Edge Data Caching with Consumer-Centric Service Prediction in Resilient Industry 5.0

Guoqiang Liu, Guanming Bao, Muhammad Bilal, *Senior Member, IEEE*, Angel Jones, Zhipeng Jing, Xiaolong Xu* , *Senior Member, IEEE*

***Abstract*—Industry 5.0, emerging as a promising industry paradigm, unleashes the potential of improving consumer experience by delivering consumer-centric services, facilitating substantial growth in consumer electronics. To improve the resilience of industry 5.0, edge data caching enables sustainable and low-latency service provision by caching data at edge servers (ESs) closer to production. However, the limited caching capacity of ESs presents a formidable challenge to efficient edge data caching. Moreover, the dynamic of consumer-centric service requests further complicates the effective implementation of caching strategies. In response to the above challenges, we propose an edge data caching scheme, named SPM-ECDP, with consumer-centric service prediction for Industry 5.0. Initially, a time-series prediction model is employed to forecast the service demands. To ensure the confidentiality of data, federated learning is introduced in the model training phase. Subsequently, reinforcement learning is adopted to enable ESs to make intelligent decisions on edge data caching, consequently enhancing caching efficiency. Through comprehensive simulation experiments, the effectiveness and superiority of the proposed scheme in increasing caching hit ratio and reducing data delivery delays are demonstrated. The experimental results demonstrate that the proposed SPM-ECDP method has enhanced the hit ratio by 7.05% - 48.5% when compared to the baseline method.**

***Index Terms*—Consumer-centric, industry 5.0, edge caching, edge computing, privacy preservation, reinforcement learning**

I. Introduction

With the rapid advancement of cutting-edge technologies, including edge computing, the Internet of Things (IoT), and 5G cellular networks, Industry 5.0 has emerged as the next industrial revolution. Industry 5.0 builds upon the strengths and capabilities of Industry 4.0 but places paramount importance on three key aspects: human-centricity, sustainability, and resilience [1], [2].

As a novel human-centric industrial paradigm, industry 5.0 presents consumers with a diverse array of customer-centric services, including personalized product customization, product technology modernization, and product quality enhancement [3]. These services contribute to the standardization of electronic products, reducing electronic waste and mitigating the global environmental crisis [4]. Unlike previous industrial models, Industry 5.0 places significant emphasis on the collaboration between humans and machines, enabling automation in manufacturing processes and empowering workers to concentrate on delivering customized services and products [5]. The collection, transmission, and processing of data are essential to automated manufacturing to expedite service delivery. To offer consumers faster and more sustainable services, consideration must be given to the delay and privacy issues involved in data transmission and processing [6].

By deploying computing resources closer to the devices, edge computing effectively reduces data transmission delay and network bandwidth consumption, leading to improved service response speed and enhanced data security [7], [8]. In Industry 5.0, consumer-centric services have gained popularity due to their ability to provide consumers with high-quality products and enhance the overall consumption experience. By harnessing the potential of caching and data processing at edge nodes, edge caching effectively provides end-users with reduced latency and more resilient services, significantly improving their overall service experience [9]. Edge caching is equally applicable in industrial scenarios, as it introduces innovative data management practices to industrial processes, significantly enhancing the flexibility of production processes. However, the implementation of edge caching in Industry 5.0 still presents several challenges. Firstly, there is a conflict between the factory's flexible data requests and the limited storage capacity of the edge nodes [10]. Additionally, although artificial intelligence models play a huge role in caching decisions, factories may choose not to use sensitive data for training the models, as there is an increasing emphasis on privacy protection and safeguarding their interests [11], [12]. Thirdly, the time dynamics of factory requests for service necessitates that edge nodes time-varying adapt their caching strategies to enhance caching efficiency [13]. Therefore, it is necessary to incorporate data popularity prediction and enable intelligent caching decision-making to enhance the efficiency of edge caching in Industry 5.0 [14].

To address the challenges of applying edge caching in Industry 5.0, there is an urgent need for model training methods that can safeguard data privacy. Federated learning, a distributed training method, play a significant role in the process of training artificial intelligence models. It ensures

*Corresponding author

Guoqiang Liu is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: bwq0709@gmail.com.

Guanming Bao is with the School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, China, 210044. E-mail: guanmingbao02@gmail.com.

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, Bailrigg, Lancaster LA1 4WA, United Kingdom. E-mail: m.bilal@ieee.org.

Angel Jones is with School of Continuing and Professional Studies, University of Virginia, Virginia, USA. E-mail: Angel.Jones@virginia.edu.

Zhipeng Jing is with Reading Academy, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: 304674313@qq.com.

Xiaolong Xu is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: xlxu@ieee.org.

privacy preservation during training by enabling each training node to locally train the model on its own data, without the need for uploading private data [15], [16]. Upon completing local model training, each training node simply needs to upload the updated model parameters to the central server to update the global model, thereby fundamentally eliminating the risk of privacy disclosure during data upload [17], [18]. Federated learning is widely utilized in edge caching owing to its privacy-preserving features during training [19], [20]. On the other hand, to fully utilize the limited caching space of edge nodes, the ability of reinforcement learning algorithms to enable agent to make independent decisions plays an important role intelligent caching decision-making. These algorithms empower edge nodes to dynamically adjust their caching decisions in response to changes in the cache environment through continuous interactions. This adaptive approach enables the edge nodes to make more efficient and effective caching decisions, optimizing the utilization of available storage space [21]. By integrating these two methods, dynamic optimization of the edge caching strategy can be achieved, ensuring the privacy of training data, and maximizing the caching hit ratio [22].

In this paper, we propose a novel edge caching scheme, named SPM-ECDP, to provide secure and high-performance caching for consumer electronics suppliers. Specifically, service demand is predicted to determine the services that will be requested in the future. To ensure privacy protection during the training of predictive models, a federated learning framework is introduced. Additionally, the paper presents an intelligent caching method, which aims to determine the optimal caching strategy, leading to significantly improved caching efficiency. The main contributions of this paper are as follows:

- Design an intelligent edge caching scheme for Industry 5.0, enhancing the elasticity of the production process while prioritizing privacy preservation for the industrial Internet.
- Utilize Long Short-Term Memory (LSTM) for predicting future service demand and integrating federated learning to preserve privacy during model training, named SPM.
- A Deep Deterministic Policy Gradient (DDPG)-based approach is meticulously crafted to enable intelligent caching decisions, thereby yielding significantly improved caching hit ratios and substantially reduced data transmission delay, named ECDP.
- Conduct comprehensive experiments to validate the performance of the proposed caching scheme SPM-ECDP in improving caching hit ratio and reducing data delivery latency.

The rest of the paper is organized as follows: Section II describes the related work. In section III, we introduce the models and define the problem. In section IV, the detailed description of caching scheme is provided. In Section V, through simulation experiment, we analyze the effectiveness of the proposed edge caching scheme. The conclusion of this research is illustrated in Section VI.

II. RELATED WORK

In this section, detailed introductions are provided for three aspects of related work. The following is a comprehensive presentation of these aspects.

A. Federated Learning in Edge Caching

Many articles have researched federated learning in edge caching. Addressing the challenges posed by the constant movement of vehicles and the rapid obsolescence of content on the edge network, Yu et al. [23] introduced a mobile-aware proactive edge caching scheme based on federated learning. This innovative approach not only safeguards user privacy but also enables dynamic content caching in a seamless manner. Li et al. [24] presented a collaborative edge caching scheme based on federated deep reinforcement learning, which effectively achieved dynamic adaptation of edge caching while ensuring privacy protection. Furthermore, this scheme addressed the caching challenges associated with high-precision maps for smart connected cars. To strike a balance between caching cost and transmission delay, Li et al. [25] introduced an innovative edge caching scheme based on community detection and attention-weighted federated learning. Within this framework, attention-based weighted federated learning is employed to predict data popularity, consequently achieving the dual benefits of user privacy protection and enhanced prediction accuracy.

B. Reinforcement Learning in Edge Caching

With the superiority of reinforcement learning in intelligent decision-making, many studies have begun to integrate reinforcement learning into edge caching. Aiming at the problems of edge caching device deception and cached content being attacked, Xu et al. [26] proposed a secure edge cache solution that encourages edge caching devices to participate in caching work and uses reinforcement learning methods to derive the content provider's optimal payment strategy. Wang et al. [27] introduced a device-to-device assisted heterogeneous collaborative edge caching framework. This innovative approach addresses the joint optimization problem utilizing a deep Q-learning network and enhances the training efficiency of the Q-learning network by incorporating an attention-weighted federated deep learning model. The experimental results demonstrate that the proposed framework effectively reduces the average latency of content access and improves the caching hit ratio. To minimize content access delay and traffic costs, Wang et al. [28] devised an intelligent edge caching framework. This framework incorporates multi-agent-based reinforcement learning to empower edge nodes with the capacity to make adaptive decisions and achieve intelligent collaborative caching among them.

C. Application of Edge computing in Industrial Internet

Edge caching offers data delivery services with reduced latency for the Industrial Internet, thus prompting numerous studies to explore its application in the Industrial Internet domain. While edge caching offers low-latency data services for

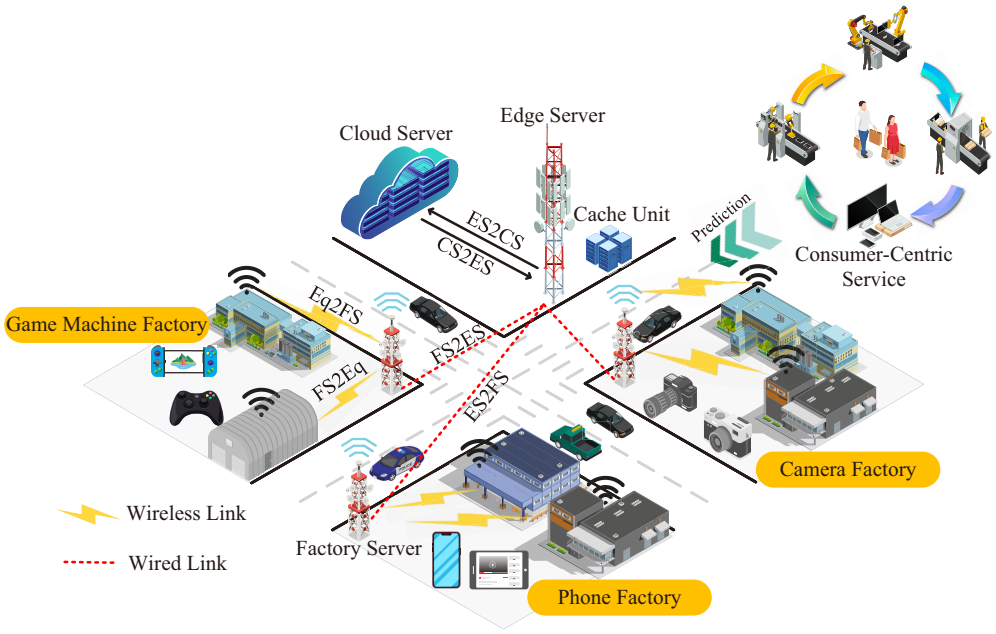


Fig. 1: Edge data caching for consumer-centric service in industry 5.0

the industrial Internet, conventional edge caching approaches often overlook the variability of user interests, leading to a potential decline in user experience quality. To address this issue, Li et al. [29] proposed a recommendation-assisted edge caching method aimed at enhancing user experience while optimizing the caching hit ratio. Li et al. [30] introduced a proactive caching strategy that considers both location and movement trajectory, accounting for the spatial mobility of industrial equipment. The effectiveness of this scheme has been demonstrated through its notable advantages in terms of throughput and real-time performance. Xu et al. [31] leveraged digital twin to construct a virtual world that accurately reflects the real world, while also implementing effective caching decisions.

Extensive researches have been conducted on edge caching, as mentioned above. Some of these studies address privacy preservation for sensitive data, while others concentrate on optimizing the utilization of caching storage on ESs. However, only a limited number of investigations have taken into consideration privacy preservation and intelligent caching decisions simultaneously. To mitigate the aforementioned issue, this paper proposes a novel method, situated within the context of the Elastic Industry 5.0, which aims to achieve more efficient edge caching while upholding robust privacy protection through a synergistic integration of federated learning and reinforcement learning.

III. SYSTEM MODEL

There are three parts in this section. Firstly, the introduction of edge caching framework for resilient industry 5.0. Then, the caching model and the service popularity model are described. Finally, the problem involved in edge caching is formulated. Table I lists the definition of notations.

A. System Framework

From Fig. 1, it is evident that each factory occupies a specific geographical area as the production site. Within each factory, there are factory servers (FSs) responsible for receiving service requests from the equipment and providing the necessary data support to these equipment based on the requests. These FSs can be represented as $F = \{f_1, \cdots, f_i, \cdots, f_N\}$, where f_i represents the i_{th} FS, and N represents the total number of FSs. The equipment within the factory is connected to the FS through wireless links, and these equipment can be represented as $E = \{e^1, \cdots, e^i, \cdots, e^j\}$. Where $e^i = \{e^i_1, ..., e^i_j, ..., e^i_{M_i}\}$ denotes the set of equipment connected to f_i, and M_i represents the number of equipment connected to f_i . In this paper, FS does not possess storage capabilities and can only receive and forward service requests. This consideration is aligned with the efficient practices of modern industry, as equipping each factory with separate servers containing independent computing and storage resources would be wasteful and counterproductive to the cost optimization in industrial settings. Therefore, in the remote areas outside the factories, there are ESs equipped with computing and storage resources. These ESs are connected to the FSs through wired links. Their primary function is to receive service requests forwarded by FS and send data to the relevant FS, thereby providing services to multiple FS within the same region simultaneously. The set of service is represented as $S = \{s_1, ..., s_k, ..., s_K\}$, where k denotes the length of the set S. The execution of services requires essential data support, and the data required for a service may involve various types and multiple data items. For the sake of simplicity, we use d_k to represent all the data required for s_k, and z_k represents the size of the d_k. All the service requests received by f_i can be represented as $R_i = \{r_{i,1}, ..., r_{i,k}, ..., r_{i,K}\}$, where $r_{i,k}$ denotes the number of requests from factory equipment within the coverage area of

TABLE I: NOTATIONS DEFINITION

Notation	Definition
F	Set of factory servers(FSs).
f_i	The i_{th} factory server $\in F$
E	Set of equipment in factories.
e_j^i	The j_{th} equipment located within the coverage area of f_i.
S	Set of services.
s_k	The k_{th} service $\in S$
d_k	The data required for service s_k.
z_k	The size of d_k.
R_i	The service requests received by f_i.
$r_{i,k}$	The number of requests for s_k received by f_i.
r_i^ϕ	The service requests received by f_i during ϕ.
U	The set of data cached by ES.
Q	The upper limit of ES's caching capacity.
q_k	A binary variable used to distinguish the caching location of d_k.
t_k	The time required for the transmission of d_k.
T	The time required for the all data transmission.
μ_i^ϕ	The caching hit ratio during time period ϕ.
μ	The caching hit ratio of the entire caching system.
τ	The loss between $r_i^{\phi+1'}$ and $r_i^{\phi+1}$.
∇	The MSE loss function.
Γ	The function of model training.
Θ	The model parameter acquisition function.
θ_i	The parameters of $\varkappa_i$.

f_i. To distinguish requests from different time periods, it can be further represented as $r_{i,k} = \{r_{i,k}^1, ..., r_{i,k}^\phi, ..., r_{i,k}^\Phi\}$, where $r_{i,k}^\phi$ represents the number of requests for s_k received at time ϕ. However, due to the relatively limited storage capacity of the ESs compared to cloud server, caching all the data required for all services is not feasible. Therefore, ES needs to be connected to the cloud server to promptly request data from them when it cannot meet all the data requirements of the incoming service requests. This ensures the smooth operation of the factories.

B. Edge Caching Model

In the proposed framework of this paper, only two devices are capable of providing data storage since the FS lacks storage capabilities. These devices are the ES and the cloud server. Assuming the cloud server possesses wireless storage resources and can cache all the data, while the capacity of ES is limited. Therefore, the data cached by ES can be represented as $U = \{u_1, ..., u_l, ..., u_L\}$, where L represents the maximum number of data items that ES can cache. We distinguish the location where data is cached using a binary variable . The value of q_k can be determined based on the following equation:

$$q_k = \begin{cases} 1, & d_k \in U \\ 0, & d_k \notin U \end{cases}, \tag{1}$$

where $d_k \in U$ represents d_k cached by the ES, denoting $q_k = 1$; $d_k \notin U$ represents d_k that is not cached by ES but exists in the storage space of the cloud server, indicating $q_k = 0$. The upper limit of the caching capacity of the ES is represented by Q . Consequently, when performing edge caching, ES must satisfy the following equation:

$$\sum_{k=1}^{K} z_k \times q_k \leq Q. \tag{2}$$

This equation represents that the sum of the storage of data in the ES must not exceed the storage upper limit Q of ES.

C. Data Transfer Model

Service requests originating from the equipment within the factory are sent to the FS, which, in turn, forwards relevant data requests to the ES. If ES caches the requested data, it directly sends the data to FS through a wired link. The reason for not sending the data directly to the devices is due to the concern for factory privacy protection. The devices within the factory are only connected to their own on-site FS. Once FS receives the data, it transmits the data to the devices via a wireless link. The bandwidth of the wired link between the FS and the ES is denoted as w_a, and the bandwidth of the wireless link between FS and the equipment is denoted as w_b. Due to the differences in hardware facilities and transmission methods, the bandwidth between FS and ES is greater than the bandwidth between FS and the equipment, e.g., $w_a > w_b$. If the ES does not cache the requested data, it proceeds to request the data from the cloud server. Once the data is obtained from cloud server, ES forwards it to the FS. The connection between ES and the cloud server is established through a wired link, and its bandwidth is represented as w_c. The relationship between the three different bandwidths can be expressed as $w_c \geq w_a > w_b$. Once the transmission bandwidths between different devices are obtained , It is easily to determine the time required for the single transmission of d_k as

$$t_k = \frac{z_k}{w_b} + \frac{z_k}{w_a} + \frac{z_k}{w_c} \times (1 - q_k), \tag{3}$$

$$s.t.\ \ w_c \geq w_a > w_b. \tag{4}$$

Therefore, the total time taken to transmit all the data over all time periods can be calculate as

$$T = \sum_{\phi=1}^{\Phi} \sum_{i=1}^{N} \sum_{k=1}^{K} t_k \times r_{i,k}^\phi. \tag{5}$$

Another major metric for evaluating caching algorithms is the caching hit ratio, which will have a significant impact on the effectiveness of the caching system. In the proposed caching architecture in this paper, not only the types of cached data are considered but also the frequency of requests for different data items is taken into account. The caching hit ratio of a single FS in ϕ time period can be calculated as

$$\mu_i^\phi = \frac{\sum_{k=1}^{K} q_k \times r_{i,k}^\phi}{\sum_{k=1}^{K} r_{i,k}^\phi} = \frac{\sum_{k=1}^{K} r_{i,k}^\phi}{\sum_{k=1}^{K} r_{i,k}^\phi} \times q_k, \tag{6}$$

where

$$i \in [1, N], \phi \in [1, \Phi]. \tag{7}$$

Therefore, the caching hit ratio of the entire caching system can be determined as

$$\mu = \sum_{i=1}^{N} \sum_{\phi=1}^{\Phi} \mu_i^{\phi}. \tag{8}$$

D. Problem Definition

In order to achieve optimal performance in the edge caching strategy for resilient industry 5.0 and to promote further growth in consumer electronics, both the time consumption and caching hit ratio of the edge caching solution need to be taken into consideration. Therefore, the problem can be defined as

$$max(\mu), min(T), \tag{9}$$

$$s.t. \sum_{k=1}^{K} z_k \times q_k \leq Q. \tag{10}$$

A higher caching hit ratio and lower data transmission time consumption can efficiently facilitate the delivery of required data to the equipment within the factory when they request services. This, in turn, enhances the performance of the edge caching solution in serving industrial production and improves the efficiency of the production process in the consumer electronics industry, ultimately leading to an enhanced consumer experience.

IV. DESIGN OF SPM-ECDP

In the following parts, detailed information on the various components of the caching scheme SPM-ECDP is provided.

A. Service Requests Prediction

During a certain time span, the FS receives numerous diverse service requests. Although FS does not possess the ability to cache data required by services, it still store these service requests record in its limited storage space. As mentioned in Section III, the received service requests are represented by $R_i = \{r_{i,1}, ..., r_{i,k}, ..., r_{i,K}\}$, where $r_{i,k} = \{r_{i,k}^1, ..., r_{i,k}^{\phi}, ..., r_{i,k}^{\Phi}\}$ denotes the number of service requests at different time intervals. With the service requests record for different time intervals, the use of time-series forecasting models allows for predictions to be made, thereby obtaining the number of service requests expected in future time intervals. Firstly, the requests are divided based on time intervals to obtain the training dataset, denoted as $R_i^T = [r_i^1, \cdots, r_i^{\phi}, \cdots, r_i^{\Phi}]$, where $r_i^{\phi} = [r_{i,1}^{\phi}, ..., r_{i,k}^{\phi}, ..., r_{i,K}^{\phi}]$. When using LSTM for time-series prediction of request counts, is used as the input to the model, and the output represents the predicted results. The choice of LSTM as the temporal prediction model, as opposed to GRU or RNN, stems from its capability to effectively mitigate issues such as gradient explosion and vanishing gradient, while also demonstrating proficiency in modeling long-term sequences and capturing temporal characteristics. The prediction process can be represented using the following equation as

$$r_i^{\phi+1'} = \sigma\left(\left[r_i^{\phi-\zeta}, \cdots, r_i^{\phi}\right]\right), \tag{11}$$

where $r_i^{\phi+1'}$ represents the predicted value of the number of service requests received in the $\phi+1$ period and ζ is the length of training step. In order to measure the error between the predicted values and the actual values as the basis for model training, the following equation is used to calculate the error

$$\tau = \nabla(r_i^{\phi+1'}, r_i^{\phi+1}), \tag{12}$$

where ∇ represents the Mean Squared Error (MSE) loss function and τ is the error. The model training is performed using the gradient descent method, continuing until the τ of converges, with the aim of obtaining a highly accurate predictive model. The service request prediction is illustrates in Fig. 2.

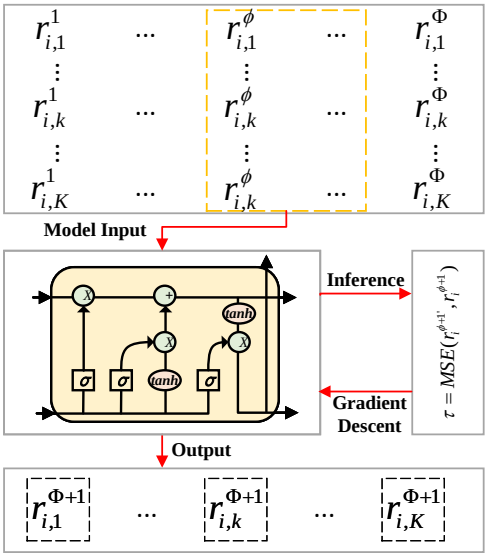


Fig. 2: Consumer-centric service requests prediction

B. Distributed Training with Privacy Preservation

As the ES provides data caching services for multiple FS within the entire region, and service requests from factories within the same region exhibit certain similarities, it is not feasible to rely solely on the model output results from individual FS for caching decisions. Instead, it is necessary to consider the characteristics of service requests from the entire region. Therefore, a collaborative training approach involving multiple FSs within the same region is needed, allowing all request records to contribute to the model, thereby training a model with strong feature inclusiveness and high usability. As mentioned in the previous part, the time-series forecasting model on f_i using service request records exclusive to f_i. However, these records may involve sensitive information, and in order to safeguard their own interests, factories may choose not to upload service request records to a central node for model training. In order to achieve effective protection of sensitive data while obtaining a global model, we introduce federated learning into the model training process. We use $\varkappa_i$ to represent the model trained with R_i^T as input, and the θ_i

represents the parameters of $\varkappa_i$. The process of obtaining $\varkappa_i$ and θ_i can be represented as

$$\varkappa_i = \Gamma(R_i^T), \tag{13}$$

$$\theta_i = \Theta(\varkappa_i), \tag{14}$$

where Γ represents the model training function and Θ represents the model parameter acquisition function.

After obtaining the parameters of each predictive model, FSs only need to upload the model parameters to the ES for the aggregation of the global model $\varkappa_g$. This process can be represented as

$$\theta_g = \sum_{i=1}^{N} \frac{\left|R_i^T\right|}{\sum_{i=1}^{N}\left|R_i^T\right|}\theta_i, \tag{15}$$

where θ_g is the parameters of global model $\varkappa_g$. Once the θ_g is obtained, it will be sent from the central aggregation node ES to the training nodes FSs for further training until the global model $\varkappa_g$ converges. Subsequently, each FS utilizes θ_g to build model and employs this model for inference, thereby obtaining the number of service requests for the next time interval. The model inference process is represented as

$$\varkappa_g = \Xi(\theta_g), \tag{16}$$

$$r_i^{\Phi+1'} = \sigma\left(\left[r_i^{\Phi-\zeta}, r_i^{\Phi-1}, r_i^{\Phi}\right]\right), \tag{17}$$

where Ξ is the model construction function which takes θ as input and $r_i^{\Phi+1'}$ is the predicted service requests in $\Phi+1$ period.

The sum of predicted request number for services can be calculated as

$$r^{\Phi+1'} = \sum_{i=1}^{N} r_i^{\Phi+1'}. \tag{18}$$

The detailed presentation of the prediction process for service request number is shown in Algorithm 1.

Algorithm 1: Consumer-Centric Service Prediction with Privacy Preservation

Input: the service request records R_i^T in $f_i \in F$
Output: the sum of predicted request number for services $r^{\Phi+1'}$

Initialize parameters θ_g;
while *the global model* $\varkappa_g$ *converged* **do**
  for $f_i \in F$ **do**
    Constructing $\varkappa_i$ using θ_g;
    for *each epoch* **do**
      Getting prediction result $r_i^{\phi+1'}$ using Eq. (11);
      Getting the loss between predicted value and actual value τ using Eq. (12);
      Adjusting model parameters using the gradient descent method.
    end
    Getting model parameters θ_i;
  end
  The set $\theta = (\theta_1, \cdots, \theta_i, \cdots, \theta_N)$ is obtained;
  Using Eq. 15 to get θ_g;
end
for $f_i \in F$ **do**
  Using Eq. 16 to construct $\varkappa_g$ based on θ_g;
  Using Eq. 17 to get $r_i^{\Phi+1'}$;
end
The sum of predicted request number for services is obtained by Eq. 18.
Return $r^{\Phi+1'}$;

C. Intelligent Data Caching Decisions

As an entity responsible for executing data edge caching, the ES has a limited storage capacity, denoted by Q. Therefore, ES cannot cache all data, and a caching decision scheme needs to be formulated for edge caching decisions. The objective of this decision-making process is to leverage the limited caching space to provide optimal caching. In the proposed framework in this paper, different services have varying request frequencies, and thus, data size alone cannot serve as the sole criterion for caching decisions. This can be illustrated with the following example.

1) case 1: The number of data requests is 50 and the size of data is 3MB.
2) case 2: The number of data requests is 30 and the size of data is 2MB.
3) case 3: The number of data requests is 40 and the size of data is 2MB.

The storage capacity of the ES is assumed to be 4 MB. Based on the principle of request frequency priority, the correct caching decision may seem to be caching case 1. However, it would be wiser to cache case 2 and case 3 because this approach takes both the request frequency and data size into consideration in the decision-making process. By doing so, it fully utilizes the limited caching space available.

To effectively implement intelligent caching decisions, reinforcement learning has been proven to be a viable approach. However, due to the limited computational resources in ES, employing complex algorithms like Twin Delayed Deep Deterministic Policy Gradient (TD3) may partially impact the efficiency of intelligent caching decisions, mainly due to TD3's adoption of a twin critic network. On the other hand, simpler algorithms like Actor-Critic (AC) may not achieve optimal decision-making [32]. Additionally, reinforcement learning algorithms such as Deep Q-Network (DQN) may have limitations in handling continuous actions [33]. Therefore, for edge caching decisions, we have opted for an improved version of AC known as DDPG, which builds upon AC by integrating deterministic policies and experience replay techniques. This allows DDPG to strike a balance between avoiding excessive resource consumption and performing well in handling problems in continuous action spaces. The agent interacts with the caching environment to collect experience samples, and randomly samples experience tuples from experience replay buffer for model training. Additionally, the updating of target network parameters is performed using a soft update strategy,

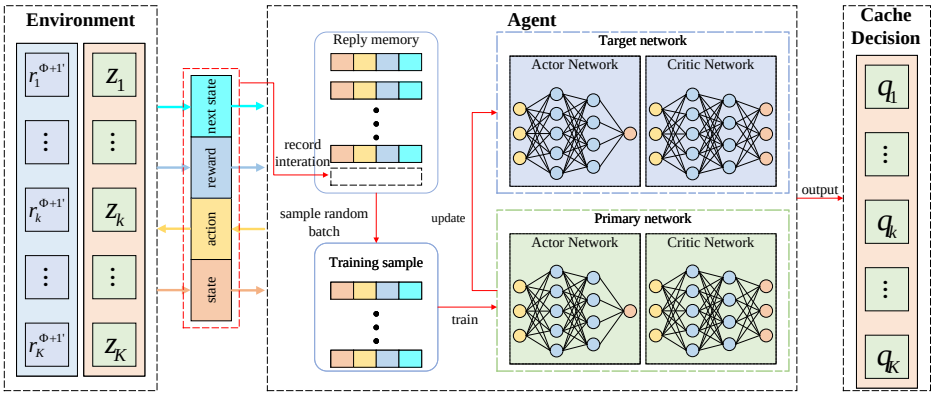


Fig. 3: Intelligent edge caching decision with DDPG

gradually approaching the current network parameters. The specific caching decision process is illustrated in the Fig. 3.

Algorithm 2: Intelligent Data Caching Decisions with DDPG (ECDP)

Input: The predicted number of service requests $[r_1^{\Phi+1'}, \cdots, r_k^{\Phi+1'}, \cdots, r_K^{\Phi+1'}]$ and the size of data $[z_1, \cdots, z_k, \cdots, z_K]$

Output: The caching decision $[q_1, \cdots, q_k, \cdots, q_K]$

Construct state space S, action space A;
Initialize parameters of actor network $\mathcal{A}$ and critic network $\mathcal{C}$: θ_a, θ_c;
Initialize parameters of target actor network $\mathcal{A}'$ and critic network $\mathcal{C}'$: $\theta_a^{'}, \theta_c^{'}$;
for *each episode* **do**
  for *each step* **do**
    Choose an action a;
    Compute the reward rd using Eq.16 and a new state $s^{'}$;
    Store $\{s, a, r, s^{'}\}$ in buffer;
    Randomly select $\{s, a, r, s^{'}\}$ from buffer;
    Compute y using target critic network the following formula that $y = r + \gamma \cdot \mathcal{C}^{'}(s^{'}, \mathcal{A}^{'}(s^{'}|\theta_a^{'})|\theta_c^{'})$;
    Compute the loss by the following foumula that $L = \frac{1}{N}\sum_{i=1}^{N}(y - \mathcal{C}(s, a|\theta_c))^2$;
    Update the actor policy using the sampled policy gradient;
    Update the parameters of target actor network by the following formula that $\theta_a^{'} \leftarrow \tau \cdot \theta_a + (1-\tau) \cdot \theta_a^{'}$;
    Update the parameters of target critic network by the following formula that $\theta_c^{'} \leftarrow \tau \cdot \theta_c + (1-\tau) \cdot \theta_c^{'}$;
  end
end
Return the final action $a = [q_1, \cdots, q_k, \cdots, q_K]$, where $q_k = 0$ indicates that d_k is not cached, and $q_k = 1$ signifies the opposite.

It can be observed that the construction of the reinforcement learning environment is based on the predicted values of future time-period request counts for each service, and the sizes of the data required by each service serve as the basis for calculating rewards. The calculation of the reward can be expressed as

$$rd = \begin{cases} rd + q_k \times r_k^{\Phi+1}, \sum_{k=1}^{K} z_k \times q_k \leq Q \\ rd - \alpha, \sum_{k=1}^{K} z_k \times q_k > Q \end{cases}, \tag{19}$$

where is a constant used to control rd. The reason for doing this is that the cached data size has exceeded the storage limit of ES when $\sum_{k=1}^{K} z_k \times q_k \leq Q$, and it is necessary to avoid such situations as much as possible. At this point, a penalty is imposed on rd by subtracting λ from it, enabling reinforcement learning to autonomously avoid caching data that exceeds the limit. The training process is repeated until rd converges. At this point, the final caching decision is output, e.g., $[q_1, \cdots, q_k, \cdots, q_K]$ in Fig. 3. $q_k = 0$ represents that d_k is cached by ES, while $q_k = 1$ indicates that d_k is not cached by ES. The detailed presentation of the intelligent caching decision-making is shown in Algorithm 2.

The discussion above outlines the proposed caching scheme, which leverages federated learning for privacy preservation while training predictive models, using reinforcement learning for intelligent caching decisions.

V. PERFORMANCE RESULTS AND ANALYSIS

In this section, we verify the effectiveness and superiority of the proposed scheme.

A. Experimental Setup

The hardware environment based on this experiment is as follows: the CPU is Intel i7-8700, and the GPU is NVIDIA GeForce GTX 1080. All experiments are performed in Python 3.10 and Pytorch 1.13.0.

The dataset used in this study is a simulated dataset based on a real dataset which is collected from Nanjing.

The data set contains more than 100 million pieces of information about the service requirements of vehicle users. The information was collected from 436 RSUs covering Nanjing City from September 1, 2014, to September 30, 2014. The format of the raw dataset comprises a quadruple $< id, speed, request, region >$, where id represents the vehicle's identifier, $speed$ denotes the vehicle's velocity, $request$ signifies the data request initiated by the vehicle, and $region$ indicates the geographical location of the vehicle. To make the dataset more suitable for the scenario proposed in this paper, we have processed it. Specifically, we select five RSUs in the same region as FSs. Service requests from vehicles within the coverage of each RSU are treated as service requests from equipment inside a factory. The processed dataset includes the following fields which are shown in table II.

TABLE II: The description of the field in the Dataset

Fields	Description
FS_{id}	Unique identifier for factory server.
FS_{loc}	Geographical location of factory server.
SR_{id}	Unique identifier for service request.
SR_{loc}	Geographical location of service request originated.
$Data_{id}$	Unique identifier for data.
$Data_{size}$	Size of the each data.

The experiment in this paper involves two models, namely the time series prediction model LSTM and the reinforcement learning algorithm DDPG. The experimental parameter settings for LSTM are as follows that the input size is 100, the layers number is 10, the hidden size is 32, the output size is 100 and the learning rate is 0.001. The experimental parameter settings for DDPG are as follows that the max episodes is 1000, memory capacity is 1000, the discount factor is 0.9.

B. Performance Evaluation

In this part, simulation experiments are conducted to demonstrate the effectiveness of the proposed scheme, and through comparative experiments, the superiority of the approach presented in this paper is validated. The multiple caching schemes compared with the proposed scheme are introduced in turn below.

- TD3-based caching scheme (TDCS). The caching scheme proposed in this paper which uses Twin Delayed Deep Deterministic policy gradient algorithm (TD3) to make the caching decision [34].
- AC-based caching scheme (ACS). The Actor-Critic (AC) algorithm consists of two networks, namely the Actor network and the Critic network. This reinforcement learning algorithm is utilized for making decisions regarding the content to be cached in this paper [35].
- Random caching scheme (RCS). The edge caching scheme randomly selects a portion of data from the set of all data for caching without performing any prediction work beforehand [36].
- Cloud caching scheme (CCS). The cloud caching which means that there is no data caching in the FS, only the cloud server performs data caching [37].

The evaluation metrics involved in the experiment are described as follows:

- Reward: The feedback signals obtained by the agent in the environment after taking actions are utilized to guide the agent's learning on how to select actions in different states with the objective of maximizing cumulative rewards.
- Hit Ratio: The ratio of the number of data requests fulfilled by edge servers to the total number of data requests.
- Time Cost: The total time spent in the system to satisfy all data requests.

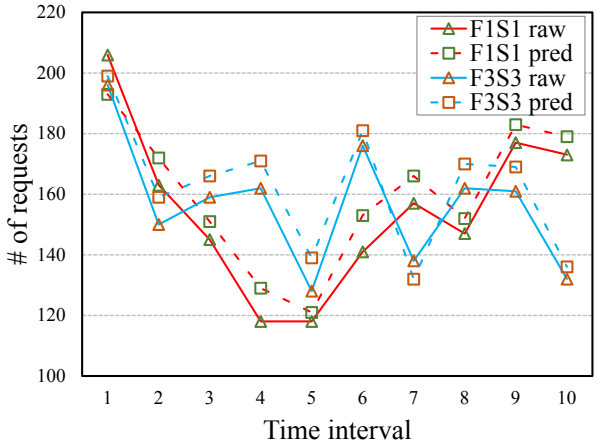


Fig. 4: The performance of prediction models based on federated learning

Fig. 4 depicts the performance of the SPM proposed in this paper. We randomly select $F1S1$ and $F3S3$, where $F1S1$ represents the number of requests for s_1 on f_1, and $F3S3$ represents a similar quantity for s_3 on f_3. The magnitude of a time interval on the x-axix is 0.5 hour. The solid and dashed lines represent the actual values and the predicted values, respectively. The remarkable consistency between the actual and predicted values across diverse time periods unequivocally demonstrates the accuracy of the time series predicting model. Regrettably, there is a degree of deviation between the predicted and actual values in certain time periods, which is attributed to the incorporation of the federated learning framework during the model training process, resulting in a partial loss of model accuracy.

Fig. 5 shows the performance of ECDP at different learning rates, in which the lr_a is the learning rate of actor network and the lr_c is the learning rate of critic network. It can be seen from the figure that when lr_a=1e-3, lr_c=1e-2, the reward value rises rapidly, and the highest reward is reached after about 80 rounds. But then decreases to a certain extent, indicating that the learning rate is too large. Reduce the learning rates to lr_a=1e-4, lr_c=1e-3, respectively. At this time, we see from the figure that the reward value also rises rapidly, and it will reach its peak after about 80 rounds. The difference from the previous curve is that the blue curve does not cause the reward value to drop in the subsequent training, which shows that the

current learning rate is better for network training. To prevent the reward from increasing too rapidly due to the learning ratio, we adjust the learning rate to lr_a=1e-5, lr_c=1e-4, and the reward value changes as shown in the red curve. It can be seen from the figure that the reward value rises in steps at the initial stage and reaches the peak after about 400 rounds, and then only fluctuates in a small range. Compared with the other two curves, although the red curve takes more time to reach the peak reward value, the reward value is much higher. Considering that further reducing the learning rate will lead to too long learning time, we no longer reduce the learning rate, but only take the best combination, that is,lr_a=1e-5, lr_c=1e-4.

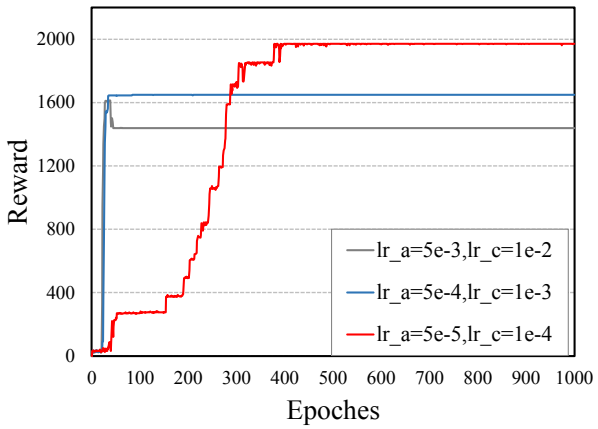


Fig. 5: Rewards under different learning rates

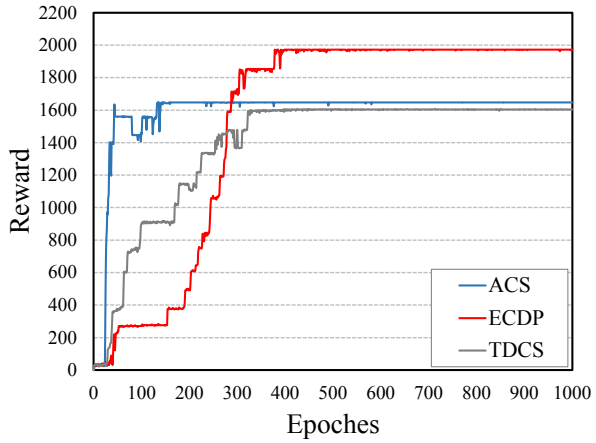


Fig. 6: Comparison on rewards with TDCS, ACS and ECDP

Fig. 6 presents a comparison of the performance gap between the three different reinforcement learning algorithms that are ECDP, ACS, and TDCS. To ensure a fair comparison of the performance of different reinforcement learning algorithms, both AC and TD3 maintain the same learning rate as the optimal learning rate of DDPG. Additionally, TD3 maintains consistent learning rates for its two critic networks. It can be learned that as the numb er of epochs increases, the rewards obtained by the three algorithms exhibit an upward trend. However, their rates of improvement vary significantly.

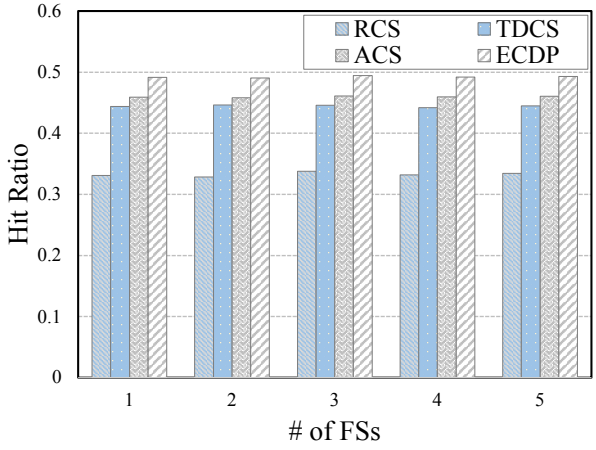


Fig. 7: Hit ratio under RCS, ACS, TDCS and ECDP

The ACS shows the most rapid increase in rewards, reaching a stable state after approximately 200 epochs. Similarly, the reward of TDCS follows a comparable trajectory, with its rewards stabilizing around 300 epochs. It is important to note that while TDCS final stable reward is slightly lower than that of ACS, it still demonstrates commendable performance. On the other hand, the ECDP experiences a slower growth in rewards but manages to achieve the highest stable reward among the three schemes. Through rigorous mathematical computations, it can be deduced that the final stable reward achieved by ECDP surpasses that attainable by ACS by an improvement of 19.7% and exceeds that achievable by TDCS by an improvement of 22.9%. The reason behind this situation lies in the dual advantages of DDPG, namely its effectiveness in handling continuous actions and its relatively simple structure. This outcome highlights the unique strengths and trade-offs of ECDP in the context of the learning task.

The information from Fig. 7 reveals distinct caching hit ratio for each FS under different caching schemes. Notably, the caching scheme RCS attains the lowest caching hit ratio, with the caching scheme TDCS following closely behind. Through computation, it is observed that the caching hit ratio achieved by the proposed scheme ECDP averages 7.05%

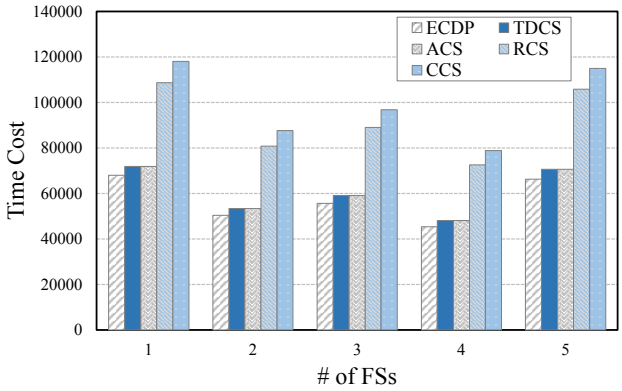
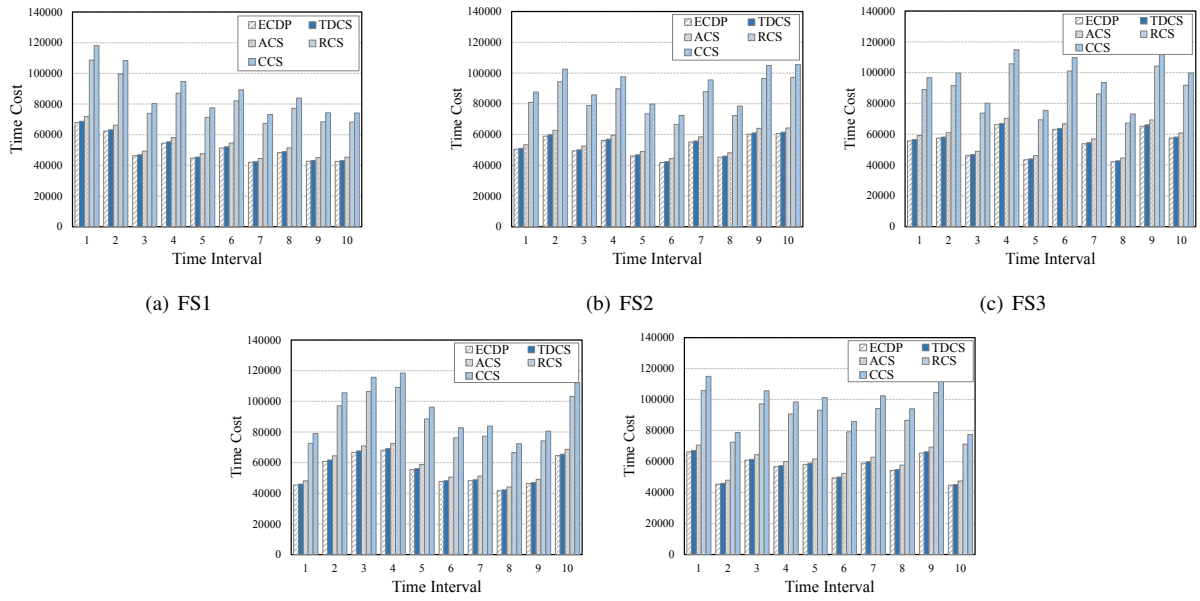


Fig. 8: Time cost under CCS, RCS, ACS, TDCS and ECDP



(a) FS1 (b) FS2 (c) FS3

(d) FS4 (e) FS5

Fig. 9: Time cost of caching scheme in different time periods

higher than ACS, 10.75% higher than TDCS, and 48.05% higher than RCS across five FSs. Conversely, the proposed caching scheme ECDP achieves the highest caching hit ratio. Furthermore, it is important to mention that the caching hit ratio achieved by ECDP consistently remain close across different FSs. This consistent performance underscores the stability and robustness of the proposed caching scheme.

From Fig. 7, it is evident that the ECDP caching scheme achieves the highest edge caching hit ratio, which theoretically implies that it should have the shortest transmission time. This inference is confirmed by Fig. 8. The time consumption of data transmission based on the other two reinforcement learning algorithm scheme, ACS and TDCS, is similar. However, due to its random caching data selection, the RCS requires more time than the reinforcement learning-based caching schemes. The CCS incurs the most significant time cost because it does not involve edge caching; therefore, all the data needs to be transmitted from the cloud server to the terminal device. Through computational analysis, it is determined that the time required for data transmission under the ECDP scheme is reduced by 5.6%, 5.7%, 37.4%, and 42.4% compared to the time spent under the TDCS, ACS, RCS, and CCS schemes, respectively. The fundamental reason behind this situation is that the caching hit ratio achieved by ECDP is higher than that attained by several other benchmark schemes. Consequently, the time required for content transmission is reduced.

To further assess the robustness of ECDP, experiments are conducted on multiple FSs during different time intervals, as depicted in Fig. 9. The time intervals between the periods were set at 30 minutes. The results show that across various time intervals, the ECDP consistently incurre the least amount of time for multiple FSs, while the CCS resulted in the most significant time cost. This finding aligns with the characteristics observed in Fig 8.

VI. Conclusion

In this paper, we present a novel edge caching scheme with consumer-centric service prediction in resilient industry 5.0, leveraging federated learning and reinforcement learning. Specifically, the prediction model is used to predict the possible number of requests for each service in the future period while federated learning is used for privacy protection. Based on the prediction results, intelligent caching decisions are made to pursue higher caching rewards, replacing complex manual decisions, making our method more in line with the flexibility that Industry 5.0 focuses on. Experimental results demonstrate that the proposed scheme outperforms other comparison schemes. In this study, the consideration was given to the privacy concerns associated with model training, while the security of the edge data storage was inadvertently overlooked. In future research endeavors, we intend to further investigate the efficient implementation of data security in intelligent edge caching systems.

ACKNOWLEDGEMENTS

This work was supported in part by National Key Research and Development Program of China under Grant no.2021YFE014400, National Natural Science Foundation of China under Grant no. 92267104 and no. 62372242, and Natural Science Foundation of Jiangsu Province of China under no. BK20211284.

References

[1] J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, T. Wuest, D. Mourtzis, and L. Wang, "Industry 5.0: Prospect and retrospect," *Journal of Manufacturing Systems*, vol. 65, pp. 279–295, 2022.

[2] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021.

[3] S. Nahavandi, "Industry 5.0—a human-centric solution," *Sustainability*, vol. 11, no. 16, p. 4371, 2019.

[4] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[5] X. Xu, J. Gu, H. Yan, W. Liu, L. Qi, and X. Zhou, "Reputation-aware supplier assessment for blockchain-enabled supply chain in industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, pp. 5485–5494, 2022.

[6] X. Xu, S. Tang, L. Qi, X. Zhou, F. Dai, and W. Dou, "Cnn partitioning and offloading for vehicular edge networks in web3," *IEEE Communications Magazine*, vol. 61, no. 8, pp. 36–42, 2023.

[7] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[8] T. Bai, C. Pan, Y. Deng, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Latency minimization for intelligent reflecting surface aided mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2666–2682, 2020.

[9] X. Zhou, M. Bilal, R. Dou, J. J. P. C. Rodrigues, Q. Zhao, J. Dai, and X. Xu, "Edge computation offloading with content caching in 6g-enabled iov," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2023, doi: 10.1109/TITS.2023.3239599.

[10] J. Zhao, X. Sun, Q. Li, and X. Ma, "Edge caching and computation management for real-time internet of vehicles: An online and distributed approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2183–2197, 2020.

[11] S. Liu, C. Zheng, Y. Huang, and T. Q. Quek, "Distributed reinforcement learning for privacy-preserving dynamic edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 749–760, 2022.

[12] Z. Li, G. Li, M. Bilal, D. Liu, T. Huang, and X. Xu, "Blockchain-assisted server placement with elitist preserved genetic algorithm in edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2023, doi: 10.1109/JIOT.2023.3290568.

[13] T. Zong, C. Li, Y. Lei, G. Li, H. Cao, and Y. Liu, "Cocktail edge caching: Ride dynamic trends of content popularity with ensemble learning," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, pp. 208–219, 2023.

[14] X. Xu, C. Yang, M. Bilal, W. Li, and H. Wang, "Computation offloading for energy and delay trade-offs with traffic flow prediction in edge computing-enabled iov," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2022, doi: 10.1109/TITS.2022.3221975.

[15] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 13–23.

[16] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[17] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9, doi: 10.1109/IJCNN48605.2020.9207469.

[18] X. Xu, H. Li, Z. Li, and X. Zhou, "Safe: Synergic data filtering for federated learning in cloud-edge computing," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1655–1665, 2022.

[19] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[20] Z. Li, M. Bilal, X. Xu, J. Jiang, and Y. Cui, "Federated learning-based cross-enterprise recommendation with graph neural networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 673–682, 2022.

[21] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained Markov decision processes," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9797–9806.

[22] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2020.

[23] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2020.

[24] C. Li, Y. Zhang, and Y. Luo, "A federated learning-based edge caching approach for mobile edge computing-enabled intelligent connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 3360–3369, 2023.

[25] D. Li, H. Zhang, T. Li, H. Ding, and D. Yuan, "Community detection and attention-weighted federated learning based proactive edge caching for d2d-assisted wireless networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023, doi: 10.1109/TWC.2023.3249756.

[26] Q. Xu, Z. Su, and R. Lu, "Game theory and reinforcement learning based secure edge caching in mobile social networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3415–3429, 2020.

[27] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 154–169, 2020.

[28] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2499–2508, doi: 10.1109/INFOCOM41043.2020.9155373.

[29] Z. Li, X. Gao, Q. Li, J. Guo, and B. Yang, "Edge caching enhancement for industrial internet: A recommendation-aided approach," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 941–16 952, 2022.

[30] X. Li and J. Wan, "Proactive caching for edge computing-enabled industrial mobile wireless networks," *Future Generation Computer Systems*, vol. 89, pp. 89–97, 2018.

[31] X. Xu, Z. Liu, M. Bilal, S. Vimal, and H. Song, "Computation offloading and service caching for intelligent transportation systems with digital twin," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 757–20 772, 2022.

[32] A. Zanette, M. J. Wainwright, and E. Brunskill, "Provable benefits of actor-critic methods for offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 13 626–13 640, 2021.

[33] J. Shi, L. Zhao, X. Wang, W. Zhao, A. Hawbani, and M. Huang, "A novel deep q-learning-based air-assisted vehicular caching scheme for safe autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4348–4358, 2020.

[34] L. Yao, X. Xu, M. Bilal, and H. Wang, "Dynamic edge computation offloading for internet of vehicles with deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2022, doi: 10.1109/TITS.2022.3178759.

[35] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[36] Y. Chen, M. Wen, E. Basar, Y.-C. Wu, L. Wang, and W. Liu, "Exploiting reconfigurable intelligent surfaces in edge caching: Joint hybrid beamforming and content placement optimization," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7799–7812, 2021.

[37] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital twin empowered content caching in social-aware vehicular edge networks," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 239–251, 2021.