Hindawi
Mobile Information Systems
Volume 2021, Article ID 9981767, 11 pages
https://doi.org/10.1155/2021/9981767

Research Article

Data Collection and Analysis of Track and Field Athletes' Behavior Based on Edge Computing and Reinforcement Learning

Di Han

Jilin Agricultural University, Changchun 130118, China

Correspondence should be addressed to Di Han; handi@jlau.edu.cn

Received 18 March 2021; Revised 11 April 2021; Accepted 20 April 2021; Published 30 April 2021

Academic Editor: Jianhui Lv

With the development of multimedia technology, the computer auxiliary system has become an effective means of daily training in track and field. This paper designs a data acquisition and analysis system for track and field athletes. The system uses sensor modules attached to the athlete's body to collect movement data for analysis. The whole system is implemented by edge computing architecture. In order to reduce average response time, the DDPG algorithm is used to optimize the resource allocation of the edge layer. Experimental results show that the response time of the proposed algorithm can be controlled within 1 s. Meanwhile, the SVM algorithm on the edge server is arranged to classify the data, and the overall recognition accuracy is over 90%.

1. Introduction

Track and field sports is a large category, which includes a variety of subevents, such as race walking, running, and javelin throw. Athletes need to spend a bunch of time and energy developing skills and promoting endurance. A British research institute conducted a survey on the training time of more than 100 elite athletes. Result suggests that, in order to prepare for the 2012 Olympics in London, they trained six hours a day, six days a week, 12 months a year on average, and some athletes even incredibly spent 10000 hours during 4 years, consuming 1.1 million calories a year on average. For track and field athletes, scientific training methods are crucial to the improvement of their competitive level. The personal experience and teaching skills of the coach play a vital role in the training process. However, with the development of information technology, athletes can be effectively trained via the utilization of big data and software analysis [1].

There are studies on track and field sports auxiliary training system. Cucco [2] developed a system for evaluating and improving the performance characteristics of an athlete, such as speed, agility, and quickness. They took advantage of smart sensors to detect the athlete's relative position with training target and display the results to them. Ma et al. [3] developed a C/S mode monitoring system on the athlete training process, which was based on mobile artificial intelligence technology. This system used GPS to obtain real-time position information of athletes and provided real-time guidance. Guo [4] designed a VR system to train athletes to master self-balance by detecting foot pressure data. The authors designed the whole system in the form of VR games to stimulate the enthusiasm of sports training.

Inspired by the above related works, a data acquisition and analysis system based on edge computation and reinforcement learning for track and field training is devised in this paper. The wearable sensing module in the system is used to collect activity data or physical sign of athletes and using the data for further analysis. Given that the battery life and memory resource of these devices are limited, the mobile edge computing architecture (MEC) is introduced, which is suitable for solving the problems faced by our scenario. As a complementary technology of cloud computing, MEC can make up for the shortcomings of cloud computing and effectively solve the pressure of mass data brought by the Internet of Things. MEC plays an important role in many application scenarios. For example, Mao et al. [5] studied the tradeoff between the power consumption of mobile devices and the execution delay of computing tasks in a multiuser MEC system. They proposed an online

algorithm based on Lyapunov optimization to determine whether the computation is performed locally or offloaded to the edge nodes. Li et al. [6] focused on QOE-optimized video delivery under the edge computing environment; they proposed an algorithm containing Integer Linear Programming (ILP) formula to solve this problem. Wang et al. [7] used edge computing structure to process social network data. Meanwhile, in order to optimize deployment strategy and maximize economic benefits, they designed a hybrid optimization model ITEM to reduce computing costs.

Edge computing is an extension concept of cloud computing. Its main purpose is to reduce the communication transmission costs between users and computing processing nodes. However, when the number of users increases, the computing requests from multiple users may exceed the capacity of the server, resulting in network congestion. In addition, the performance of the edge server is always fluctuating and changing under the influence of real-time changes on task requests load, power supply, and network conditions, which bring challenges to ensuring the performance of task execution. This situation can be abstractly described as a distributed resource optimization problem, and reinforcement learning is a classic solution to this issue. In 2015, Google DeepMind published a paper in Nature [8], which proposed a model that combined reinforcement learning (RL) and deep learning (DL), named deep reinforcement learning (DRL). Its outstanding performance in the field of game AI soon made DRL a new research focus. Since there are various kinds of tasks in edge computing, how to transfer the computing requests to the appropriate server so as to minimize the cost is a major optimization problem. Using simple models to solve this problem is relatively inefficient. RL algorithm can be used to optimize resource allocation under edge computing environment, which provides users with more efficient service.

The contributions of this work are summarized as follows:

(1) A data collection and analysis system for track and field sports is devised

(2) Deep deterministic policy gradient (DDPG) algorithm is used to reduce the service response time via edge computing structure

The rest of the paper is organized as follows: In Section 2, more related works are introduced, including information about the edge computing structure and the DDPG algorithm. Section 3 depicts the main structure of our system. In Section 4, the experiment results of the key performance indicators of the system are reported, and Section 5 gives the conclusion of this paper.

2. Related Works

2.1. Wearable Device-Based Track and Field Auxiliary Training System. With the development of information technology, the utilization of multimedia assists athletes in training. Even the real-time tactics analysis system of the game situation has been well known to people. A computer-aided system for throwing events training has emerged in the United States since the early 1970s. This system utilized cameras to record the athletes' technical movements; then the records are analyzed to get the angle data of the athletes' hand when throwing. Finally, it compares with the standard data to improve the technical action of throwing items. With the increasing maturity of MEMS, there are more sports assistance systems choosing to collect athletes' activity data and other pieces of relevant information through wearable devices. Thomas et al. [9] developed a training monitoring system for swimming, where multiple accelerometers were placed in the key positions of the human body, and semi-Markov model (SMM) was used in the system to accurately segment and label swimming activities with high accuracy. Valkova et al. [10] asked 75 mentally handicapped athletes to wear a GT3X activity recorder and record their activity data in two days before the competition and concluded that the local Special Olympic program is beneficial for people with a mental disability. Lee and Drake [11] interviewed 20 technical athletes, they combined athletic training and performance with the collection and evaluation of personally relevant data in an effort to better understand their own abilities, and the study also examines the individual relationships that technical athletes have with their data. Wachowicz and Mrozek [12] aligned the activity data collected by smart wearable devices with meteorological data to explore the influence of weather on athletes' on-the-spot performance and fuzzy join technique was used in this work. In a sentence, as smart devices continue to evolve, as more functions are integrated into such auxiliary systems, the system will consequently become overburdened.

2.2. Edge Computing. Edge computing is a form of distributed computing in which the main processing and data storage are placed at the edge nodes of a network. Edge nodes provide services based on the principle of proximity to quickly respond to the requests from the smart devices, so as to meet the basic needs in real-time business, security, privacy protection, and so on. The edge computing layer is at the top of the physical entity. In the MEC structure, a large number of small dedicated servers are deployed on the edge of the network close to the mobile user. Users can offload computation-intensive tasks to the edge cloud close to them and effectively run the application on their own mobile terminal. A common MEC architecture is shown in Figure 1, which mainly consists of 3 layers. The bottom layer consists of IoT nodes, which includes all kinds of intelligent devices and sensing nodes for data collection, including motion data, physical sign data, and environmental state data, to monitor subjects. Simple processing of the raw data may be conducted in this layer; otherwise, the data are cached and directly transferred to the edge computing layer for further data analyzing. The edge computing layer receives the data and processes the simple tasks and feeds back results down to the smart devices to display to the users, while some complex tasks are uploaded to the cloud layer for computing.

Edge computing has been applied in various fields in recent years. For example, Luan et al. [13] developed an auxiliary system with a smart medicine box, called MEMO

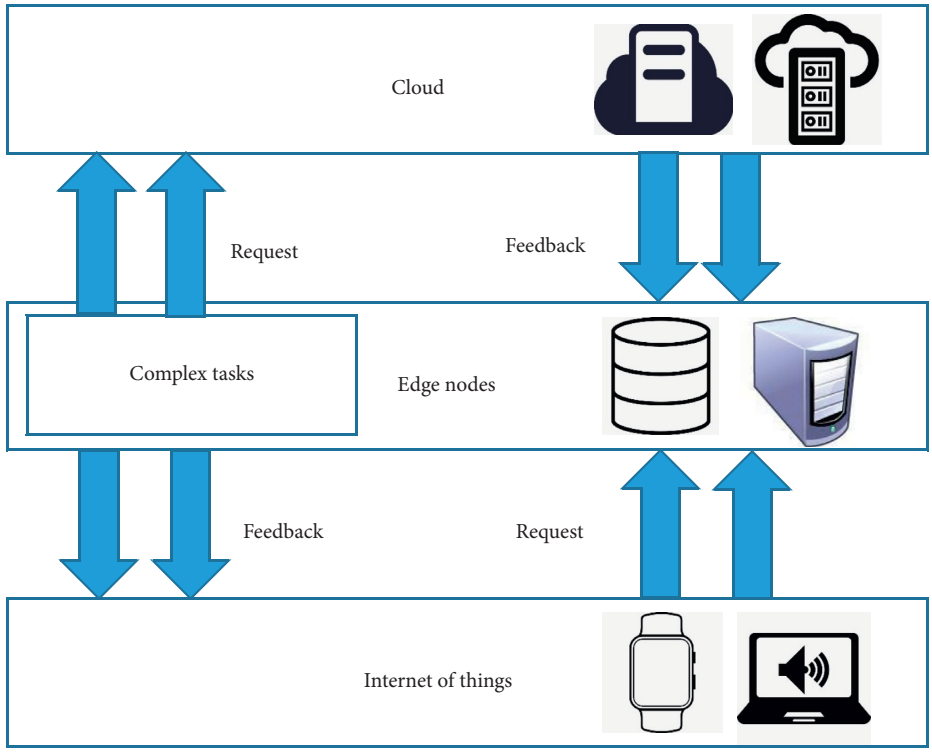


Figure 1: Classical edge computing architecture.

box system, in order to help the treatment of depression. The whole work is implemented by edge computing architecture, which improves the real-time performance of the system. Liu et al. [14] improved the algorithm for a patrol robot system and deployed it under the edge computing architecture, which greatly shortened the delay of the system information transmission. With the gradual popularization of 5G network, edge servers are faced with greater load, and how to properly optimize resource allocation will become a new research hotspot.

2.3. Reinforcement Learning. Reinforcement learning is a branch of machine learning methods, and its most significant feature is "learning from interaction." Agents constantly learn knowledge according to the rewards or punishments they receive after the interaction with the environment. Since the paradigm of RL learning is very similar to how humans learn knowledge, RL is seen as an important way to achieve universal AI. Combining RL with DL, deep Q network (DQN) has achieved amazing achievements in the field of AI game robots. For example, Yoon and Kim [15] applied DQN algorithm to visual fighting games, and the game AI trained by them achieved good results in competitions. The experiment results showed the potential of the DQN approach for the two-player real-time fighting game. Shen and Kurshan [16] proposed an enhanced threshold selection policy for fraud alert systems by applying DQN algorithm to fraud detection systems. This method not only reduced the loss caused by fraud but also improved the operating efficiency of the prewarning system. In addition, DRL can also play a role in combinatorial optimization issues such as resource allocation and task scheduling. For example, Wu et al. [17] proposed a hybrid learning strategy based on DQN for a multiuser multiserver MEC network. Simulation results showed that the hybrid approaches reach lower costs than other comparable groups. Yihang et al. [18] designed the Prioritized Memories Deep Q-Network (PM-DQN); this algorithm was applied to solve the joint routing and resource allocation problem in cognitive radio ad hoc network for minimizing the transmission delay and power consumption. Ma [19] constructed an application layout optimization strategy named min-cost to solve the layout problem of intelligent public transportation application in the city effectively. According to the simulation results, the proposed strategy can effectively reduce the total service cost of program on the basis of guaranteeing service delay.

3. Methodology

3.1. Overall Structure. This work is to design data acquisition and analysis system for track and field athletes. This paper implements the whole system using the edge computing architecture and optimizes the resource allocation of the whole system using the DRL algorithm, DDPG. The overall system architecture is shown in Figure 2.

According to Figure 2, the main operation of the system can be roughly divided into the following steps:

(1) Data acquisition: collecting various kinds of data from athletes, including acceleration, angular velocity, and heart rate, through sensor modules on the athletes. The data are cached on the sensing module and ready to be forwarded to the upper edge nodes.

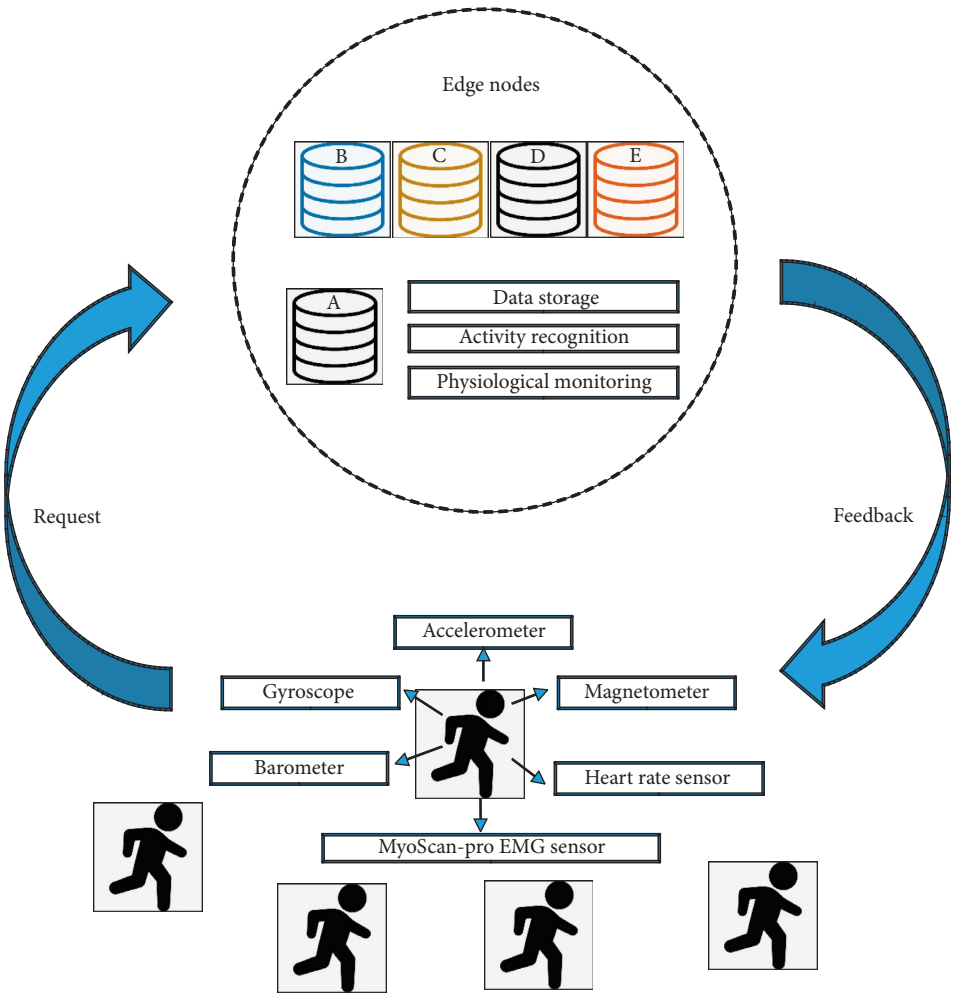


Figure 2: Overall system architecture.

(2) Request submit: the sensing module transmits data and requests information to the edge node A according to the preset policy.

(3) Request forwarding: the node A forwards subsequent tasks to other nodes after processing part of task requests. In this step, node A may find itself already overburdened and forward the requests to homogeneous node like *D* for processing.

(4) Task execution: the edge node A calculates the task and generates the final result information.

(5) Result feedback: the edge node A sends back the result down to the smart devices, which is responsible for displaying the data analysis results to the user.

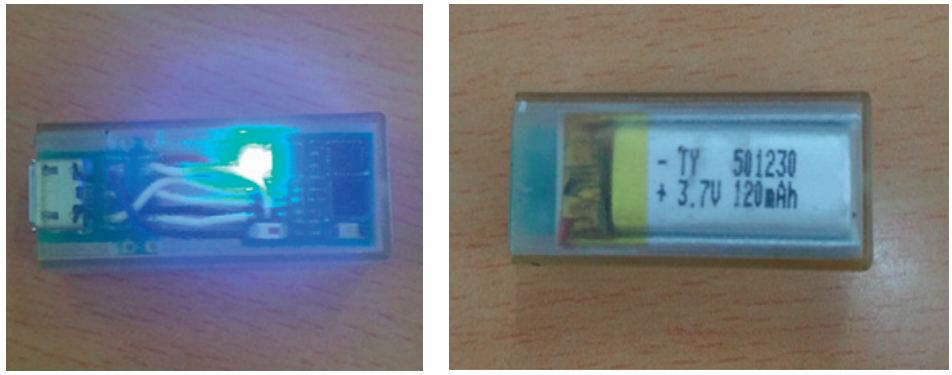
According to the steps descripted above, the pressure on edge server is mainly from multiple users' requests. Too many requests in a period of time may cause congestion at the edge nodes. Therefore, how to optimize the resource allocation of the whole edge layer is vitally important.

3.2. Data Collection and Analysis. This section introduces details of the athletes' activity data collection and analysis, where the MPU 9250 module is integrated into the sensing nodes as depicted in Figure 3.

The acceleration range of the module is ±2 g, ±4 g, ±8 g, and ±16 g, the gyroscope range is ±250, ±500, ±1000, and ±2000/s, and the magnetometer is up to ±4800 UT. According to previous experiments, the node can be used for about 2 weeks if it is used intermittently and for about 24 hours if it is used continuously. The node can be charged via micro-USB interface.

The sampling frequency of the node is preset as 50 Hz. Athletes wear 5–10 of these modules for data collection and place them in body positions such as the legs and wrists. Also, sensors for recording heart rate and EMG signals are used here. All data are sent to the nearest gateway node via Bluetooth and then are forwarded to the edge computing layer.

In order to reduce the energy consumption of the sensing equipment, all further data processing is carried out on the edge computing layer. Different types of tasks have their own processing methods. Here, this paper takes the activity analysis based on inertial sensing data as an example. The edge computing node synchronizes the data by timestamp and segments the data using the sliding window



(a) (b)

Figure 3: Activity data collection module. (a) Front. (b) Back.

algorithm. The data need to be feature extracted to form feature vector set in order to reduce the dimension of the inputs. In this paper, the commonly used features are extracted, including mean and variance, and their calculation formulas can be referred to in Table 1.

After the feature vector set is calculated, the pretrained machine learning model is deployed for activity analysis, such as identifying the category of the current activity. Here, this paper chooses to use support vector machine (SVM) algorithm to recognize activity and finally output the label that the activity belongs to. The whole process is summarized as Figure 4. The areas in the dashed frame are all processed by edge nodes.

3.3. DDPG-Based Dynamic Deployment Algorithm. The stable operation of the whole system depends on the reasonable system resources allocation. A dynamic resource deployment algorithm based on DDPG is proposed.

3.3.1. Problem Mapping. The problem of dynamic resource deployment is to adjust the deployment strategy of edge service resource allocation given the load of service requests at different times or on different servers. This issue with an example is shown in Figure 5.

Task loads at different time points of two edge nodes A and B are shown in Figure 5. The numbers on the left mean different types of service requests. The darker the color is, the more the service requests are. For example, there are anumber of requests for Service 4 on node A at time T, but there are many requests on node B at the same time. A proper allocation strategy should transmit the coming requests for Service 4 on node B to node A so as to reduce the load of B and ensure the service quality of the entire edge network.

Here, assume that the number of edge servers in the scenario is N and the number of service request types is M. In a continuous period of time, according to the request records of all services from mobile users on the edge server, the dynamic service deployment strategy σ is obtained, that is, matrix x of $N \times M$ is calculated for each period of time, as

Table 1: Feature extraction formulas.

Feature	Formula
Mean	$\overline{a}$
Variance	$\sum_{i=1}^{n} (a_i - \mu)^2$
Maximum	$\max(a_i)$
Minimum	$\min(a_i)$
Range	$\max(a_i) - \min(a_i)$
ZCR	$\sum_{i=1}^{n} \text{sig}(a_i > 0)$
Median	$\text{median}(a_i)$
MAD	$\text{median}(\lvert a_i - \text{median}(a_i)\rvert)$
Information entropy	$-\sum_{i=1}^{m} (p_i * \log(p_i))$
Kurtosis	$E[(a_i - \mu)^4/\sigma^4]$
Skewness	$E[(a_i - \mu)^3/\sigma^3]$
Coefficient	$\text{cov}(X, Y)$

ZCR: zero crossing rate; MAD: absolute median difference.

shown in formula (1), and each element p_{ij} in X represents the ratio of service resources for deploying service s_j on the edge server node$_i$.

$$x = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & p_{22} & \cdots & p_{2M} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N1} & p_{N2} & \cdots & p_{NM} \end{bmatrix}. \quad (1)$$

Based on x, the whole service layer adjusts service deployment through a certain policy; then y is defined as shown in

$$y = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}. \quad (2)$$

Y represents the service allocation adjustment of a certain kind of service. For example, a_{ij} represents the number of requests forwarded by the ith node to the jth node. The function of system deployment strategy σ is to establish the relationship between x and y; that is, when the system state is x, the probability of taking action y is

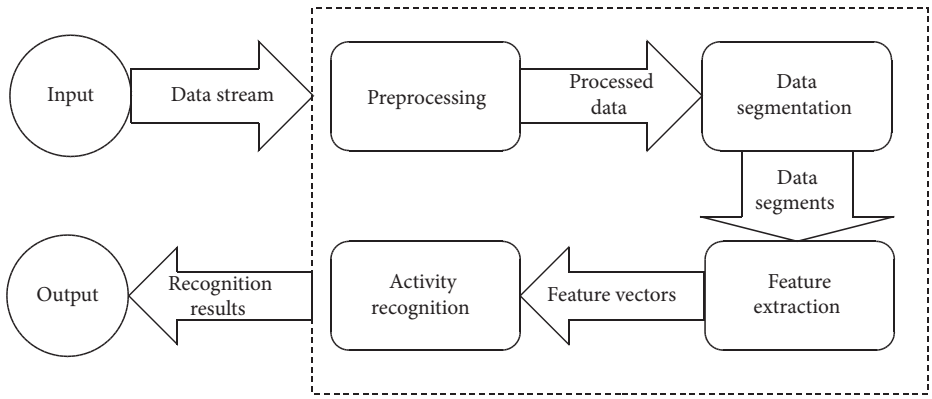


Figure 4: Process of activity recognition.

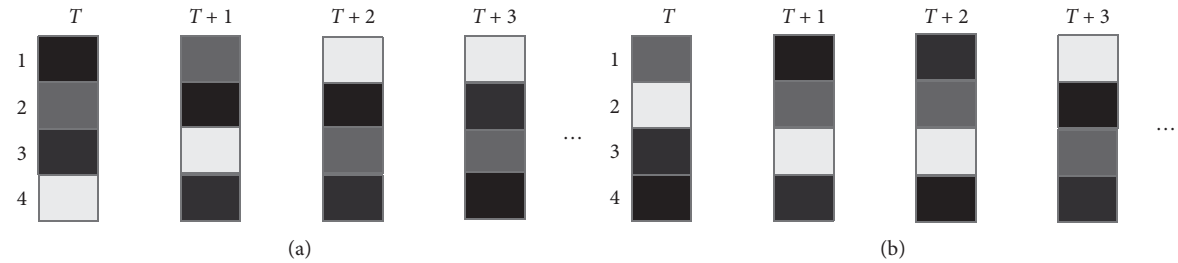


Figure 5: Example of load conditions on edge severs.

$$\sigma(y|x) = p(y|x). \tag{3}$$

That is, σ represents the conditional probability distribution of the system performing y operation in x state. Meanwhile, variable r is defined to evaluate the system behavior, as shown in

$$r = \begin{cases} 1, & \text{if } ct(x_t, y_t) \le 0, \\ -1, & \text{if } ct(x_t, y_t) > 0, \end{cases} \tag{4}$$

where ct is the mean response time of the edge layer. When $ct \le 0$ means that the strategy adopted at time t can shorten the response time, then r is positive. Otherwise, r is negative. The ultimate goal of r is to minimize the average service response time.

To simplify the model, the following assumptions are presented:

(1) There is no interaction or dependency between services

(2) In network transmission, the length of request message and response message is approximately same

(3) The unit transmission delay between the sensing module and the edge node is fixed, and so is the delay between the edge node and other edge nodes

3.3.2. Algorithm Description. RL methods mainly have two branches, namely, policy-based approach and value-based approach. The DDPG algorithm is a combination of the two branches, which has the advantages of them.

As shown in Figure 6, DDPG algorithm is mainly composed of environment, experience replay memory unit, actor network module, and critic network module. The environment is the interaction space of the agent. During the interaction, the agent obtains the interaction samples and stores them in the experience replay memory unit for training the networks. In order to optimize the learning process, DDPG algorithm takes the idea of DQN algorithm and builds a pair of artificial neural networks with identical structure for the networks, namely, the online part and the target part. Online network is used to train and update network parameters, while target network uses periodic soft update strategy to follow online network and assist online network in training. The ultimate goal is to optimize the deployment strategy σ discussed in the previous section.

According to the process shown in Figure 6, the overall algorithm steps are described as follows:

The system starts to run and conducts initialization. The network parameters in actor and critic are initialized, and the online network parameters are copied to the target network, that is, $\xi_a' \leftarrow \xi_a$, $\xi_c' \leftarrow \xi_c$. Set the size of experience replay memory as U and the size of minibatch observation data as V.

For each period, we have the following:

According to strategy σ and the current system state x_t, select an action y_t and instruct the system to execute the action. Here, $y_t = \sigma(x_t|\xi_a) + N_t$, N_t is the random noise.

The system executes y_t and obtains the reward value r_t through calculation and then enters a new resource allocation state x_{t+1}.

A state transition data set of size V is randomly sampled from experience replay memory as a small-

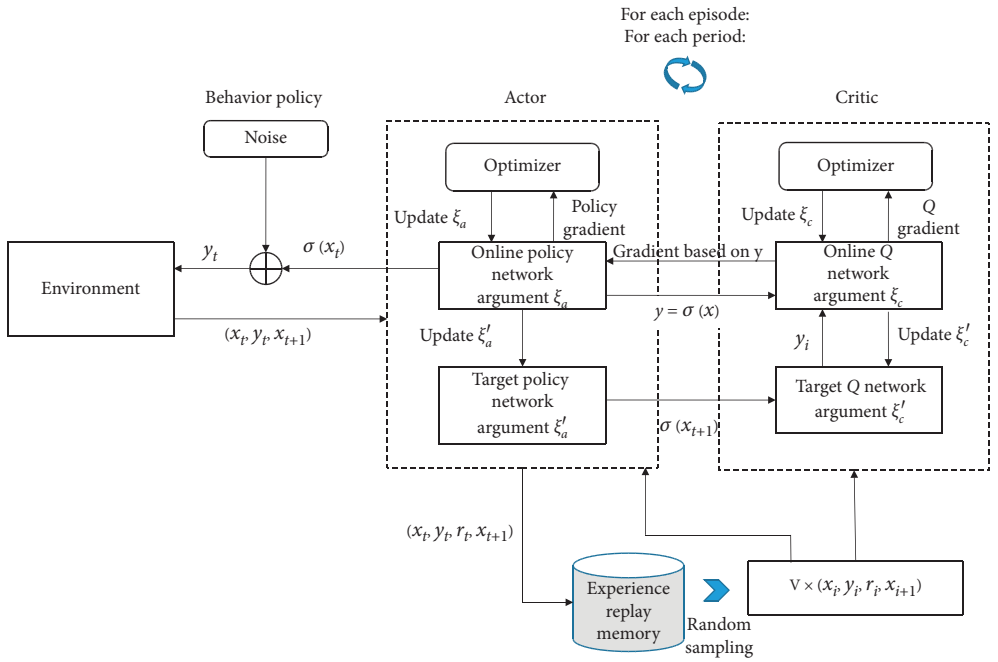


Figure 6: Process of the DDPG.

batch training data set for online policy network and online Q network. Each element in the set is expressed as (x_i, y_i, r_i, x_{i+1}), $i+1 \le V$.

Calculate the gradient of online Q network, which is represented as ∇_{ξ_c}Loss. Similar to the supervised learning method, Mean Square Error (MSE) is used here to calculate Loss. The formula is as follows:

$$\text{Loss} = \frac{1}{N}\sum_i \left(y_i - Q(x_i, y_i|\xi_c)\right)^2, \tag{5}$$

where Q is the state action value function, and its expression is

$$Q(x, y) = (1-\gamma)c(x, y) + \gamma \sum_{x \in X} P\{x'|x, y\}V(x'), \tag{6}$$

where c represents the cost of executing action y when the system state is x. $P\{x'|x, y\}$ is the probability that the system becomes x' after action y. V is the optimal state value function obtained by the Behrman optimization equation, and the formula is

$$\%\ V(x) = \min_{y \in Y}\left\{(1-\gamma)c(x, y) + \gamma \sum_{x \in X} P\{x'|x, y\}V(x')\right\}, \quad \forall x \in X. \tag{7}$$

Finally, based on the standard backpropagation method, ∇_{ξ_c} is obtained.

Update the value of ξ_c and critic network through optimizer.

Calculate the gradient of policy network, which is represented as $\nabla_{\xi_a} J$. The formula is

$$\nabla_{\xi_a} J \approx \frac{1}{N}\sum_i \nabla_y Q(x, y|\xi_c)|_{x=x_i, y=\sigma(x_i)} \nabla_{\xi_a}\sigma(x|\xi_a)|x_i. \tag{8}$$

Update the value of ξ_a and actor network through optimizer.

Soft update the Target Policy Network and Target Q Network; that is,

$$\begin{aligned} \xi'_c &\leftarrow \tau\xi_c + (1-\tau)\xi'_c, \\ \xi'_a &\leftarrow \tau\xi_a + (1-\tau)\xi'_a, \end{aligned} \tag{9}$$

where τ is a small value, which is beneficial to the stability of target network in the update.

4. Experiment and Results

4.1. Introduction of Experiment Data. This paper plans to test the effectiveness of the resource deployment algorithm and tests the analysis ability of the system through the recognition results of athletes' activity data. First, the data from 40 track and field athletes in different events are collected, including sprint, race walk, javelin throw, pole vault, high

Table 2: Configuration of SVM.

Parameter	Value
Kernel function	RBF
Class weight	Balanced
Decision function shape	ovr
Gamma	0.2
C	0.8

jump, and long jump. Data are collected for 3 min from each athlete, and the obtained data set is used as training data for the classifier. After training, the classifier SVM is deployed on the edge node waiting to process the task request. Table 2 gives the parameter configuration of SVM.

Then, three edge servers around the training site are deployed to handle service requests. The server configuration is shown in Table 3.

Finally, DDPG algorithm on the edge server is deployed, and the parameter settings of the algorithm are shown in Table 4.

Meanwhile, we set up two control groups for comparison with DDPG-based algorithm. One is the average-based deployment algorithm, the idea is that, for each service, the proportion of resources deployed on the server is the same. The other is the frequency-based algorithm, which allocates resources according to historical service requests.

4.2. Performance of DDPG-Based Dynamic Deployment Algorithm

4.2.1. Algorithm Convergence. This experiment mainly tests the convergence of the algorithm. We add a hidden layer with 128 neurons to the actor network and critic network of DDPG algorithm. Figure 7 shows the function loss trend after 10,000 episodes of training.

According to Figure 7, loss value of the network fluctuates greatly in the first hundreds of episodes. As the number of network training episodes increases, the loss value decreases gradually. After reaching 5000 episodes, the loss value basically converges to (0, 1), and the fluctuation becomes small. It suggests that the convergence of the algorithm based on DDPG proposed in this paper can be guaranteed.

4.2.2. Algorithm Comparison Experiment. In order to compare resource allocation algorithms, we collect user request data for several days and used the above three algorithms to carry out resource allocation on edge nodes.

Average-based method does not consider the historical service request information, so it directly sets the resource ratio of the edge server service to be the same. Frequency-based method determines the current resource allocation based on the configuration of the previous period. DDPG-based method allocates the resource based on the preset network parameters. The three algorithms have experimented with the same test set, and the average response time of the server was recorded, as shown in Figure 8.

Table 3: Configuration of edge nodes.

Parameter	Description	Value
N	Number of edge nodes	3
R	Resource capacity	[500, 800]
F	CPU cycle frequency	3.2 GHz
CMI	Average number of cycles per instruction execution	600

Table 4: Parameter settings.

Parameter	Description	Value
U	Experience replay memory size	300
V	Minibatch size	32
Γ	Reward attenuation factor	0.9
λ_a	Learning rate of actor network	0.001
λ_c	Learning rate of critic network	0.002

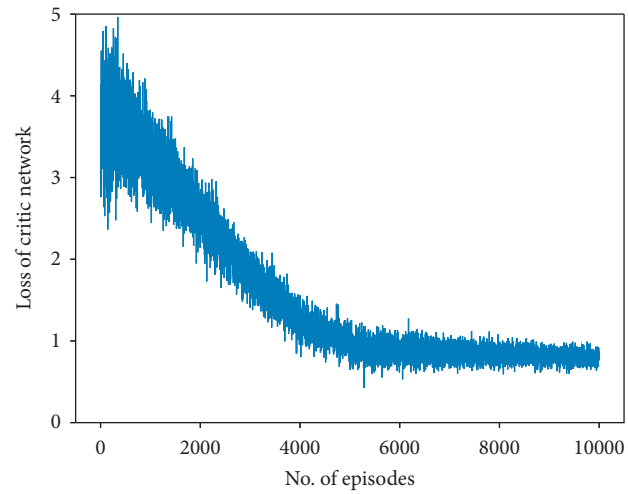


Figure 7: Loss trend of critic network.

Results in Figure 8 suggest that the average response time of the DDPG algorithm is the lowest, which can be controlled within about 1 s. Moreover, the user requests during the whole observation period are relatively stable. Relatively, there are more requests and the response time is relatively high only during 8 : 00–11 : 00 in the morning and 13 : 00–15 : 00 in the afternoon. To conclude, the DDPG algorithm can achieve a better deployment strategy through learning and exploring. It performs well in the track and field scenario in our work.

4.3. Performance of SVM. In order to test the system's ability on activity analyzing, the collected data are identified by the classification algorithm deployed on the edge nodes. In addition to SVM algorithm, we also introduce two other classifiers: decision tree (DT) and k-nearest neighbor (KNN) algorithm. They are frequently used in similar pattern recognition works, which are typical machine learning classifiers. We put the classification results as confusion matrix inFigure 9.

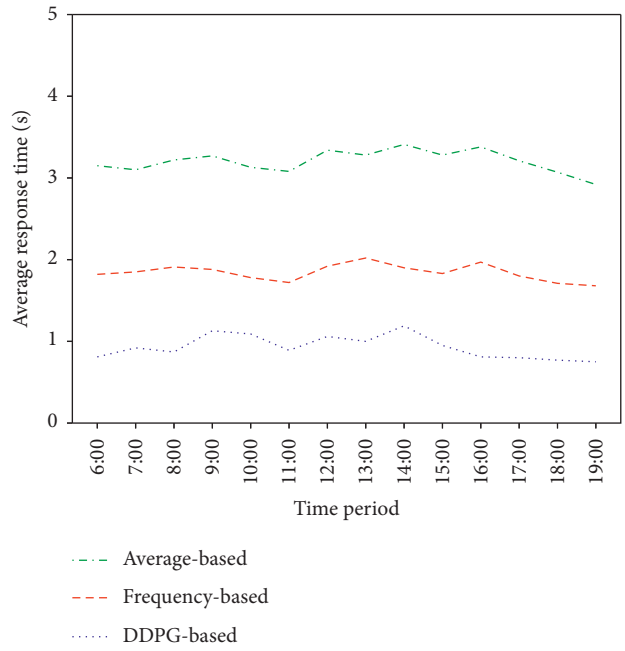


Figure 8: Comparison of 3 methods on average response time.

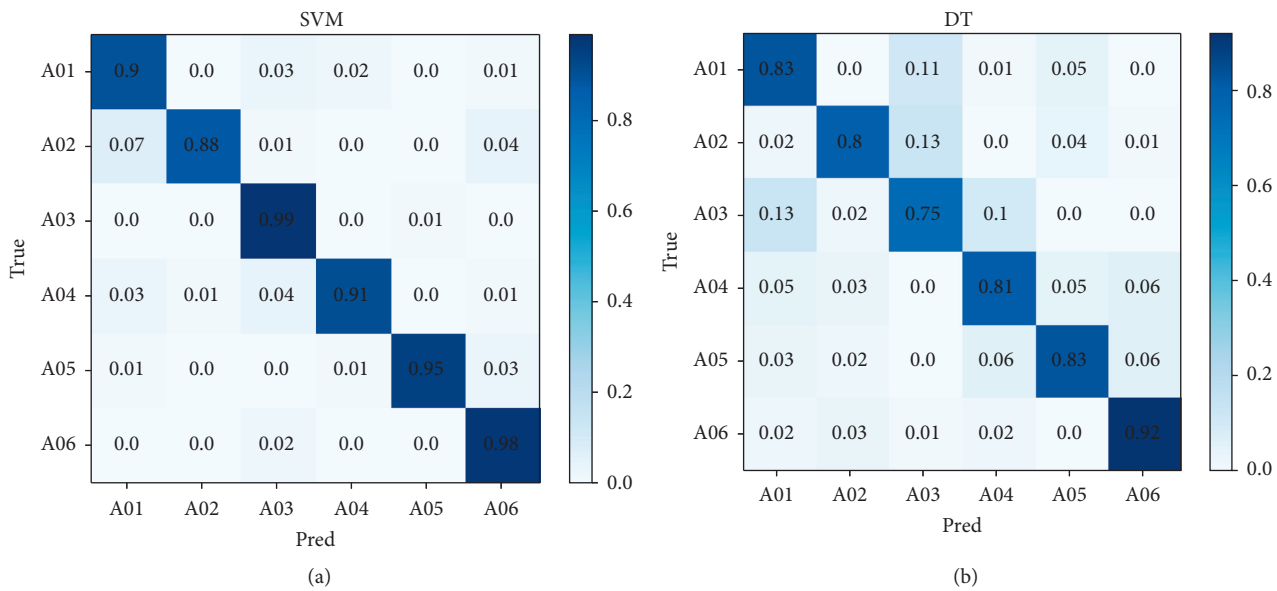


Figure 9: Continued.

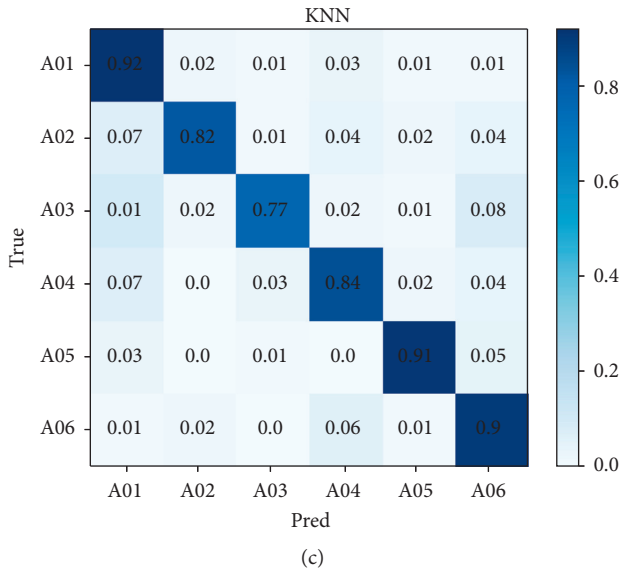


Figure 9: Recognition accuracy of SVM (a), DT (b), and KNN (c).

In Figure 9, A01-A06 represents the 6 kinds of activities: sprint, race walk, javelin throw, pole vault, high jump, and long jump. The darker the color block on the diagonal of the matrix is, the higher the recognition accuracy of the activity is. The recognition accuracy of SVM for all categories reached more than 88%, higher than DT and KNN. Accuracy of KNN for A01 is slightly higher than that of SVM. Given the overall recognition accuracy, SVM (93.5%) is better than DT (82.3%) and KNN (86.0%). It is better to complete the data analysis task using the SVM in our scenario.

5. Conclusion

This paper uses edge computing technology to implement data acquisition and analysis system for track and field athletes. For optimizing resource allocation and reducing the response latency, this paper introduces the DDPG algorithm to implement a dynamic resource allocation algorithm. Compared with other algorithms in the control group, the proposed algorithm has a lower average response time, within 1 s under the test environment. Meanwhile, the SVM algorithm is deployed in the edge server for activity analysis. However, the types of activities discussed in this work are relatively small. Therefore, the activity sets will be expanded in future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] E. Bobkova and E. Parfianovich, "Neural networks for forecasting and modeling training in track-and-field athletics," *Human Sport Medicine*, vol. 18, no. 5, pp. 115–119, 2018.

[2] M. Cucco, "Smart athletic training system," US20160271447A1, 2016.

[3] B. Ma, S. Nie, M. Ji, J. Song, and W. Wang, "Research and analysis of sports training real-time monitoring system based on mobile artificial intelligence terminal," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8879616, 10 pages, 2020.

[4] S. Guo, "The design and application of system software about test evaluation and training of the athlete's movement function," in *Proceedings of the 5th International Conference on Computer, Automation and Power Electronics, CAPE 2017*, pp. 110–113, Dalian, China, October 2017.

[5] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, December 2016.

[6] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-assisted video delivery architecture for wireless heterogeneous networks," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 534–539, Heraklion, Greece, July 2017.

[7] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, Honolulu, HI, USA, April 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] O. Thomas, P. Sunehag, G. Dror et al., "Wearable sensor activity analysis using semi-Markov models with a grammar," *Pervasive and Mobile Computing*, vol. 6, no. 3, pp. 342–350, 2010.

[10] H. Valkova, L. Qu, and F. Chmelik, "An analysis of the physical activity of special olympic athletes with the use of an accelerometer," *Journal of US-China Medical Science*, vol. 11, no. 4, pp. 176–187, 2014.

[11] V. Lee and J. Drake, "Physical activity data use by technoathletes: examples of collection, inscription, and identification," in *Proceedings of the 10th International Conference of*

the Learning Sciences: The Future of Learning, vol. 2, pp. 321–325, Sydney, Australia, July 2012.

[12] A. Wachowicz and D. Mrozek, "Fuzzy join as a preparation step for the analysis of training data," *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis*, Springer, Berlin, Germany, 2019.

[13] L. Luan, W. Xiao, K. Hwang, M. S. Hossain, G. Muhammad, and A. Ghoneim, "MEMO box: health assistant for depression with medicine carrier and exercise adjustment driven by edge computing," *IEEE Access*, vol. 8, pp. 195568–195577, 2020.

[14] X. Liu, B. Dong, P. Li, B. Yuan, and K. Wang, "Research and application of image recognition of substation inspection robots based on edge computing and incremental learning," *ROBOMECH Journal*, 2021.

[15] S. Yoon and K. Kim, "Deep Q networks for visual fighting game AI," in *Proceedings of the 2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 306–308, New York, NY, USA, August 2017.

[16] H. Shen and E. Kurshan, "Deep Q-network-based adaptive alert threshold selection policy for payment fraud systems in retail banking," 2020, http://arxiv.org/abs/2010.11062.

[17] Y.-C. Wu, T. Q. Dinh, Y. Fu, C. Lin, and T. Q. S. Quek, "A hybrid DQN and optimization approach for strategy and resource allocation in MEC networks," *IEEE Transactions on Wireless Communications*, 2021.

[18] D. Yihang, Z. Fan, and X. Lei, "A kind of joint routing and resource allocation scheme based on prioritized memories-deep Q network for cognitive radio ad hoc networks," *Sensors*, vol. 18, no. 7, p. 2119, 2018.

[19] X. Ma, "Optimal deployment of applications based on reinforcement learning in edge computing scenarios," *Computer Engineering and Design*, vol. 42, no. 1, pp. 15–23, 2021.