International Journal of Engineering

Journal Homepage: www.ije.ir

A Latency Reduction Method for Cloud-fog Gaming based on Reinforcement Learning

S. M. Jameii*[a], K. Khanzadi[b]

[a] Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran
[b] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

PAPER INFO

Paper history:
Received 27 January 2022
Received in revised form 27 March 2022
Accepted 01 April 2022

Keywords:
Reinforcement Learning
Computation Latency
Cloud-fog Gaming
Principal Component Analysis

ABSTRACT

Unlike traditional gaming where a game run locally on a user's device, in cloud gaming, an online video game runs on remote servers and streams directly to a user's device. This caused players to become independent of having high hardware resources in their local computers. Since video games are a kind of latency-sensitive application, cloud servers far from users are not suitable. In fog computing, fog nodes are in the vicinity of users and can reduce the latency. In this paper, a latency reduction method based on reinforcement learning is proposed to determine which computing fog node can run the video games with the lowest latency. In the proposed method, a Principal Component Analysis (PCA) based approach is used to extract the most important features of each video game as the input of the learning process. The proposed method was implemented using Python. Experimental results show that the proposed method compared to some existing methods can reduce the frame latency and increase the frame rate of video games.

doi: *10.5829/ije.2022.35.09c.01*

1. INTRODUCTION

Nowadays, users are capable of executing video games on different platforms such as smartphones, personal computers, etc. Execution of these video games is independent of the computation resources of the users' local devices. In 2009, cloud gaming was introduced and its main idea was to run the video game on remote servers and stream them directly to a user's device [1]. Hence, video games would be developed for computers in the cloud rather than for personal computers. Afterward, game producers rent the cloud's computer and bandwidth. User Experience (UX) is an important aspect of playing video games in which latency is one of the most important issues in UX for playing video games in Cloud-Fog Computing (CFC). In 2013, Huang et al. [2] proposed the first open-source cloud gaming system called "Gaming anywhere". Cloud gaming systems such as Gaikai, OnLive, and Stream My Game, had been proposed before Gaming anywhere but these cloud gaming systems suffered from inappropriate response time. Bonomi et al. [3] proposed the first fog computing paradigm in 2012. Fog computing has characteristics such as low latency which can bring services to the edge of the network. Another advantage of fog computing is geographical distribution. In a fog computing environment, many nodes in each region can serve as sufficiently as possible. We should consider another type of latency called system latency for playing a video game on the fog and cloud nodes[1]. This latency is the delay between the mouse or keyboard actions and the resulting pixel changes on the user's display and should be considered for calculating the total latency of video games.

In this paper, a latency reduction method based on reinforcement learning is proposed to determine which computing fog node can run the video games with the lowest latency. At the first of the proposed method, a Principal Component Analysis (PCA) based approach is used to extract the most important features of each video game as the input of the learning process.

Reinforcement learning is dynamically learning by

*Corresponding Author Institutional Email: Jamei@qodsiau.ac.ir (S. M. Jameii)

[1] https://www.nvidia.com/en-us/geforce/news/reflex-low-latency-platform

Please cite this article as: S. M. Jameii, K. Khanzadi, A Latency Reduction Method for Cloud-fog Gaming based on Reinforcement Learning, ***International Journal of Engineering, Transactions C: Aspects***, Vol. 35, No. 09, (2022) 1674-1681

adjusting actions based on continuous feedback to maximize a reward [4]. Since the objective of this paper is dynamically select the best fog node in a distributed manner to play the video game, we can have a learning agent for each fog node to interact with the environment. This kind of learning expresses how states can be mapped to actions to maximize reward signals. This way the agent does not tell what actions it should do, but it discovers which actions have the most reward signal.

Markov Decision Processes (MDP) is a mathematical framework to describe an environment in reinforcement learning. It provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker [5]. Since the problem of this paper will be solved by reinforcement learning, the Markov chain is a very useful framework to model this problem based on by taking a sequence of actions. From a time perspective, MDPs can be sub-divided into two categories named Discrete-Time Markov Chain (DTMC) and Continuous-Time Markov Chain (CTMC) [6]. A random process can be defined as a chain of random variables. There is a feature named Markov property that refers to a memory-less property of random processes. A random process has Markov property if the future probability distribution is dependent only on the current state but not on a sequence of events that preceded it [6]. MDPs, consist of three aspects: sensing, action, and goal. An agent must sense the state of the environment and then consider actions that take effect on the state. Each method that can solve this kind of problem, is known as the reinforcement learning method.

Since each video game has many features (more than 26 features) and analyzing these features incurs high overhead, we have used PCA to reduce the number of features. So essential features which are a lot informative will be selected and less informative features will not be considered. PCA combines essential features with a substituted feature which leads to smaller sets of features. Since the feature reduction is done before starting the game and it is done on the machines of the users (not on fog nodes), it does not incur significant overhead.

The contributions of this paper are as follows:

1) A PCA-based approach is used to extract the most important features of each video game to be considered as the input of the learning process.
2) A distributed reinforcement learning process is proposed that can compute the score of each fog nod and select the best fog node to play each video game based on its characteristics and consuming resources.
3) The proposed method can significantly reduce the frame latency and increase the frame rate of video games.

The rest of our paper is organized as follows: In section 2, we overview the related works. In section 3, we present the proposed algorithm. Section 4 is the simulation and experimental results. Finally, we conclude the paper in section 5.

2. RELATED WORKS

In this section, some of the works that used reinforcement learning for decision-making in cloud-fog environments were reviewed. Talaat et al. [7] used reinforcement learning for resource allocation and process migration. For resource allocation, the reinforcement algorithm selects the best fog server based on the fast response time. For process migration, the reinforcement algorithm selects the process for migration based on the process weight to designate priority for selecting the suitable process for migration. Zhang et al. [8] proposed a framework named EdgeGame to adjust video bit rate adaptively to match the network dynamics. Also, in the paradigm of cloud gaming to compensate for the dynamic nature of networks; they used deep reinforcement learning to adjust the traffic from the edge nodes to the users and to accommodate the varying bandwidth in the dynamic network. In the paradigm of Mobile Edge Computing (MEC), Zhang and Zheng [9] proposed a technique for task migration based on the Deep-Q network. In their work, the agent can learn optimal task migration policy from previous experiences without the need for a user's mobility pattern in the future. Chen et al. [10] proposed an adaptive real-time video game streaming policy in the dynamic network based on deep reinforcement learning to control bit rate adaptively. In 2018, Chen et al. [11] proposed a computation of floading algorithm based on deep q-network named Darling to learn the optimal policy without knowing prior knowledge of network dynamics. This algorithm was proposed for the MEC in which mobile devices are not fully capable of computing intensive tasks locally. Thus there should be a policy to determine whether to compute the tasks locally or offload them to the MEC server, considering the dynamic nature of the network. Dutreilh et al. [12] proposed an automatic decision-making approach for resource allocation without previous knowledge of the application performance model. In their work, the agent learns to add, maintain or reduce the number of VMs allocated to the application. In 2018, Wang et al. [13] tried to make a tradeoff between energy consumption and service delay in vehicular networks. They proposed a novel model to depict the users' willingness of contributing their resources to the public. In 2018, Dinh et al. [14] proposed a model-free reinforcement learning offloading mechanism in which mobile users can learn their long-term offloading strategy to maximize their long-term utilization. This mechanism

was proposed to prevent a scenario in which many mobile users offload their tasks to the same edge node at the same time. In 2019, Huang et al. [15] proposed an online offloading framework that utilizes deep reinforcement learning that learns the offloading decisions from the experience. This offloading policy was done in the MEC network which could decide to compute tasks locally or offload them to the MEC server. Because the channel state conditions are time-varying in wireless networks, the offloading decisions and resource allocations should adapt themselves to these dynamic conditions.

3. THE PROPOSED METHOD

In this section, a latency reduction method based on reinforcement learning is proposed to determine which computing fog node can run the video games with the lowest latency. At first in subsection 3.1, a Principal Component Analysis (PCA) based approach is used to extract the most important features of each video game as the input of subsection 3.2.

3. 1. Feature Selection using PCA PCA is a feature reduction method in which the original data transfers to a smaller space which leads to the reduction of the features. On the other hand, PCA combines essential features with a substituted feature which leads to a smaller set of features. In this section, by utilizing the PCA method, we try to apply PCA to reduce the 26 features of video games to 2 primary components and then extract the most important features from them. The steps of applying PCA to reduce the features are as follows:

1. Assume X_1, X_2, … ,X_Z are feature set and each X_i represents as N×1 vectors. (Z is the total number of features and is assumed equal to 26 and N is the number of rows in the dataset. At the first step, each feature is normalized between 0 to 1).
2. The average vector is calculated as follows:

$$\bar{X} = \frac{1}{Z}\sum_{i=1}^{Z} X_i \quad (1)$$

3. For each vector, subtract the average vector of it and produce the matrix A=[Φ_1, Φ_2. …. Φ_Z] (N×Z matrix) as follows:

$$\Phi_i = X_i - \bar{X} \quad (2)$$

4. Covariance matrix is calculated as follows:

$$Cov = \frac{1}{Z}\sum_{i=1}^{Z} \Phi_i \, \Phi_i{}^T \quad (3)$$

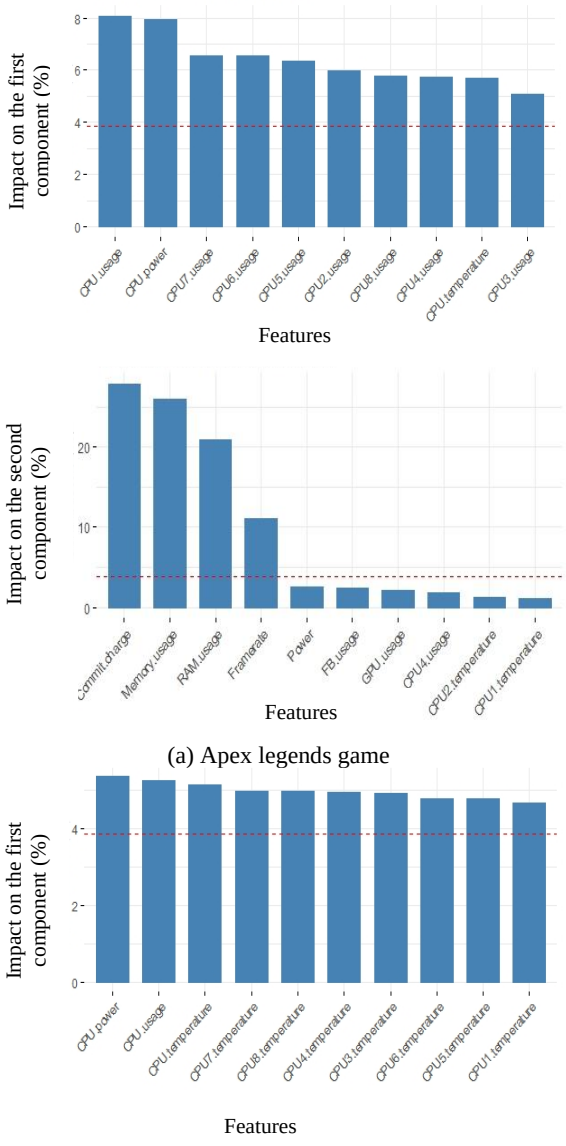
where $\Phi_i{}^T$ is the transformation matrix of Φ_i

5. From the covariance matrix, we can compute eigenvalues and eigenvectors and sort the eigenvectors ascendingly. The highest value of eigenvalue means the highest significance of the corresponding features. Top 2 eigenvectors with the highest eigenvalue form the principal components of the data set.

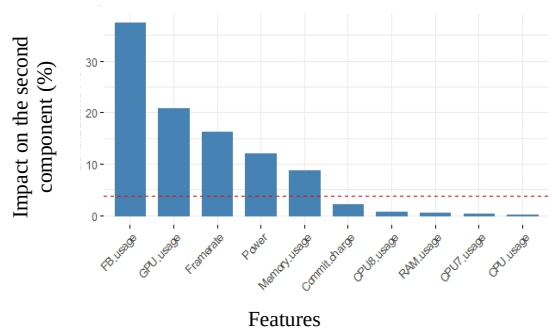
The impact of the important features on the 2 primary components categorized by video games are depicted in Figure 1.

In Table 1, the selected features for each video game that have more impact on the primary components are demonstrated.

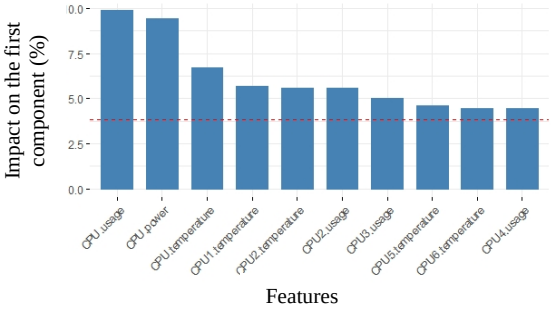
3. 2. Fog Node Score Calculation Algorithm1 calculates the score for each fog node. The less difference between video game consumed resource and fog node remainder resource is, the more score the related fog node

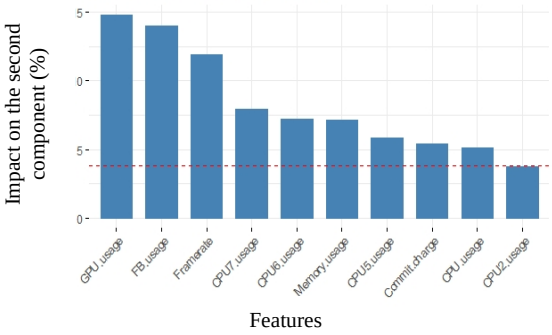


(a) Apex legends game

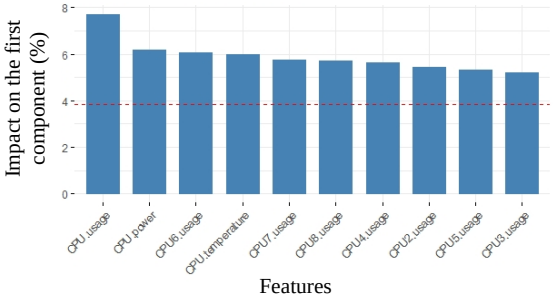


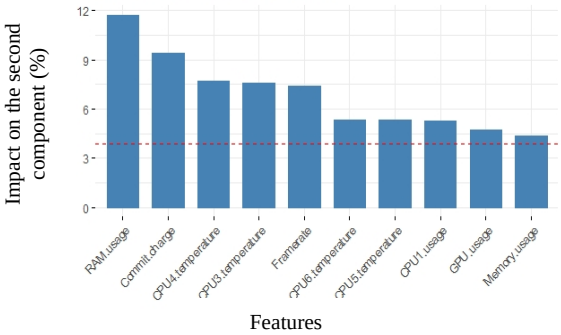
(b) Battlefield 4 game



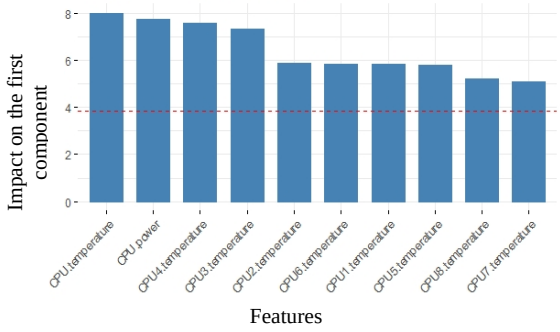


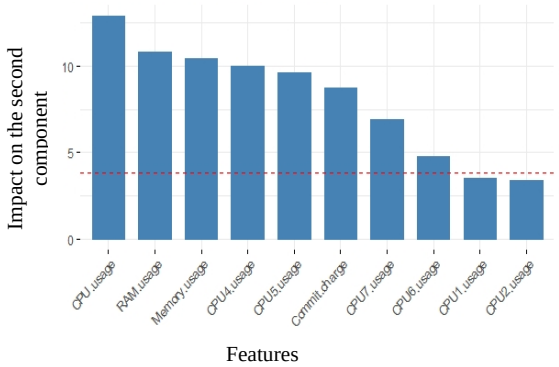
(c) Warthunder game





(d) Forza horizon 4 game





(e) CS: GO game

Figure 1. The impact of the important features on the 2 primary components categorized by video games

TABLE 1. Selected features for each video game

Video game	Selected features
Apex Legends	CPU.usage, CPU.power, Commit. charge
Battlefield 4	CPU.power, CPU.usage, FB.usage
Warthunder	CPU.usage, CPU.power, GPU.usage
Forza horizon 4	CPU.usage, CPU.power, RAM.usage
CS: GO	CPU.temprature, CPU.power, CPU.usage

Algorithm 1

- **Input:** Randomly generated numbers for fog nodes resources and the captured resources from video games.
- **Output:** Fog node scores for each video game

For each video game
  For each fog node
    Calculate the difference between video game consumed resources and the remainder of fog nodes resources
    Calculate the plural of differences of all fog node resources
    Sort plural of differences in ascending manner
    Give more scores to fewer plural differences

Return the calculated fog node score.

will receive. To be more clear, for example, two fog nodes have been compared to see which of them is a better fog node in terms of remainder resources.

In this paper, only 3 fog nodes have been considered, thus scores are designated between 0 to 2. Only features that have been selected by PCA are considered for selecting the fog node. In the next subsection, the produced scores for fog nodes are used as input for the reinforcement algorithm for selecting the best fog node.

3. 3. Calculating the Fog Node Selection Priorities using Reinforcement Learning In this subsection, an algorithm based on reinforcement learning is proposed for selecting the best fog node. After calculating the scores of the fog nodes by Algorithm 1, the *Q* matrix will be initialized. Each element of this matrix has two parts which are called the actions 0 and 1. With these actions, we can say whether the agent selects the current fog node or selects another fog node with a higher score. Each action has its related q-value. The action with a higher q-value will be selected. The agent periodically monitors the environment that consists of fog nodes. The agent learns which fog nodes could receive more scores. The problem is modeled as a Markov Decision Process (MDP) in Figure 2. The MDP that models our approach to select the best fog node is defined as Equation (4):

$$M=<B,A,T,P,\gamma> \quad (4)$$

in which,

$B = \{(n, p) \mid 1< n < n_{max} \wedge 1< p < p_{max}\}$ is the state of the MDP where n is the fog node index and p is the score which is calculated for fog node.

- $A = \{a \mid 0 < a \leq 1\}$ is the action set. When the agent compares two fog nodes' scores, if the first fog node has higher a score than the second fog node, the action 0 will be selected which means the selection of the first fog node with the probability of Ps_1s_1 or Ps_2s_2. But if the second fog node has a higher score than the first fog node, action 1 will be selected which means the selection of the second fog node with the probability of Ps_1s_2 or Ps_2s_1.

P is the probability distribution *p (r| s, a)* of observing reward *r* when the agent is in state *s* and action *a* is taken.

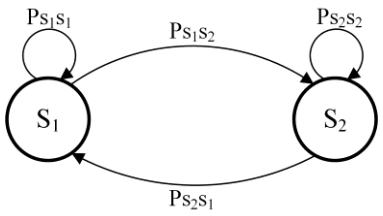


Figure 2. MDP for solving the stated problem

- γ, $0< \gamma <1$ is defined as a discount factor that determines how important a future reward is. When it has a value near 0, the agent tends more to the current state and when it has a value near 1, the agent tends more to the future state.
- T is the probability distribution *P (s'| s, a)* of transition to state *s'* when the agent is in state *s* and action *a* is taken.

According to Dutreilh et al. [12], *T* and *P* are difficult to estimate because they require heavy experimentation and measurement. Due to this fact and to overcome these limitations, reinforcement learning has been proposed to learn these two parameters by interaction with the environment. In the proposed method, Q-learning has been used as one of the reinforcement learning approaches. After calculating each score of fog nodes by Algorithm 1, the Q-matrix will be created with mentioned characteristics. In this step, the agent interacts with the environment and then updates the current state and q-value in Q-matrix. Then the action with a higher q-value will be selected. Eventually, the q-values in Q-table will be updated. The agent updates Q-values by Q-learning formula which is as Equation (5):

$$Q(CurrentState, \text{Action}) = (1 - LearningRate) * Current_q + LearningRate * (reward + Discount * MaxFuture_q) \quad (5)$$

Now the Q-table has been formed and we can use its policies for selecting the fog nodes. Algorithm 2 for calculating the Q-table is as follows:

Algorithm 2

- **Input:** Calculated scores by Algorithm 1 for each video game.
- **Output:** Q-table policies for selecting the best fog node

For each score calculated score by Algorithm 1
Create fog node objects and assign the related scores
Initialize the Q-table
Agent updates the current state and actions by interaction with the environment
Observing possible actions from Q-table, then the agent selects the action with higher q-value
Updating q-values in q-table
If learning process reward == pre-designated reward:
Move to the next episode
Else:
Continue steps from the beginning
The end of the episodes

4. PERFORMANCE EVALUATION

4. 1. Experimental Setup The proposed method was implemented using *Python 3.6*. At first, we ran 5 video games distinctly to capture resource consumption by *MSI Afterburner*. The obtained data frame in CSV format had 26 columns of features. Thus we had to extract the most important features which had the most variance among others. By using the PCA approach mentioned in the previous section, the first 3 features have been extracted. Then by using *Python* and utilizing object-oriented programming, fog nodes have been created as objects. A total of 3 fog nodes have been considered. For each fog node and each episode of the learning process, resources have been initialized using calculated scores by Algorithm 1. By assigning fog node scores, the learning process starts. Totally 5000 episodes have been considered. For each episode, 200 steps have been considered.

4. 2. Experimental Results Our data has been captured from playing 15 minutes of 5 video games named Battlefield 4, Warthunder, Counter Strike Global Offensive (CS:GO), Forza horizon 4, Apex legends. Each tuple of the data contains 26 attributes. Each tuple indicates one second. All of the features have been normalized, transferred into the same scale between zero and one.

The results of the reinforcement learning for determining each fog node priority are stated in Table 2 categorized by each video game. For instance, to play apex Legends video game, fog node number 0 has the highest priority.

To evaluate the suggested fog nodes priorities, the frame rate per second and frame latency have been measured for three fog nodes and all of the five video games during 15 minutes. Each game was run for 15 minutes and the average of the results is demonstrated in Table 3. The fog node with a higher frame rate has a lower frame latency. As can be seen in Table 3, for the Apex legends game, fog node 0 has the lowest latency compared to the other fog nodes. For Battlefield 4 game, fog node 2 has the lowest latency and for CS: GO game, fog node 0 has the lowest latency. Fog node 1 has the lowest latency for the Forza Horizon 4 game and the Warthunder game, fog node 0 has the lowest latency.

TABLE 2. Fog node priority for each video game

	Priority #1	Priority #2	Priority #3
Apex Legends	Fog node #0	Fog node #1	Fog node #2
Battlefield 4	Fog node #2	Fog node #0	Fog node #1
CS: GO	Fog node #0	Fog node #1	Fog node #2
Forza horizon 4	Fog node #1	Fog node #2	Fog node #0
Warthunder	Fog node #0	Fog node #2	Fog node #1

TABLE 3. The average of FPS and frame latency of fog nodes for video games

Video Games	Fog nodes	Average FPS	Average frame latency (ms)
Apex legends	Fog node # 0	143	6.26
	Fog node # 1	139	6.31
	Fog node # 2	135	6.37
Battlefield 4	Fog node # 0	195	4.61
	Fog node # 1	166	5.36
	Fog node # 2	268	3.19
CS: GO	Fog node # 0	123	7.3
	Fog node # 1	55	23.8
	Fog node # 2	22	45.5
Forza horizon 4	Fog node # 0	120	7.1
	Fog node # 1	240	3.9
	Fog node # 2	184	5
Warthunder	Fog node # 0	290	2.8
	Fog node # 1	42	29.3
	Fog node # 2	129	6.7

Based on Table 3, the best fog nodes are selected for each video game and the corresponding results (FPS and latency) are compared to the approaches proposed by Zhang et al. [8] and Chen et al. [11]. Table 4 and Figure 3 demonstrate the results of comparing the average frame per second of the proposed method and the approaches proposed by Zhang et al. [8] and Chen et al. [11].

As can be seen, the proposed method has a higher average frame rate in all video games. This is because, in the proposed method, reinforcement learning is utilized to determine the best fog node for playing video games considering the frame rate. The objective of Zhang et al. [8] was to reduce bandwidth consumption and the objective of Chen et al. [11] was to maximize the long-term utility performance. After the proposed method, the EdgeGame has a better frame rate than the Darling approach.

Table 5 and Figure 4 demonstrate the result of comparing the average frame latency of the proposed method and the approaches proposed by Zhang et al. [8] and Chen et al. [11].

TABLE 4. The average of frame per second

	Apex Legends	Battlefield 4	CS: GO	Forza horizon 4	Warthunder
proposed method	143	268	123	240	290
EdgeGame [8]	118	255	115	200	188
Darling [11]	80	145	45	110	90

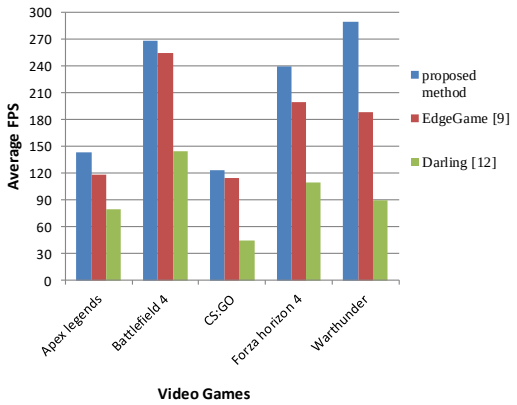


Figure 3. Comparing the average of frame per second

TABLE 5. The average frame latency

	Apex Legends	Battlefield 4	CS: GO	Forza horizon 4	Warthunder
proposed method	6.26	3.19	7.3	3.9	2.8
EdgeGame [8]	8.8	5	9	6.3	5.6
Darling [11]	14	8.1	13.4	12	11

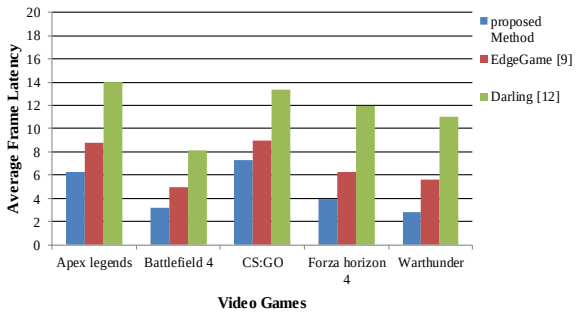


Figure 4. Comparing the average of frame latency

It can be observed that the average frame latency in the proposed method is less than two other approaches. This can be explained by the reason that the proposed method has q better frame rate and more frame rate leads to less frame latency. Among all video games, playing Battlefield 4 resulted in the least frame latency, and CS:GO resulted in the most frame latency.

5. CONCLUSION

Cloud gaming is a new paradigm that makes game players independent of having high-end hardware on their local computers. Since video games are a kind of latency-sensitive application, a latency reduction method based on reinforcement learning was proposed in this paper to appropriately select the fog node for running the video games on it with the lowest latency. We tried to apply PCA to reduce the 26 features of video games to 2 primary components and then extracted the most important features from them. The proposed method was implemented using Python and 5 video games named Battlefield 4, Warthunder, Counter Strike Global Offensive (CS:GO), Forza horizon 4, Apex legends were run for 15 minutes. Experimental results demonstrated that the proposed method compared to some existing methods reduced the frame latency and increase the frame rate of video games.

6. REFERENCES

1. Ross, P. E., "Cloud computing's killer app: Gaming," ***IEEE Spectrum***, Vol. 46, No. 3, (2009), 4-14, doi: 10.1109/MSPEC.2009.4795441.
2. Huang, C. Y., Hsu, C. H., Chang, Y.C., and Chen, K.T., "GamingAnywhere: an open cloud gaming system," in 4th ACM multimedia systems conference, Oslo, Norway (2013), 36-47, https://doi.org/10.1145/2537855.
3. Bonomi, F., Milito, R., Zhu, J., and Addepalli, S., "Fog computing and its role in the internet of things," first edition of the MCC workshop on Mobile cloud computing, New York, United States, (2012), 13-16, https://doi.org/10.1145/2342509.2342513.
4. Yaghmaee, F., Koohi, H., " Dynamic Obstacle Avoidance by Distributed Algorithm based on Reinforcement Learning," ***International Journal of Engineering, Transactions B: Applications***, Vol. 28, No. 2, (2015), 198-204, doi: 10.5829/idosi.ije.2015.28.02b.05.
5. rezaei, H., Motameni, H., Barzegar, B., " A Hidden Markov Model for Morphology of Compound Roles in Persian Text Part of Tagging," ***International Journal of Engineering, Transactions B: Applications***, Vol. 34, No. 11, (2021), 2494-2507, doi: 10.5829/IJE.2021.34.11B.12.
6. Mo, J., "Performance modeling of communication networks with Markov chains," Synthesis Lectures on Data Management, Vol. 3, No. 1, (2010), 1-90, doi: 10.2200/S00269ED1V01Y201004CNT005.
7. Talaat, F.M., Saraya, M.S., Saleh, A.I., Ali, H. A., and Ali, S.H., "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment", ***Journal of Ambient Intelligence and Humanized Computing*** (2020), 1-16, https://doi.org/10.1007/s12652-020-01768-8.
8. Zhang, X., Chen, H., Zhao, Y., Ma, Z., Xu, Y., Huang, H., and Wu, D. O. "Improving cloud gaming experience through mobile edge computing", ***IEEE Wireless Communications***, Vol. 26, No. 4, (2019), 178-183, doi: 10.1109/MWC.2019.1800440.
9. Zhang, C., and Zheng, Z., "Task migration for mobile edge computing using deep reinforcement learning," ***Future Generation Computer Systems***, Vol. 96, (2019), 111-118, https://doi.org/10.1016/j.future.2019.01.059.
10. Chen, H., Zhang, X., Xu, Y., Ren, J., Fan, J., Ma, Z. and Zhang, W., "T-Gaming: A Cost-Efficient Cloud Gaming System at Scale", ***IEEE Transactions on Parallel and Distributed Systems***, Vol. 30, No. 12, (2019), 2849-2865, doi: 10.1109/TPDS.2019.2922205.

11. Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y. and Bennis, M., "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," ***IEEE Internet of Things Journal***, Vol. 6, No. 3, (2018), 4005-4018, doi: 10.1109/JIOT.2018.2876279.
12. Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N. and Truck, I., "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," The Seventh International Conference on Autonomic and Autonomous Systems, Venice, Italy, (2011), 67-74, ISBN: 978-1-61208-134-2.
13. Wang, Y., Wang, K., Huang, H., Miyazaki, T. and Guo, S., "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," ***IEEE Transactions on Industrial Informatics***, Vol. 15, No. 2, (2018), 976-986, doi: 10.1109/TII.2018.2883991.
14. Dinh, T.Q., La, Q.D., Quek, T.Q. and Shin, H., "Learning for Computation Offloading in Mobile Edge Computing," ***IEEE Transactions on Communications***, Vol. 66, No. 12, (2018), 6353-6367, doi: 10.1109/TCOMM.2018.2866572.
15. Huang, L., Bi, S. and Zhang, Y.J.A., "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," ***IEEE Transactions on Mobile Computing***, Vol. 19, No. 11, (2019), 2581-2593, doi: 10.1109/TMC.2019.2928811.

Persian Abstract

چکیده

بر خلاف بازی های قدیمی که یک بازی روی یک دستگاه کاربر بصورت محلی اجرا می شد، در بازی های ابری یک بازی ویدئویی برخط روی سرورهای راه دور ابری اجرا می شوند و نتایج پردازش ها بصورت مستقیم به دستگاه کاربر ارسال می‌شوند. این باعث می‌شود که بازیکنان از داشتن منابع سخت افزاری با قابلیت بالا در کامپیوتر های محلی خود بی نیاز شوند. از آنجا که بازی‌های ویدئویی نوعی از برنامه های حساس به تاخیر هستند، سرورهای ابری که دور از کاربران قرار گرفته اند مناسب نیستند. در محاسبات مه، گره های مه در مجاورت کاربران قرار گرفته اند و قادرند تاخیر را کاهش دهند. در این مقاله، روشی مبتنی بر یادگیری تقویتی جهت کاهش تاخیر ارائه شده است تا مشخص کند کدام گره محاسباتی مه می تواند بازی های ویدئویی را با کمترین تاخیر اجرا کند. همچنین در روش پیشنهادی، یک روش مبتنی بر تحلیل مولفه اصلی (PCA) استفاده شده تا مهمترین ویژگی های هر بازی ویدئویی را به عنوان ورودی فرآیند یادگیری استخراج کند. روش پیشنهادی توسط نرم افزار پایتون پیاده سازی شد. نتایج آزمایشات نشان می دهد روش پیشنهادی در مقایسه با برخی روش‌های موجود می تواند برای بازی های ویدئویی تاخیر فریم را کاهش داده و نرخ فریم را افزایش دهد.