} = g\} \tag{7}$$

where Req_t is the set of all requests issued during time slot t, and g_{req} denotes the originating region of request req. To represent these spatial distributions uniformly, we define SD_t as the complete spatial distribution of requests during time slot t

$$\text{SD}_t = \{D_g^t | \forall g \in \mathcal{C}\} \tag{8}$$

and the full spatio-temporal distribution of requests across all time slots is then defined as

$$\text{STD} = \{\text{SD}_t | \forall t \in \mathcal{T}\}. \tag{9}$$

This comprehensive representation of request patterns captures both spatial variations and temporal dynamics. By analyzing historical STD data, we can identify recurring patterns, trends, and anomalies that inform our prediction models.

B. Prediction of User Requests Distribution

Effective edge computing systems must anticipate future user demand patterns to optimize replica placement proactively rather than reactively. In this section, we develop prediction models that forecast future request distributions by leveraging the spatio-temporal patterns captured in the historical *STD* data.

Objective: Given the historical spatio-temporal distribution $[\text{SD}_1, \text{SD}_2, \ldots, \text{SD}_{T_{\text{in}}}]$ over an input time window of length T_{in}, predict the future spatial distribution $\hat{\text{SD}}_{T_{\text{in}}+1}$ for the next time slot. We employ two different prediction approaches tailored to the distinct characteristics of grid-based and graph-based scenarios.

Prediction for Grid-Based Scenario: For the grid-based scenario, we employ a convolutional long short-term memorys (ConvLSTMs) approach, which extends traditional LSTM networks by incorporating convolutional operations to capture spatial dependencies alongside temporal patterns. Fig. 3 illustrates the prediction process, where the spatial distribution of requests from T_{in} historical time slots is sequentially processed

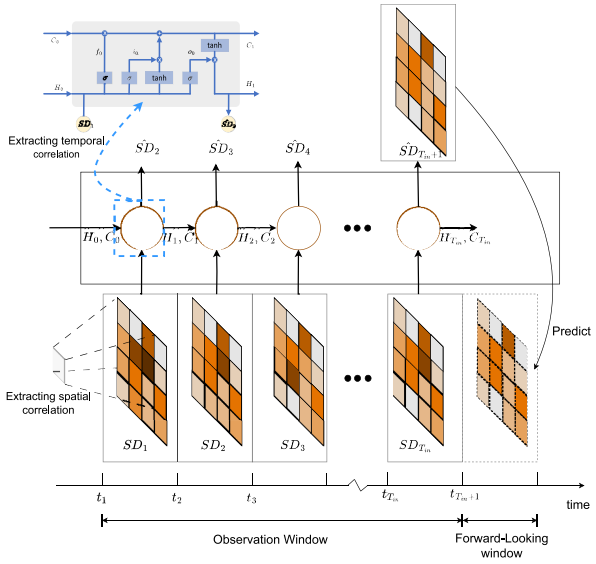

Fig. 3. Spatio-temporal request prediction process for grid-based scenario using ConvLSTM, illustrating the extraction of spatial and temporal correlations across an observation window to predict future request distributions over a forwardlooking window.

through the ConvLSTM network to produce the prediction $\hat{\text{SD}}_{T_{\text{in}}+1}$. At each time step, the network maintains two state vectors: the hidden state H_t representing the current output and the cell state C_t storing long-term memory. The core operations of the ConvLSTM at each time slot t are formalized in the following equations:

$$\begin{aligned} i_t &= \sigma(W_{xi} * \text{SD}_t + W_{hi} * H_{t-1} + b_i) \\ f_t &= \sigma\left(W_{xf} * \text{SD}_t + W_{hf} * H_{t-1} + b_f\right) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * \text{SD}_t + W_{hc} * H_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} * \text{SD}_t + W_{ho} * H_{t-1} + b_o) \\ H_t &= o_t \circ \tanh(C_t) \end{aligned} \tag{10}$$

where i_t, f_t, o_t represent the input, forget, and output gate coefficients respectively, controlling information flow. σ denotes the sigmoid activation function, which outputs values between 0 and 1. tanh is the hyperbolic tangent activation function, which outputs values between −1 and 1. $*$ denotes convolution, $\circ$ is the Hadamard product, and H_t and C_t are the hidden and cell states at time t. Unlike standard LSTM, ConvLSTM applies 2-D convolution to SD_t, extracting spatial features across neighboring cells while retaining temporal dependencies via gating mechanisms. In our implementation, we utilize the CMS-LSTM architecture [52], which has demonstrated strong performance in spatio-temporal prediction tasks.

Prediction for Graph-Based Scenario: For transportation networks represented as graphs, the standard convolution operation is no longer applicable since graph nodes have varying numbers of neighbors with no inherent ordering. Instead, we employ graph convolutional networks that can operate directly on graph-structured data. In our graph-based approach, spatial features are extracted using graph convolution operations that aggregate information from neighboring nodes. Specifically, we use Chebyshev polynomials as convolution

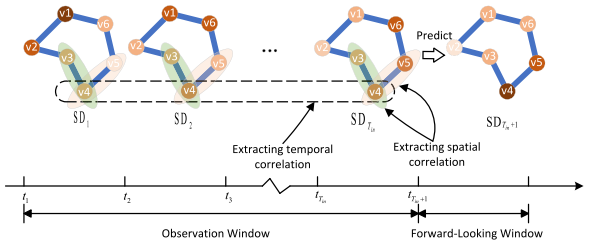

Fig. 4. Spatio-temporal request prediction process for graph-based scenario using graph convolution.

kernels to efficiently approximate spectral graph convolutions. For temporal feature extraction, we apply standard convolution operations along the time dimension. The spatio-temporal feature extraction module in the lth layer can be expressed as

$$\mathrm{SD}^{(l)} = \mathrm{ReLU}\Big(\Phi * (\mathrm{ReLU}(\Theta *_g \mathrm{SD}^{(l-1)}))\Big) \tag{11}$$

where $*_g$ represents the graph convolution operator with Chebyshev polynomial kernels Θ of order K, and Θ is the standard convolution kernel applied in the temporal dimension, and $\mathrm{SD}^{(l)}$ represents the feature maps at layer l. We set $K = 1$ to balance accuracy and computational cost, the feature extraction and the prediction schematic are shown in Fig. 4. Taking node v_4 as an example, the model first extracts spatial correlation features between this node and its neighbors using graph convolution at each time step. Then, temporal convolution is applied across time steps to capture temporal patterns. For our graph-based scenario, we implement the attention-based spatio-temporal graph convolutional networks (ASTGCNs) [53], which further enhances prediction accuracy by incorporating attention mechanisms that dynamically weight the importance of different spatial and temporal components.

V. Proactive Replica Placement Strategy

With the prediction methods established in the previous section, we can forecast the request arrival rate distribution for the next time slot, providing us with the set of expected request rates $\mathcal{R} = \{r_1^t, r_2^t, \ldots, r_n^t\}$. This predictive information enables us to make proactive replica placement decisions that anticipate future demand patterns rather than merely reacting to current conditions. The original optimization problem $\mathcal{P}1$ involves two sets of decision variables: 1) placement variables X^t and 2) scheduling variables Y^t. Solving this joint optimization problem directly is computationally intractable for large-scale systems due to its NP-hard nature. Therefore, we adopt a decomposition approach that breaks the problem into more manageable subproblems: First, we determine the number of replicas needed based on the predicted workload. Next, we solve for the optimal placement of these replicas. Finally, we determine the optimal request scheduling given the replica placement. This decomposition approach solves them sequentially to achieve a near-optimal solution.

A. Approximation Algorithm for Replica Placement

Determining the Number of Replicas: Deploying more replicas enhances reliability and processing capacity but incurs higher rental costs. To isolate the benefits of spatio-temporal prediction, we set the number of replicas k to the minimum required to handle the predicted workload

$$k = \lceil \frac{\sum_{i=1}^{n} r_i^t}{\mathrm{Cap}} \rceil \tag{12}$$

where Cap is the maximum processing capacity of each replica and $\sum_{i=1}^{n} r_i^t$ represents the total predicted request volume. The ceiling function ensures that we deploy enough replicas to handle the entire workload. With the number of replicas fixed at k, our original problem $\mathcal{P}1$ transforms into

$$\mathcal{P}2: \min \left(\sum_{i=1}^{n} r_i^t \sum_{j=0}^{n} y_{ij}^t d_{ij} + \sum_{i=1}^{n} \mu r_i^t \right) \tag{13}$$

$$\text{s.t. } (3), (4), (5), (6) \tag{14}$$

$$\sum_{j=1}^{n} x_j^t = k. \tag{15}$$

Here, this formulation adds constraint $\sum_{j=1}^{n} x_j^t = k$, which specifies that exactly k replicas must be deployed across edge clouds. It can be proven that $\mathcal{P}2$ is a generalization of the p-median problem, a well-known NP-hard facility location problem where p facilities must be placed to minimize the total distance between demand points and their nearest facilities.

Optimal Approach for Request Scheduling: Once the replica placement is determined, we need to decide how to schedule user requests to the deployed replicas. Given a fixed placement decision $\mathbf{X}^* = [x_0^*, x_1^*, \ldots, x_n^*]$, the problem $\mathcal{P}2$ simplifies to

$$\mathcal{P}3: \min \left(\sum_{i=1}^{n} r_i^t \sum_{j=0}^{n} y_{ij}'^t d_{ij} + \sum_{i=1}^{n} \mu r_i^t \right) \tag{16}$$

$$\text{s.t. } (3), (5), \tag{17}$$

$$y_{ij}'^t \le x_j^{*t} \quad \forall C_i \in \mathcal{C} \quad \forall S_j \in \mathcal{S}_+ \tag{18}$$

$$y_{ij}'^t \in [0,1] \quad \forall C_i \in \mathcal{C} \quad \forall S_j \in \mathcal{S}_+. \tag{19}$$

Here, problem $\mathcal{P}3$ is a standard Linear Programmings (LPs) problem, which can be solved in polynomial time using various LP solvers. In our implementation, we use Gurobi[1], a high-performance commercial optimization solver.

DRL-Based Placement (DRLP) Algorithm for Replica Placement: Determining the placement variables $\mathbf{X}^*$ in $\mathcal{P}3$ is core challenge due to its NP-hard nature and exponential action space $O(2^n)$. To address this efficiently in an online setting, we propose an enhanced DRLPs algorithm, leveraging the spatio-temporal predictions from Section IV to adaptively place k replicas across n edge clouds.

As shown in Fig. 5, to solve problem $\mathcal{P}3$ using DRL methods, take the spatial distribution of requests as the state S of the system, and action a is defined as the replica placement decision taken in the current state. To minimize the response latency, the reward r is defined as the inverse of the response latency under the corresponding placement decision, and then the reward is maximized when the latency is minimized. Since the action space reaches $O(2^n)$, a deep neural network is used

[1]http://www.gurobi.com/.

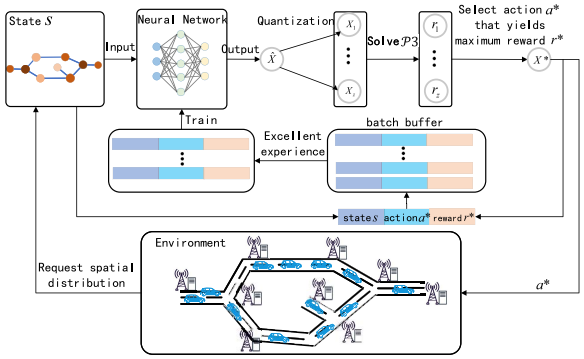

Fig. 5. Schematic of DRL-based replica placement, illustrating the process of training a neural network with state inputs, quantizing outputs, selecting optimal actions to maximize rewards, and interacting with an environment reflecting request spatial distribution in edge-cloud systems.

as the policy network, and the state s is fed into the policy network. The network will output a continuous action, and the agent makes a replica placement decision based on the policy network output.

1) *State s:* The spatial distribution of requests SD_t at the current time slot, representing the current state of the system.
2) *Action a:* Replica placement decision, determining which edge servers will host replicas, represented as a binary variable vector $X^t = [x_1^t, x_2^t, \ldots, x_n^t]$.
3) *Reward r:* Defined as the inverse of the total response latency under a given placement decision, i.e., $r = 1/(\sum_{i=1}^{n} r_i^t \sum_{j=0}^{n} y_{ij}^t d_{ij} + \sum_{i=1}^{n} \mu r_i^t)$.

The detailed training and decision process of the DRLP Algorithm is shown in Algorithm 1. The policy network is pretrained to avoid the cold start problem to obtain an acceptable network parameter θ_r^*. The regional central server, as an agent, regularly inputs the request spatial distribution as a state into the policy network, which will learn and make decisions online. A request spatial distribution state s is input to the network, and the network output is a continuous action decision *relaxed_pla*. As shown in line 2 of the algorithm, a quantization operation (Quantization) is performed on *relaxed_pla* to generate a discrete placement decision.

We design a quantization operation, i.e., the k-edge clouds with the highest probability value in *relaxed_pla* are selected to form the initial placement action. Then the initial action is mutated several times based on the idea of mutation in the genetic algorithm to generate the set of feasible actions. Then, we solve the problem $\mathcal{P}3$ for all the candidate placement actions in the generated *action_list* to obtain the corresponding reward values (lines 4–6), select the placement solution with the largest reward as action *action*, and store the state s, action *action* and the corresponding reward *reward* in the *batch buffer*. As shown in lines 10–14 of the algorithm, when the buffer is full, the network parameters are updated with an elite strategy, i.e., the 30% of data with the highest rewards in the *batch buffer* is selected to train the policy network. The policy network parameters are updated using the cross-entropy method. This process is carried out continuously over time,

Algorithm 1 DRL-Based Replica Placement Algorithm

Require: Pre-trained policy network $\varphi_{\theta_r^*}$, State s
Ensure: Replica placement scheme for state s
1: $relaxed\_pla \leftarrow \varphi_{\theta_r^*}(s)$
2: $action\_list \leftarrow Quantization(relaxed\_pla)$
3: $reward\_list \leftarrow \phi$
4: **for** a **in** $action\_list$ **do**
5: $\quad reward\_list \leftarrow reward\_list \cup Reward(a)$
6: **end for**
7: $action \leftarrow action\_list[argmax(reward\_list)]$
8: $reward \leftarrow reward\_list[argmax(reward\_list)]$
9: Push $(s, action, reward)$ into *batch buffer*
10: **if** *batch buffer* is full **then**
11: $\quad$ Select the 30% of data in the batch buffer with the highest rewards to form $\{(s, action, reward)\}$
12: $\quad$ Train policy network with $\{(s, action, reward)\}$
13: $\quad$ Empty *batch buffer*
14: **end if**
15: **return** *action*

and the strategy network parameters are updated periodically while making decisions online.

B. Forward-Looking Replica Placement Algorithm

Algorithm 2 is the spatio-temporal prediction-based replica placement algorithm [spatio-temporal prediction-based replica placement approach (STPRPA)] proposed in this article, a forward-looking replica placement strategy. The request prediction model is trained in the central cloud with historical data, and the pretrained spatio-temporal prediction model Φ_θ and the pretrained policy network $\varphi_{\theta_r^*}$ are deployed on the regional central server. The regional center server records the spatial distribution of the average request arrival rate for each time slot, indicating the distribution of demand for the service. As shown in line 3 of the algorithm, if the observed number of historical time slots is less than the input time window length, a replica placement decision is made based on the observation of the previous time slot, i.e., responsive replica placement. Here, *DRLP*(SD) is used to indicate that when facing the requested spatial distribution SD, Algorithm 1 is invoked to obtain a replica placement scheme corresponding to SD. Otherwise, the request space distribution for this time slot is predicted at the beginning of each time slot (line 6), and then a forward-looking replica placement decision is made based on the prediction result. This prediction of future user request information guides making the proper replica placement decision and improves replica placement efficiency.

At regular intervals (one week, for example), the updated average request arrival rate records are uploaded to the central cloud for incremental training. The model with updated parameters is deployed to the regional central server to ensure prediction performance. The policy network learns and updates online during the decision-making process.

For Algorithm 2, the computational complexity is concentrated on calling Algorithm 1 to obtain the replica placement scheme. In this case, DRL methods make the fast inference

Algorithm 2 STPRPA

Input: Pre-trained prediction model Φ_θ,Pre-trained policy network $\varphi_{\theta_r^*}$

Output: Placement schemes F for every time slot

```
Empty F
while t < T do
    if t < T_in then
        p_scheme ← DRLP(SD_{t−1})   ▷ Responsive replica placement
    else
        SD̂_t ← Φ_θ(SD_{t−T_in,...,SD_{t−1}})
        p_scheme ← DRLP(SD̂_t)       ▷ Forward-Looking replica placement
    end if
    F ← F ∪ p_scheme
end while
return F
```

online when obtaining the placement action. Thus the main computational complexity lies in solving for the action rewards in line 5 of Algorithm 1, which can be solved quickly in polynomial time complexity by the Gurobi solver. Thus, STPRPA can obtain a near-optimal replica placement scheme with forward-looking perspective in polynomial time. STPRPA achieves replica placement via proactive forecasting, and its computational complexity is dominated by the spatio-temporal prediction stage, with the remainder incurring the standard costs of DRL. For CMS-LSTM on a $G \times G$ grid, each 2-D convolution requires $\mathcal{O}(G^2k^2)$ operations per layer and the LSTM gates require $\mathcal{O}(G^2h^2)$ operations (where k is the kernel size and h is the hidden-state width). For ASTGCN on a graph with N nodes and E edges, graph convolutions cost $\mathcal{O}(EK)$ operations per layer (with polynomial order K), and the attention blocks cost $\mathcal{O}(Nh^2)$.

VI. Performance Evaluation

In this section, we rigorously evaluate the proposed STPRPAs using real-world datasets across the grid-based and graph-based scenarios outlined in Section III-A. We compare STPRPA against three representative baseline approaches, focusing on QoSs and service provider profitability. In addition, we provide an in-depth analysis of performance variations, particularly between weekdays and weekends.

A. Experimental Settings

Datasets: For the grid-based application service scenario, we utilize the public bicycle demand dataset from New York City (CitiBikeNYC[2]). The original dataset undergoes preprocessing, where the geographic area is partitioned into a 12×12 grid. Each grid region is equivalent to an edge cloud covering approximately 1km^2, providing network access and computational services to users within its coverage area. For the graph-based application service scenario, we employ the California highway traffic dataset PeMSD8. In this scenario, 170 edge clouds are formed by detectors distributed along the highway network, providing network access and services to adjacent road segments. This dataset reflects vehicular IoT applications with graph-structured spatial dependencies. Both datasets are segmented into hourly time slots and split into training, validation, and test sets at an 8:1:1 ratio, ensuring robust model evaluation. These traffic-related datasets accurately mirror the spatio-temporal dynamics of edge computing demands, as validated in prior studies [23], [54], [55].

Comparison Methods: We benchmark STPRPA against three distinct approaches.

1) *Responsive Placements (RPs):* This method utilizes the average request arrival rate from the previous time slot as the basis for decision-making at the beginning of each subsequent time slot. RP dynamically adjusts replica placement by reactively following request patterns without employing prediction mechanisms [11].
2) *LSTM-Based Replica Placements (LRPs):* This method employs the widely adopted LSTMs network for temporal prediction, without considering spatial correlations. LSTM predicts the average request arrival rate for each grid area or road section, with replica placement performed based on these predicted values.
3) *Perfect Prediction-Based Placement (Ideal):* This represents an ideal scenario where perfect prediction of request arrival rates for future time slots is assumed possible. This approach is included to assess the performance gap between our proposed forward-looking replica placement strategy and the theoretical optimal case.

Training Details: For LSTM architecture, each hourly snapshot is flattened from a 12×12 demand grid and embedded to 64 dimensions via a fully connected layer with ReLU. The embedding sequence then passes through two stacked LSTM layers (hidden size $h = 128$, dropout 0.2 between layers). The model is trained using the Adam optimizer (lr $= 0.0001$) with a batch size of 64.

Evaluation Setup: We assume that each replica possesses identical processing capacity, set at 90 requests per time slot [11]. Following [23], the wireless communication delay between user devices and base stations is considered constant (μ) and set to 0, which does not impact our experimental results. When a request is forwarded to the central cloud, the response latency is configured at three times the maximum response latency for edge processing, reflecting the higher latency of cloud-based processing.

B. Performance Metrics

In the experiments, we use two metrics to evaluate the performance of all approaches:

QoS: A primary objective for service providers is to deliver high QoS. In our evaluation, we define the QoS for each time slot as the overall request response time within that slot. This metric aligns with the optimization goal defined in (1), where lower values indicate better performance.

Profit of Service Provider: The ultimate objective for service providers is to generate profit from user access to their

[2]https://ride.citibikenyc.com/system-data

TABLE III
COMPARISON OF DIFFERENT PREDICTION METHODS

Dataset	Method	RMSE	MAE	MAPE
CitiBikeNYC	ARIMA	15.15	5.67	19.9%
	LSTM	8.02	3.11	11.2%
	STPRPA (Ours)	**6.79**	**2.69**	**9.3%**
PeMSD8	ARIMA	40.51	23.04	24.7%
	LSTM	34.08	21.69	22.6%
	STPRPA (Ours)	**28.02**	**18.12**	**17.8%**

services [11], [56]. Profit is calculated as revenue minus deployment costs. While more replicas generally result in higher service quality, they also incur greater deployment expenses. One of the key objectives of our forward-looking replica placement approach is to optimize the tradeoff between service quality and deployment overhead.

Regarding revenue calculation, we adopt the service level agreements (SLAs) model utilized in [4]. If the response latency(q) for user request q is below l_{ideal}, the user receives perfect service quality and is charged r_{ideal} for access. If the response latency falls between l_{ideal} and $l_{\max}$, revenue decreases linearly proportional to service quality degradation. If response latency exceeds $l_{\max}$, the service provider receives only the minimum revenue $r_{\min}$. Larger values for l_{ideal} and $l_{\max}$ indicate that the service has lower QoS requirements and can tolerate higher response latency. These parameters can be adjusted to accommodate service types with varying sensitivity to latency. The revenue obtained from serving request q can be expressed as follows:

$$\text{Revenue}(q) = \begin{cases} r_{\text{ideal}} \ , \text{latency}(q) \le l_{\text{ideal}} \\ r_{\min} \ , \text{latency}(q) > l_{\max} \\ r_{\text{ideal}} - \gamma(r_{\text{ideal}} - r_{\min}), \text{otherwise} \end{cases} \tag{20}$$

where, $\gamma = [(\text{latency}(q) - l_{\text{ideal}})/(l_{\max} - l_{\text{ideal}})]$. In our simulations, the full revenue and deployment cost are configured according to [11], i.e., $r_{\text{ideal}} = 0.044$ \$ per 1K requests, and $r_{\min}$ is set to $r_{\text{ideal}} * 0.5$. Physical resources are rented on-demand, with prices referencing Amazon's cloud instances. Pick the cloud instance named *c5ad.8xlarge* and rent it for 1.376 \$ per hour[3]. The ideal latency l_{ideal} is set to 5 ms, and the maximum tolerable latency $l_{\max}$ is set to the maximum response latency that the request can be processed in the edge cloud, i.e., only the minimum revenue $r_{\min}$ can be obtained if it is processed in the central cloud.

C. Analysis of Spatio-Temporal Prediction Results

We select ARIMA and LSTM as baseline prediction methods for our proactive replica placement study based on their widespread adoption in recent edge computing literature [12], [25], [28]. ARIMA represents traditional statistical time-series modeling, while LSTM exemplifies DL approaches that capture temporal dependencies. Root mean square errors (RMSEs), mean absolute errors (MAEs), and mean absolute percentage errors (MAPEs) are employed as evaluation metrics, where RMSE emphasizes larger errors through its quadratic nature, MAE represents the average magnitude of errors without considering their direction, and MAPE measures the average relative error as a percentage, providing insight into the prediction accuracy relative to the actual values.

[3]https://aws.amazon.com/cn/ec2/pricing/on-demand/

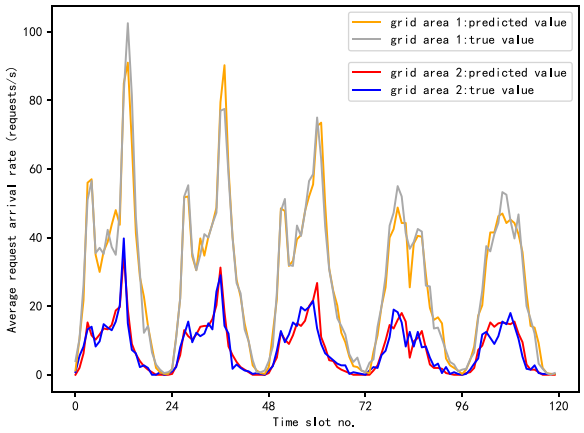

Fig. 6. Predicted versus Actual request arrival rates for two grid areas.

The comparative prediction accuracy of ARIMA, LSTM, and our proposed STPRPA is presented in Table III. For the CitiBikeNYC dataset with 144 grid regions, our STPRPA achieves the best performance with an RMSE of 6.79, MAE of 2.69, and MAPE of 9.3%. Similarly, for the PeMSD8 dataset with 170 edge clouds, STPRPA further improves them to an RMSE of 28.02, MAE of 18.12, and MAPE of 17.8%. The intrinsic spatio-temporal complexity of user request distributions poses significant challenges for traditional time-series prediction methods like ARIMA, resulting in considerably higher error rates compared to DL approaches. This performance gap stems from ARIMA's limited capability to model high-dimensional nonlinear features inherent in edge computing request patterns. In contrast, both the LSTM and our STPRPA approach achieve substantially better prediction performance by leveraging deep neural architectures. Specifically, STPRPA demonstrates superior accuracy by simultaneously extracting dynamic correlation features of user requests in both temporal and spatial dimensions, achieving RMSEs of 6.79 and 28.02 on the CitiBikeNYC (144 grid regions) and PeMSD8 (170 edge clouds) datasets, respectively. This enhanced prediction accuracy provides a more reliable foundation for subsequent replica placement decisions, potentially leading to more efficient resource utilization and improved service quality. Since the LSTM baseline was already superior to ARIMA, for subsequent experiments we kept only the more representative LSTM to maintain focus.

Fig. 6 illustrates the comparison between predicted and actual request arrival rates for two representative grid areas selected based on their distinct temporal patterns (one exhibiting regular peak-valley fluctuations and another showing more irregular demand). The visualization demonstrates that our spatio-temporal predictive learning method effectively captures both the overall trend and fine-grained fluctuations in request arrival rates across diverse grid areas. This accurate prediction of spatial demand heterogeneity is crucial for

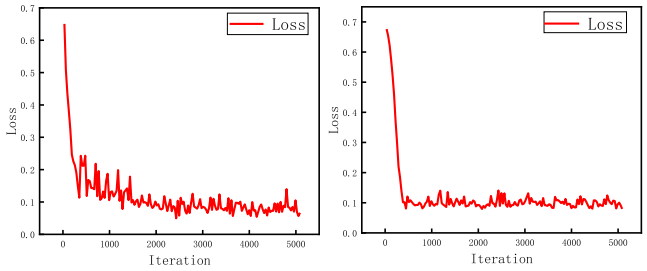

Fig. 7. Convergence of our STPRPA during training. (a) Grid-based scenario. (b) Graph-based scenario.

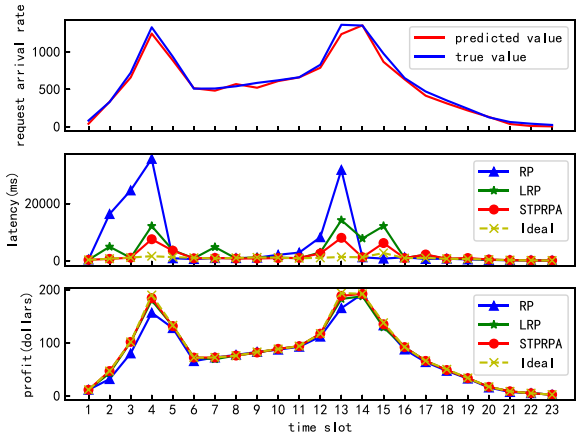

Fig. 8. Latency and profit comparison of placement strategies over a workday in grid-based scenario.

optimizing replica placement in edge computing environments where resources are constrained and demand varies significantly across locations.

D. Analysis of Replica Placement Results

Convergence Analysis of DRL Model: Fig. 7 illustrates the evolution of the loss value during the pretraining process of our DRL model, with a batch size of 64 and learning rate of 0.0001. During the initial 1000 iterations, we observe a consistent decrease in loss values, indicating that the agent effectively learns improved strategies through environmental interaction, progressively reducing the gap between policy network outputs and exemplary strategies in the batch buffer. After approximately 1000 iterations, the loss value stabilizes at a consistently low level in both application scenarios, signifying model convergence and the acquisition of near-optimal decision-making capabilities. This convergence stability is particularly important in edge computing environments where consistent performance under varying load conditions is essential for reliable service delivery.

Temporal Variation Analysis Within the Same Day: Edge computing application requests demonstrate significant periodicity aligned with human activity patterns. To comprehensively evaluate our approach under different temporal patterns, we select five consecutive days from the test set for replica placement simulation experiments, comprising three weekdays and two weekend days, thus capturing both workday and leisure request patterns.

Fig. 8 presents detailed experimental results for each time slot during a representative weekday in the grid-based application service scenario. The three vertically aligned subplots share a common horizontal axis representing 24 hourly time slots. The top subplot displays both predicted and actual request burden patterns across the entire region, while the bottom two subplots illustrate the corresponding latency and profit metrics for each time slot under different placement strategies.

The selected day represents a typical workday pattern with two distinctive peaks in request arrival rates, corresponding to morning and evening commuting periods—a common challenge for edge resource allocation systems. When application demand exhibits an upward trend (e.g., time slots 1 through 4), the RPs approach consistently allocates fewer replicas than required due to its inherent reactive nature, resulting in insufficient capacity at edge servers. This deficiency necessitates request redirection to the central cloud, significantly increasing overall latency compared to other approaches. Consequently, service provider profits diminish substantially due to the inability to deliver high-quality service during peak demand periods. Conversely, during demand reduction phases (e.g., time slots 14 through 23), RP typically overprovisioning replicas due to the same temporal lag in response. While this excessive replica deployment produces marginally lower latency than the ideal case, it results in inefficient resource utilization and subsequently lower profitability due to unnecessary deployment costs.

In contrast, prediction-based approaches can proactively respond to anticipated changes in service demand distribution. As illustrated in Fig. 8, the Ideal approach—with perfect prediction capabilities—consistently makes optimal replica placement decisions that precisely match actual service demand regardless of demand fluctuations, maintaining minimal response latency while maximizing service provider profits. In real-world implementations, however, prediction accuracy becomes the critical factor. Our results demonstrate that STPRPA achieves near-ideal performance in both QoS and profitability metrics due to its highly accurate spatio-temporal predictions of future request patterns. The LSTM-based approach LRP, while superior to RP, shows inferior performance compared to STPRPA due to its limited consideration of spatial correlation factors. This prediction accuracy deficit propagates to subsequent replica placement decisions, resulting in increased response latency and reduced profit. This comparison clearly illustrates the crucial importance of incorporating both spatial and temporal dimensions in edge computing replica placement strategies.

Performance Analysis Across Multiple Days: Figs. 9(a) and 10(a) present the comparative performance across five consecutive days in the grid-based scenario. Our STPRPA approach achieves an average overall response latency approximately 57.64% and 36.13% lower than RP and LRP, respectively, while concurrently delivering average profit improvements of approximately 4.85% and 1.14%. These significant improvements translate to potential enhancement

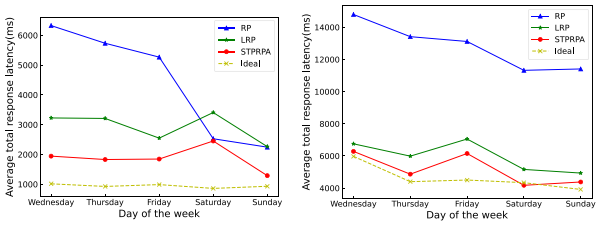

Fig. 9. Average latency comparison of methods over five days in grid-based and graph-based scenario. (a) Grid-based Scenario. (b) Graph-based Scenario.

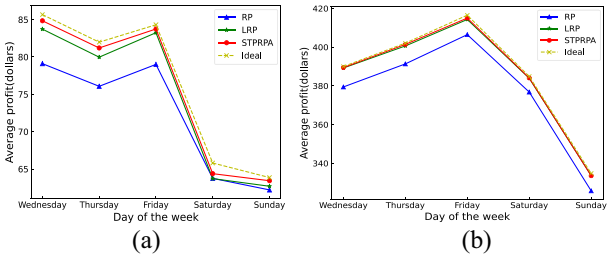

Fig. 10. Average service provider profit comparison of methods over five days in grid-based and graph-based Scenario. (a) Grid-based Scenario. (b) Graph-based Scenario.

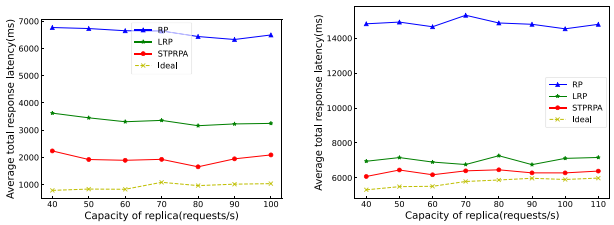

Fig. 11. Impact of replica capacity on average response latency in grid-based and graph-based scenarios. (a) Grid-based Scenario. (b) Graph-based Scenario.

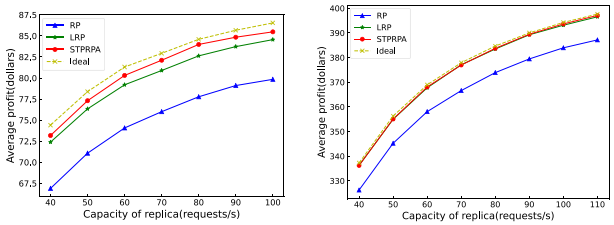

Fig. 12. Impact of replica capacity on service provider profit in grid-based and graph-based scenarios. (a) Grid-based Scenario. (b) Graph-based Scenario.

of user experience and additional revenue of thousands of dollars daily in large-scale edge deployments. The superior performance stems from STPRPA's comprehensive integration of spatio-temporal correlation analysis, which enables more appropriate and timely replica placement decisions.

Notably, in the grid-based application scenario, prediction-based approaches deliver particularly substantial benefits compared to reactive methods RP on weekdays, where the CitiBikeNYC dataset exhibits two distinct service request peaks, representing relatively rapid demand fluctuations. In contrast, weekend request patterns demonstrate more gradual variations, narrowing the performance gap between reactive and proactive placement approaches. This temporal pattern difference explains the unexpected observation on Saturday, where RP momentarily outperforms LRP in terms of average response latency—a phenomenon attributable to the more stable demand pattern and the imperfect predictions from the temporal-only model.

Figs. 9(b) and 10(b) illustrate the performance comparison in the graph-based scenario, where STPRPA achieves approximately 59.62% and 13.49% lower average overall response latency than RP and LRP, respectively, along with profit improvements of approximately 2.32% and 0.12%. An interesting observation emerges on Saturday in Fig. 9(b), where STPRPA's average response latency slightly undercuts the Ideal value. This occurs when prediction values exceed actual values, leading to the deployment of additional replicas beyond immediate requirements. These excess replicas enhance request proximity processing capabilities, resulting in lower-than-ideal average response latency but at increased deployment costs. Nevertheless, the overall profit remains remarkably close to optimal values due to the high-prediction accuracy of STPRPA, with the additional infrastructure costs substantially offset by increased revenue from enhanced service quality. This observation highlights an important consideration for service providers: the strategic tradeoff between deployment costs and user experience quality can be optimized based on application-specific requirements and business objectives.

E. Impact of Replica Processing Capacity on System Performance

To evaluate system sensitivity to hardware specifications, we analyze the effect of varying replica processing capacities on overall performance. The processing capacity range of 40-110 requests/s was selected to represent typical edge server configurations from resource-constrained IoT gateways (40 requests/s) to more powerful edge servers (110 requests/s) based on industry deployments. Figs. 11 and 12 illustrate the effect of replica processing capacity on response latency and profit metrics, respectively. Fig. 11 demonstrates that our proposed replica placement method ensures stable overall response latency across various replica processing capacities, indicating robust performance across heterogeneous edge computing environments. From a business perspective, Fig. 12 reveals a consistent trend: higher replica capacity consistently generates greater profit for service providers, assuming constant rental pricing. This relationship stems from the enhanced ability to process more requests locally at the edge, reducing costly central cloud offloading while simultaneously improving user experience through lower latency. This insight provides valuable guidance for edge infrastructure investment decisions, suggesting that investing in higher-capacity edge nodes may yield better returns than deploying more numerous lower-capacity nodes, particularly in high-demand regions identified through our spatio-temporal analysis.

TABLE IV
ORTHOGONAL EXPERIMENTS WITH THE SOTA METHODS

Dataset	Method	Latency (s)	Profit ($)
CitiBikeNYC	PDQN [44]	3.1	79.5
	STPRPA-PDQN (Ours)	**1.5**	**87.2**
	BRPS [45]	2.7	81.3
	STPRPA-BRPS (Ours)	**1.3**	**88.5**
PeMSD8	PDQN [44]	8.4	378.9
	STPRPA-PDQN (Ours)	**6.2**	**391.6**
	BRPS [45]	6.7	390.2
	STPRPA-BRPS (Ours)	**5.1**	**396.9**

F. Orthogonality With SOTA RL-Based Methods

To demonstrate the orthogonality of our proactive prediction module with RL-based replica placement methods, we selected two recent SOTA approaches for integration: PDQN [44] and BRPS [45], both of which directly learn replica placement strategies from observations of lagged workload. In our hybrid framework, the request distributions predicted by STPRPA are fed as initial states into the RL agents, thereby extending their state space with a proactive perspective. The RL agents then optimize replica placement actions based on this enriched state, leveraging their respective reward functions. As shown in Table IV, for the CitiBikeNYC dataset, the STPRPA-PDQN hybrid scheme achieves an average latency of 1.5 s and a profit of 87.2 $, compared to 3.1 s and 79.5 $ for the standalone PDQN. Similarly, STPRPA-BRPS reduces latency to 1.3 s and increases profit to 88.5 $. Consistent performance gains are observed on the PeMSD8 dataset. These results indicate that integrating STPRPA with SOTA RL methods can enhance performance by leveraging proactive spatio-temporal predictions, affirming the orthogonality of our approach.

VII. CONCLUSION

In this article, we addressed the challenge of service replica placement in an edge-cloud collaborative environment by explicitly modeling the spatio-temporal dynamics of user requests from a service provider's perspective. We proposed a novel forward-looking strategy, STPRPA, integrating spatio-temporal predictive learning with DRL to proactively place replicas based on anticipated demand. Our approach leverages convolutional and graph-based neural networks (CMS-LSTM and ASTGCN) to achieve high-prediction accuracy, coupled with a DRL-driven placement algorithm to optimize latency and resource utilization. Extensive trace-driven simulations on real-world datasets across grid-based and graph-based IoT scenarios demonstrate that STPRPA reduces average response latency by up to 59.6% compared to reactive methods and enhances service provider profitability by approximately 4.85% over temporal-only prediction baselines. These gains are most pronounced during periods of rapid request fluctuations, underscoring the value of spatio-temporal awareness in edge computing. The main limitation of our approach is its reliance on ample historical data to generate accurate proactive forecasts, which becomes especially challenging in newly deployed MEC regions where such data are scarce. Moreover, the current objective formulation optimizes only service latency and provider profit. In future work, we plan to refine our spatio-temporal prediction models by exploring hybrid architectures (e.g., transformer-based methods) to capture longer-term dependencies and improve scalability for ultradense edge deployments, and the reward function should be extended to balance multiple objectives.

REFERENCES

[1] Q. Guo, F. Tang, and N. Kato, "Resource allocation for aerial assisted digital twin edge mobile network," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3070–3079, Oct. 2023.

[2] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2021.

[3] Q. Guo, F. Tang, and N. Kato, "Federated reinforcement learning-based resource allocation for D2D-aided digital twin edge networks in 6G industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 7228–7236, May 2023.

[4] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 55–67, Jan. 2022.

[5] Q. Guo, F. Tang, and N. Kato, "Hybrid routing in FSO/RF space-air-ground integrated network," in *Proc. IEEE Global Commun. Conf.*, 2023, pp. 6585–6590.

[6] C. Li, L. Zhu, and Y. Luo, "Latency-aware content caching and cost-aware migration in SDN based on MEC," *Wireless Netw.*, vol. 27, no. 8, pp. 5329–5349, 2021.

[7] C. An and C. Wu, "Traffic big data assisted V2X communications toward smart transportation," *Wireless Netw.*, vol. 26, no. 3, pp. 1601–1610, 2020.

[8] L. Yang, M. Yuan, W. Wang, Q. Zhang, and J. Zeng, "Apps on the move: A fine-grained analysis of usage behavior of mobile apps," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[9] T. Li, Y. Li, T. Xia, and P. Hui, "Finding spatiotemporal patterns of mobile application usage," *IEEE Trans. Netw. Sci. Eng.*, early access, Nov. 30, 2021, doi: 10.1109/TNSE.2021.3131194.

[10] H. Sami, A. Mourad, H. Otrok, and J. Bentahar, "Demand-driven deep reinforcement learning for scalable fog and service placement," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2671–2684, Sep./Oct. 2022.

[11] W.-C. Chang and P.-C. Wang, "Adaptive replication for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2422–2432, Nov. 2018.

[12] J. Wang, H. Chen, F. Zhou, M. Sun, Z. Huang, and Z. Zhang, "A-DECS: Enhanced collaborative edge–edge data storage service for edge computing with adaptive prediction," *Comput. Netw.*, vol. 193, Jul. 2021, Art. no. 108087.

[13] V. Farhadi et al., "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 779–792, Apr. 2021.

[14] A. Aral and T. Ovatman, "A decentralized replica placement algorithm for edge computing," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 516–529, Jun. 2018.

[15] C. Nguyen, C. Klein, and E. Elmroth, "Multivariate LSTM-based location-aware workload prediction for edge data centers," in *Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, 2019, pp. 341–350.

[16] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio–temporal edge service placement: A bandit learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.

[17] C. Huang, G. Chen, P. Xiao, Y. Xiao, Z. Han, and J. A. Chambers, "Joint offloading and resource allocation for hybrid cloud and edge computing in SAGINs: A decision assisted hybrid action space deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 5, pp. 1029–1043, May 2024.

[18] M. Mansouri, M. Eskandari, Y. Asadi, and A. Savkin, "A cloud-fog computing framework for real-time energy management in multi-microgrid system utilizing deep reinforcement learning," *J. Energy Storage*, vol. 97, Sep. 2024, Art. no. 112912.

[19] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.

[20] Y. Wang and X. Yang, "Research on edge computing and cloud collaborative resource scheduling optimization based on deep reinforcement learning," 2025, *arXiv:2502.18773*.

[21] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, "Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *Future Gener. Comput. Syst.*, vol. 152, pp. 55–69, Mar. 2024.

[22] Q. Guo, F. Tang, and N. Kato, "Routing for space-air-ground integrated network with GAN-powered deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 11, no. 2, pp. 914–922, Apr. 2025.

[23] Y. Wang et al., "Towards cost-effective service migration in mobile edge: A Q-learning approach," *J. Parallel Distrib. Comput.*, vol. 146, pp. 175–188, Dec. 2020.

[24] C. Li, Y. Wang, H. Tang, and Y. Luo, "Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud," *Future Gener. Comput. Syst.*, vol. 100, pp. 921–937, Nov. 2019.

[25] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Dynamic service placement and load distribution in edge computing," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, 2020, pp. 1–9.

[26] C. Li, M. Song, C. Yu, and Y. Luo, "Mobility and marginal gain based content caching and placement for cooperative edge-cloud computing," *Inf. Sci.*, vol. 548, pp. 153–176, Feb. 2021.

[27] C. Li, M. Song, M. Zhang, and Y. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *J. Parallel Distrib. Comput.*, vol. 143, pp. 107–128, Sep. 2020.

[28] Z. Zaman, S. Rahman, and M. Naznin, "Novel approaches for VNF requirement prediction using DNN and LSTM," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[29] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (SAGIN)," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 276–289, Jan. 2022.

[30] Q. Xu, Z. Su, Q. Zheng, M. Luo, B. Dong, and K. Zhang, "Game theoretical secure caching scheme in multihoming edge computing-enabled heterogeneous networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4536–4546, Jun. 2019.

[31] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 2027–2057, 3rd Quart., 2021.

[32] Z. Chen, J. Hu, G. Min, and X. Chen, "Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization," *Concurrency Comput. Pract. Exp.*, vol. 33, no. 8, 2021, Art. no. e5413.

[33] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5595–5608, Nov. 2021.

[34] Y. Zhang, W. Liang, Z. Xu, and X. Jia, "Mobility-aware service provisioning in edge computing via digital twin replica placements," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11295–11311, Dec. 2024.

[35] L. Chen, Y. Bai, P. Zhou, Y. Li, Z. Qu, and J. Xu, "On adaptive edge microservice placement: A reinforcement learning approach endowed with graph comprehension," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11144–11158, Dec. 2024.

[36] A. Xu et al., "TransEdge: Task offloading with GNN and DRL in edge computing-enabled transportation systems," *IEEE Internet Things J.*, vol. 11, no. 23, pp. 38151–38166, Dec. 2024.

[37] A. Rago, G. Piro, G. Boggia, and P. Dini, "Anticipatory allocation of communication and computational resources at the edge using spatio-temporal dynamics of mobile users," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4548–4562, Dec. 2021.

[38] Q. Wu, X. Chen, Z. Zhou, L. Chen, and J. Zhang, "Deep reinforcement learning with spatio-temporal traffic forecasting for data-driven base station sleep control," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 935–948, Apr. 2021.

[39] Q. Guo, N. Kato, and F. Tang, "Energy efficient routing for FSO-RF space-air-ground integrated network: A deep reinforcement learning approach," in *Proc. 8th IEEE Int. Conf. Netw. Intell. Digit. Content (ICNIDC)*, 2023, pp. 254–258.

[40] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward deep Q-network-based resource allocation in Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9138–9150, Jun. 2022.

[41] J. Yang, Q. Yuan, S. Chen, H. He, X. Jiang, and X. Tan, "Cooperative task offloading for mobile edge computing based on multi-agent deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3205–3219, Sep. 2023.

[42] X. Gao, Y. Sun, H. Chen, X. Xu, and S. Cui, "Joint computing, pushing, and caching optimization for mobile-edge computing networks via soft actor–critic learning," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 9269–9281, Mar. 2024.

[43] T. Zhang, C. Xu, B. Zhang, X. Li, X. Kuang, and L. A. Grieco, "Towards attack-resistant service function chain migration: A model-based adaptive proximal policy optimization approach," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 6, pp. 4913–4927, Nov./Dec. 2023.

[44] T. Liu, S. Ni, X. Li, Y. Zhu, L. Kong, and Y. Yang, "Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3870–3881, Jul. 2023.

[45] M. Zheng, X. Du, Z. Lu, and Q. Duan, "A balanced and reliable data replica placement scheme based on reinforcement learning in edge–cloud environments," *Future Gener. Comput. Syst.*, vol. 155, pp. 132–145, Jun. 2024.

[46] M. Reiss-Mirzaei, M. Ghobaei-Arani, and L. Esmaeili, "A review on the edge caching mechanisms in the mobile edge computing: A social-aware perspective," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100690.

[47] Y. Jin, J. Liu, F. Wang, and S. Cui, "Ebublio: Edge assisted multi-user 360-degree video streaming," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15408–15419, Sep. 2023.

[48] H. Wang, J. Xie, and M. M. A. Muslam, "FAIR: Towards impartial resource allocation for intelligent vehicles with automotive edge computing," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1971–1982, Feb. 2023.

[49] W. Fan et al., "Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4277–4292, Apr. 2023.

[50] S. Yang, J. Tan, T. Lei, and B. Linares-Barranco, "Smart traffic navigation system for fault-tolerant edge computing of Internet of Vehicle in intelligent transportation gateway," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 13011–13022, Nov. 2023.

[51] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 3836–3851, Nov. 2022.

[52] Z. Chai, C. Yuan, Z. Lin, and Y. Bai, "CMS-LSTM: Context-embedding and multi-scale spatiotemporal-expression LSTM for video prediction," 2021, *arXiv:2102.03586*.

[53] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 922–929.

[54] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020.

[55] X. Zhao, Y. Shi, and S. Chen, "MAESP: Mobility aware edge service placement in mobile edge networks," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107435.

[56] X. Xia et al., "Graph-based data caching optimization for edge computing," *Future Gener. Comput. Syst.*, vol. 113, pp. 228–239, Dec. 2020.

Hao Zheng (Graduate Student Member, IEEE) received the M.S. degree from Central South University, Changsha, China, in 2021, where he is currently pursuing the Ph.D. degree in the research group of Zhigang Hu with the School of Computer Science and Engineering.

His research interests include computer vision and remote sensing, federated learning, and edge computing.

Zhigang Hu received the B.S., M.S., and Ph.D. degrees from Central South University (CSU), Changsha, China, in 1985, 1988, and 2002, respectively.

In 2002, he joined CSU, where he is a Professor with the School of Computer Science and Engineering. He has published over 200 research papers. His research interests include radar signal processing and classification/recognition, high-performance computing, and cloud computing.

Liu Yang received the B.S., M.S., and Ph.D. degrees from Central South University (CSU), Changsha, China, in 2002, 2005, and 2011, respectively.

She is currently an Associate Professor with the School of Computer Science and Engineering, CSU. Her research interests include knowledge graph, deep learning, and software metrics.

Dr. Yang is a member of ACM and CCF.

Aikun Xu received the M.S. degree from Central South University, Changsha, China, in 2022, where he is currently pursuing the Ph.D. degree.

His research interests include deep learning, graph neural network, deep reinforcement learning, scheduling, electric vehicles, and edge computing.

Meiguang Zheng received the B.S. and Ph.D. degrees in computer science from Central South University (CSU), Changsha, China, in 2005 and 2011, respectively.

She is currently an Associate Professor with the School of Computer Science and Engineering, CSU. She is currently leading some research projects supported by National Natural Science Foundation of China. Her research interests include federated learning, distributed machine learning, computer vision, and edge computing.

Hui Xiao received the B.E. degree from Shandong University, Jinan, China, in 2017, and the M.E. degree from Central South University, Changsha, China, in 2020, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

Her main research interests include mobile edge computing and cloud computing.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is currently a SUNY Distinguished Professor of Computer Science with the State University of New York, New Paltz, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or co-authored over 900 journal articles, book chapters, and refereed conference papers. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top five most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing.

Dr. Li has received several best paper awards and chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an AAAS Fellow and an AAIA Fellow and also a member of *Academia Europaea* (Academician of the Academy of Europe).