Accelerate Personalized IoT Service Provision by Cloud-Aided Edge Reinforcement Learning: A Case Study on Smart Lighting

Jun Na(✉), Handuo Zhang, Xin Deng, Bin Zhang(✉), and Ziyi Ye

Software College, Northeastern University, Shenyang, China
{najun,zhangbin}@mail.neu.edu.cn

Abstract. To enhance the intelligence of IoT devices, offloading sufficient learning and inferencing down to the edge environment is promising. However, there are two main challenges for applying the cloud generated model in the edge environment. On the one hand, the input may vary on dimensions or cover different situations that the cloud hasn't met. On the other hand, the model's output might not satisfy the given user's personalized preference. To make full use of the cloud generated model in the edge environment for accelerating personalized service provision, we propose cloud-aided edge learning. Unlike current federated learning and transfer learning, we focus on knowledge fusion in edge decision making and try to build the supplement/correction model. We take the personalized service provision in a smart lighting system as an example, design and implement the related deep reinforcement learning model, and take experiments based on the data generated on the open software DAILux to show our approach's effectiveness and performance.

Keywords: Edge intelligence · Edge-cloud collaborated learning · Personalized service provision · Smart lighting · Deep Reinforcement Learning (DRL)

1 Introduction

The Internet of Things (IoT) [3,20] enables all kinds of real-world objects (including human beings) to be connected to the cyber world. Considering the characteristics of human-in-the-loop, providing personalized IoT services efficiently and transparently turns to be essential. Recently, applying machine learning to speed up personalization becomes a promising way[22,36], which can extract useful knowledge from interactions happening in the physical world to produce proper reactions.

To process the continuously generated IoT data efficiently, it needs a powerful data center with enough storage and computing resources. Although cloud computing is an excellent platform to handle the enormous IoT data, pushing all the raw data to the cloud is inefficient in response latency, network

© Springer Nature Switzerland AG 2020
E. Kafeza et al. (Eds.): ICSOC 2020, LNCS 12571, pp. 69–84, 2020.
https://doi.org/10.1007/978-3-030-65310-1_6

bandwidth cost, and possible privacy concerns [8,23]. To solve these problems, edge computing [28], also known as fog computing [4], is becoming the right solution and get more attention in both research and industry domain. By offloading sufficient training and inferencing down to the edge environment, edge intelligence would be enhanced to satisfy users' personalized needs more efficiently while protecting privacy [19,27,32,37]. Combining both cloud computing and edge computing advantages to offer flexible edge-cloud collaboration gets more attention [5,27,36].

Existing studies usually focus on the underlying mechanisms of edge-cloud collaboration. However, there are more challenges to accelerate personalized service provision through deep learning. For example, data achieved by the edge node might be different from the generic dataset used to generate the global model. It does not only refer to the differences in input dimensions but also other situations occurring in the edge environment. Besides, different preferences among edge nodes may cause conflicts during knowledge fusing [17,24].

To solve the above problems, we focus on reducing the edge computation cost as much as possible by making full use of the global model and only learn to deal with the inapplicable parts. Since the successful application of Deep Reinforcement Learning (DRL) [6,9,16,30] in playing Atari and Go games, we adopt DRL to realize efficient online learning. Taking the example of offering comfortable, personalized illumination in a smart lighting system, we designed and implemented the corresponding algorithms, generated data based on an open software platform, DIALux, and tested our approaches' effect. The main contributions of this paper are as follows.

- We propose a cloud-aided edge reinforcement learning framework that supports downloading the global consensus model from the cloud center and fuses it into the edge learning process.
- To enhance the efficiency and effect by applying the downloaded pre-trained model, we put forward two integration strategies, i.e., input expansion strategy and output correction strategy.
- We conduct a case study on smart lighting as an example and present the proposed approach's effects.

2 Background and Related Work

2.1 Edge-Cloud Collaboration

Among most current studies, virtualizing the resources and services over WAN networks is the shared premises to combine cloud computing and edge computing. Researches such as Pcloud [11], CoTware [1], FocusStack [2], etc. emphasize to virtualize resources of individual devices, edge nodes and cloud to build a distributed resource pool for supporting resource-limited front-end devices. While, SpanEdge [26], CloudPath [21], ECHO [25], etc. focus on the data stream processing across different layers seamlessly.

These works establish an elastic data analysis environment. However, most of them pay attention to leveraging resources on higher layers along the path from front-end devices to cloud with fewer considerations on how these analysis results will reflect behaviors provided on the front layers. Moreover, as presented in several works [28,29,34], most existing studies lack information sharing among multiple stakeholders, while the sharing may help these edge nodes to make smarter decisions. Thus, it is still challenging to support more diversified, personalized, and delay-sensitive system behaviors in the edge environment.

2.2 Schemes for Edge-Cloud Collaborated Model Training

At present, research on improving computing power and effects based on edge-cloud collaboration is still in its early stages [34]. There are three primary schemes for edge-cloud collaborated training models.

1) Gradient sharing: Reduce the transmission size of a single model by compressing the gradient, so that the model update results are transmitted frequently and multiple times to make up for the lack of computing power of edge nodes [10,12]. The training effect in the network is independent of the same and distributed data. As a result, the sharing effect of multi-edges in heterogeneous networks with different data sets cannot be guaranteed.
2) Parameter sharing: The edge side conducts preliminary training of the model and transmits the parameters to the parameter server. The parameter aggregation method in the cloud improves the accuracy of the edge side model [15,18]. This scheme can reduce the transmission volume. It also protects the privacy and improves model accuracy, but in scenarios with high personalization requirements, parameter aggregation still has challenges.
3) Data sharing: When it is necessary to collect the original data and perform parameter aggregation or train directly on the parameter server, noise can be added to the data on the edge side or privacy leakage can be reduced by preprocessing [35]. Simultaneously, there are some methods to study how to enhance the processing capabilities of edge nodes through algorithms or model hardware [14,31].

Existing work focuses more on improving the efficiency of data analysis and model training and protecting privacy. However, the issue of how to improve the personalized intelligence at the edge through the edge-cloud collaboration still needs further studies.

2.3 Fast Personalization by Federated Reinforcement Learning

Personalization aims to understand user behavior and adapting to it, which is crucial for gaming, personal assistants, dialogue managers, etc. It is often time consuming, so a critical challenge of personalization is how to adapt to a new situation quickly. To make robots quickly adapt to the new environment by sharing their experiences, Liu *et al.* [17] proposed the Lifelong Federated Reinforcement

Learning (LFRL) architecture. Each robot learns to avoid some new types of obstacles in the new environment through reinforcement learning. After obtaining a private Q-network model, the robot will upgrade this model by fusing with models submitted by other robots through federated learning. This work assumed all agents make the same decision when they meet the same type of obstacle. However, different agents might prefer to make other decisions to adapt to their current behavior. Zhou *et al.* [38] proposed a similar federated reinforcement learning framework by building a Multilayer Perception (MLP) to compute a global Q-network output with all Q-network results. It also doesn't consider to enable agents to make personalized decisions.

Nadiger *et al.* [22] pay attention to personalization in the context of gaming. They propose an overall architecture, including the grouping policy, the learning policy, and the federation policy. By putting forward the grouping policy, this approach can avoid the risk of adding irrelevant samples, which may increase the personalization time while guaranteeing the model quality. Unlike the above works, this approach solves the problem resulting from conflict samples by only allowing similar agents to share data samples. However, as reinforcement learning is a typical online learning algorithm, considering the latency of generating a new shared model, directly updating the private model weights might overwrite some new knowledge learned during the shared model updating.

To solve these problems, we propose cloud-aided edge learning to fuse shared knowledge gained at the cloud to the edge. Unlike existing studies, we try to avoid training the whole shared model by only focusing on different situations to reduce edge computation as much as possible.

3 System Model for Cloud-Aided Edge Reinforcement Learning

3.1 Basic Ideas of the Cloud-Aided Edge Reinforcement Learning

To provide satisfactory personalized services, it requires capturing users' personalized explicit or implicit requirements by self-learning. Besides, considering the influences from the external environment and the users' changing preferences, the system should be able to adapt to these new situations to provide better services. We propose a hybrid framework focusing on how to realize and improve the self-learning and adaptive ability of an edge system. Figure 1 shows the proposed edge-cloud collaborative framework.

We focus on two key aspects to achieve smarter automation, learning, and adaptation.

1) How to share knowledge among different edge nodes with cloud assistance: Single edge environments always face the data sparsity problem. For example, lack of various states of weather, season, and system deployment. It is necessary to share knowledge among different edge systems, which will enable an edge system to make more smart decisions by taking advantage of the situations shared by others that haven't already appeared but might happen in

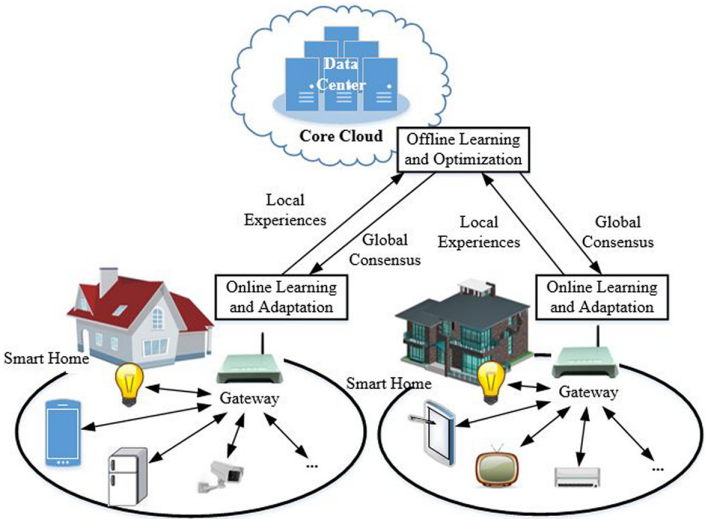


Fig. 1. The overview of collaboration for learning and adaptation in smart home systems between the edge node and the cloud.

the future. The offline learning part in Fig. 1 is in charge of sharing knowledge among different edge nodes by parameter sharing.

2) How to utilize the historical experiences/knowledge generated on cloud in making real-time decisions on edge nodes efficiently: Learning performed on cloud is based on the historical data. Therefore, the resulting knowledge usually reflects past situations that may be outdated in current states. An appropriate mechanism is needed to integrate such historical experiences with fast rules of the local environment to improve the accuracy of reactions generated by an edge decision-maker. The online learning part in Fig. 1 aims to enhance real-time decision making by applying consensus achieved through offline learning on the cloud.

3.2 Knowledge Fusion Strategies on Edge Nodes

As mentioned above, data achieved by the edge node might be different from the generic dataset used to generate the pre-trained model, including differences in either input dimensions or situations occurring in the edge environment. Besides, different preferences among edge nodes may also be different from each other. To cope with such various problems, we propose two knowledge fusion strategies to accelerate edge personalized decision making, i.e., input expansion and output correction.

The Input Expansion Strategy. As the global model and the local model are trained based on different data samples, there may be some special states

emerging in the edge environment that have not met by the cloud. In this case, the global consensus model only captures part of the knowledge about the edge environment. To complement the model to provide more accurate decisions, we propose the input expansion strategy shown in Fig. 2. As shown in Fig. 2, the current state will be sent to both the global consensus model and the local private model as input. Then, the decision-maker will produce the final action by integrating outputs of both the two models. Such fusing can be realized as follows.

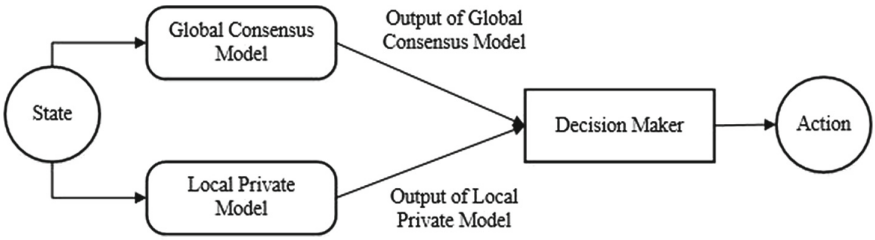


Fig. 2. The input expansion strategy.

$$H(x) = \sum_{i=1}^{N} w_i * h_i(x) \tag{1}$$

where, w_i is the weight of the output $h_i(x)$, and N is the total number of models participate in fusing.

Under the edge-cloud collaboration framework, both the global and the local model might have some information not learned by the counterpart model. So the setting of weights needs to balance the advantages of both parties. Assuming that the accuracy of both parties is the same, the weights of the two are the same, and the advantages of both parties can be guaranteed to be balanced. While, if they have different degrees of accuracy, it is necessary to ensure that the model with higher accuracy has a higher weight. For reinforcement-learning, we define accuracy as the proportion of decisions resulted in a reward greater than zero in all decisions. With this in mind, we define the following equation to compute the weight w_i based on the corresponding accuracy Acc_i.

$$w_i = \frac{Acc_i}{\sum_{i=1}^{N} Acc_i} \tag{2}$$

Considering the simplest situation that there are only one global model and one local model, the value of N is 2.

On this basis, suppose the accuracy of the private model is Acc_{edge}, and the accuracy of the global model is Acc_{cloud}. The corresponding model output value formula is

$$H(x) = \frac{Acc_{edge}}{Acc_{edge} + Acc_{cloud}} * h_{edge}(x) + \frac{Acc_{cloud}}{Acc_{edge} + Acc_{cloud}} * h_{cloud}(x) \tag{3}$$

Initially, the edge model has just started training and is still in the process of exploration. At this time, the accuracy of the local model should be set to 0 while the weight of the cloud model should be set to 1. After training for a while, as the accuracy of the edge model improves, its weight, i.e. the value of $Acc_{edge}/(Acc_{edge} + Acc_{cloud})$ will gradually increase.

The Output Correction Strategy. Because the global consensus model is achieved by integrating, it contains user consensus with similar characteristics. However, when the model is delivered to a given edge environment, it may not be able to meet the preference of a specific user. To satisfy users' personalized usage habits and requirements, we need to modify the output of the global model properly. To this end, we take the output of the global model as an additional input in training the local personalized model, as shown in Fig. 3.

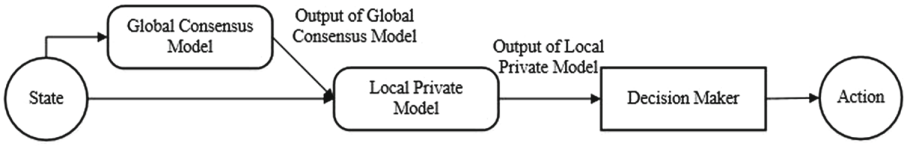


Fig. 3. The output correction strategy.

The corresponding algorithm is shown in Algorithm 1, which both accelerates the training of the local model but also improves the effectiveness of the decision making. Here, we only consider revising the final decision generated by the global model.

4 Reinforcement Learning for Providing Personalized Illuminance in Smart Lighting

Lighting plays a significant role in our daily lives. Generally, lighting includes the use of both natural illumination in the form of daylight and electric illumination provided by various light sources. Together with the flourishing of IoT, a new generation of LED lighting systems are emerging, i.e., LED-based intelligent lighting systems where LEDs are integrated with sensors and actuators to have intelligence. For example, Philips Hue is a wireless lighting product, which can cooperate with a range of smart devices such as Amazon Echo, Apple HomeKit, Google Home, etc. to provide a convenient and comfortable way for occupants to control and experience light.

Algorithm 1. Collaborative algorithm by adjusting cloud model's output.

Require:
The environment state, $S =< Bright1, Bright2, Distance, Time >$;
User's operation on the light, $I_{control}$
The globel model, $Model1$
Ensure:
The adjustment action to the light, $Action$;
Set the training episode to n;
for $i = 1$ to n **do**
  Achieve the initial state S;
  Set the max adjustment time to m;
  for $j = 1$ to m **do**
    Input S to $Model1$ and get the output $o1$;
    Combine S and $o1$ to a new state S^{-};
    Input the state S^{-} into the local model $Model2$. Train the model and get the output $Action$;
    Perform the generated $Action$;
    Achieve the next state $S^{'}$;
    Achieve use's operation $I_{control}$;
    Compute the $Reward$ based on $I_{control}$;
    Perform related iterative formula or loss function to optimize $Model2$.
    Update the state to the next iteration, $S = S^{'}$;
  end for
end for

To enhance the quality of user experience, light control strategies need to be more flexible and automatic. Thus, AI and data-mining technologies are widely adopted to seek useful information on resident behavior and the state of the environment for generating satisfactory reactions [7,13,24,33]. These approaches are usually storing and analyzing the continuous human-system interactions during the non-stop system running. Considering the successful application of DRL, we adopt DRL to realize personalized illuminance setting.

According to the definition of reinforcement learning, we use a quadruple $< S, A, P, R >$ to represent a reinforcement learning model, where S represents an environmental state, A represents an action, P represents a state transition probability, and R represents a reward value.

In reinforcement learning, the state comes from the agent's observation of the environment. We suppose there are four sensors around a light, which are two light sensors, one ultrasonic sensor, and one infrared sensor. Generally, the infrared sensor is usually used to determine whether there is a person or not to turn on or turn off the light. Thus, we only use the other three sensors to construct the current state. Specifically, we define the state as a 4-tuple $< B_feeling, B_nature, Distance, Time >$. We get the synthesized brightness (i.e., B_light) and natural light (i.e., B_nature) by the two light sensors. The distance data (i.e., $Distance$) is obtained by the ultrasonic sensor and the $Time$ is when constructing the state values.

For an LED, the action represents the adjustment of the lamp output by the model. For simplicity, we only consider the brightness in this paper. There should be two kinds of actions. One is a determined value of brightness or a predefined gear. The other is one of the operations up, down, and hold.

Reinforcement learning needs to construct reward functions for training the model. To achieve higher user satisfaction, we define the reward function as follows.

$$r = \alpha * R_{positive} + \sum \gamma^i * R_{positive} + I_{control} * R_{negative} \tag{4}$$

When the user moved to another place or the sunlight intensity changed, the algorithm should generate a proper illuminance and set the light accordingly. Whether the user adjusts the light manually after running the automated setting is used as the feedback for training a DQN. In the above equation, $R_{negative}$ is the negative feedback, which is collected if the user adjusts the light manually after an automated adaptation. In other words, the algorithm didn't find a satisfactory brightness for the user. Similarly, if the user didn't take any action after an automated adaptation, it means the algorithm meets users expect. α is the times that there is no user adjustment during an episode. γ is the reward decay rate to decrease the reward if there is no manual adjustment. i is the continuous times without manual adjustment in an episode, and $I_{control}$ is the number of manually adaptions.

5 Experiments

5.1 Dataset

We use the open software DIALux to generate a dataset for simulating the training and decision making procedure. DIALux is a lighting design software, which is a useful lighting calculation software. It can use all the lamps and lanterns provided by the lamp manufacturers and add sunlight to the scene according to actual calculation requirements. We set a 5.4 m * 3.6 m room in DIALux with a window, a variable power lamp placed on a table in the center of the room, as shown in Fig. 4. The light is 1.8 m away from the window and 0.85 m away from the ground. Taking the height of 0.85 m above the ground as the daily working plane, people can obtain the light intensity of each point on the working plane under different power under the influence of the current sunlight.

When we set the present time is 8 am, Fig. 5 shows the brightness in the room. The red point is the position of the light, and the blue line at the bottom is the window's position. We can find that light intensity around the lamp is about 300 lux, while the light intensity near the window is about 1367 lux. Different conditions of the room can be obtained by adjusting the power of the table lamp and the sunlight. Based on this basic dataset, we generate sequences to simulate interaction procedures between different users and lamps under various environments.

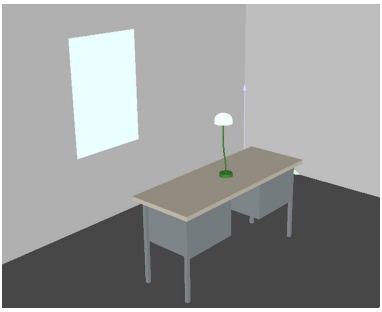


Fig. 4. Data collection environment setting in DAILux.

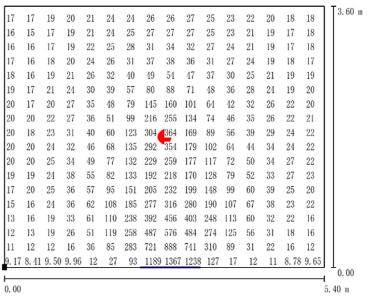


Fig. 5. Brightness value in the room. (Color figure online)

5.2 Experimental Setup

To verify the effectiveness and performance of the proposed approach, we build a three-layer neural network, an input layer, a fully connected layer and an output layer, in which four neuron activation functions are set to Relu in the input and the fully connected layer. Three neurons are set in the output layer, and the activation function is linear. The distance, current sunlight intensity, and current table light intensity are used as input of the neural network. The three output values represent increasing the lamp power, decreasing the lamp power, and keeping the power unchanged. The lamp power is adjusted through the decision output of the neural network.

We ran the experiments on a PC with an Intel Core i7-7700HQ and 16G RAM. DQN, the algorithm of the input expansion strategy, and the algorithm of output correction strategy were used for experiments. Each algorithm trained 50 episodes, and each episode carried out 600 network interactions with humans. It is known that people work in comfortable environments with an illumination of about 300 lux. If the comprehensive illumination near the lamp does not reach 290 lux or more than 320 lux, the network will receive a negative reward of -1. Otherwise, it will receive a positive reward of 1. Adjust the network through the reward value obtained, and the upper limit of the positive reward obtained is 600. Initially, we apply the same network to train both the global model and the local private model.

5.3 Experimental Results

Lab1: Comparison on Working in the Same Environment. First, we compared the rewards of the proposed two strategies and a pure DQN model. In this experiment, we train a DQN model as the global model and then fuse it with a new private model which is start training from scratch. The results are shown in Fig. 6.

We can find that using these three algorithms all quickly obtain a higher reward value. However, the reward value obtained by using pure DQN and training 50 episodes alone does not exceed 400. It means there are still some wrong decisions in these episodes, which result in a low reward. To get a higher reward, we need more episodes to train the model. On the contrary, both the input expansion strategy and output correction strategy can get a high reward in a short episode. The output correction strategy is better than the input expansion strategy from the perspective of speed and stability. It's because the global model is achieved in the same environment as the private model. Thus, the global model output can reach high rewards in most of the cases in the edge environment. However, as initialization of the local model in the input expansion strategy might bring more influences.

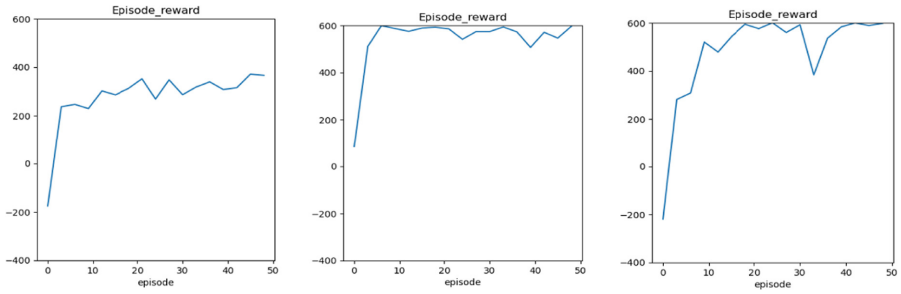
Lab2: Comparison on Satisfying Different Preferences. To compare the performance of the two proposed knowledge fusion strategies, we first train the global model with a target brightness between 290–320lux. At the same time, the user in the local environment prefers the illumination between 350–380lux. The results are shown in Fig. 7. From the results, we can see both strategies can get a high reward quickly compared with using pure DQN, as shown in Fig. 6, even though the global model is trained to get a different target brightness.

Lab3: Comparison on Training in Different Environments. To test the performance for fusing models trained in a different environment, we train a global model in the environment of around 8 am. Then, we try to fuse this model by the proposed two fusion strategies to adapt to the environment of 8 pm. Figure 8 shows the accordingly results. It can be seen from the experimental results that using the global model to perform auxiliary training on edge, both the two strategies can achieve perfect results. In the output correction strategy, the edge node needs to be adjusted briefly to adapt to the current night environment. However, the input expansion strategy can get a higher reward in the initial states.

5.4 Quantitative Comparison of the Performance of Different Algorithms

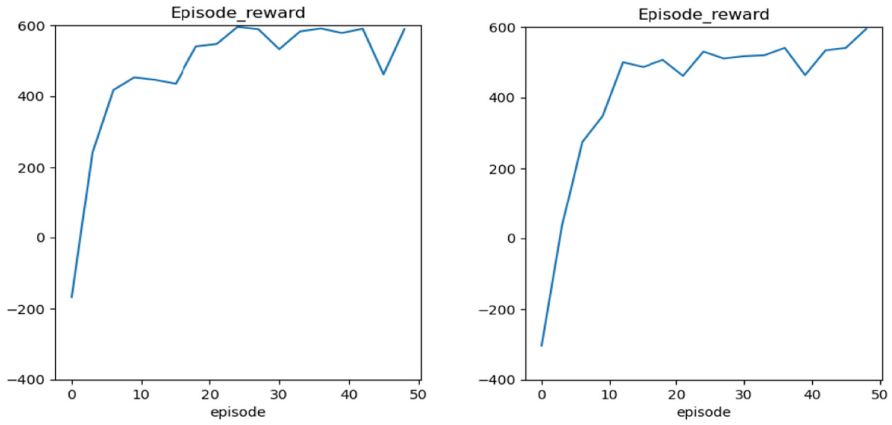
We set the condition that if the algorithm gets a reward which greater than 450 within five consecutive episodes, it is stable enough to adapt to the environment. Then, by running the above experiments, we collect the time cost, average training time, and memory size, as shown in Table 1.

We can clearly find that the average iterations of only using DQN, which is nearly ten times using either of the proposed strategies. In the latter two groups of experiments, we can see the output correction strategy requires less time than the input expansion strategy. While, from the perspective of memory occupation, our strategies need a little more memory than only using DQN as we need to load the global model.



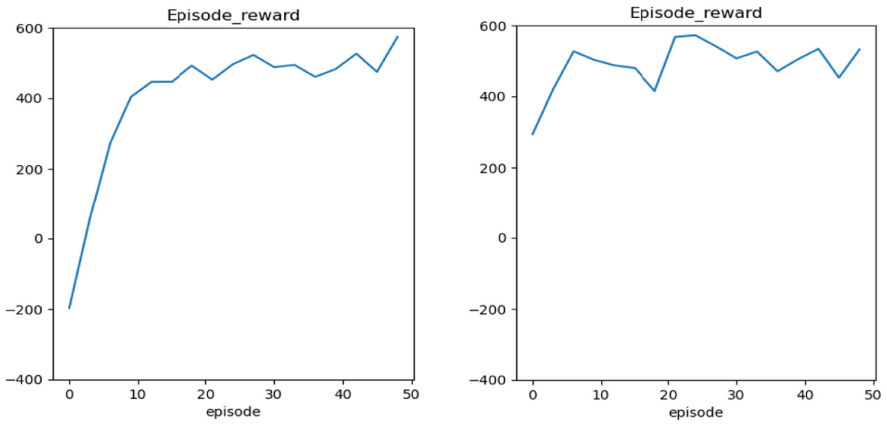
(a) Pure DQN (b) Output Correction (c) Input Expansion

Fig. 6. Rewards comparison for applying models trained in the same environment.



(a) Output Correction (b) Input Expansion

Fig. 7. Rewards comparison for applying models trained for satisfying different preferences.



(a) Output Correction (b) Input Expansion

Fig. 8. Rewards comparison for applying models trained in different environments.

Table 1. Quantitative comparison of performance of different algorithms.

Experiment	Strategy	Average iterations	Time for training	Memory occupation
Lab 1	DQN	186	6 m10 s	0.2281 GB
	Output correction	10	28 s	0.2301 GB
	Input expansion	24	1 m19 s	0.2296 GB
Lab 2	Output correction	19	56 s	0.2303 GB
	Input expansion	24	1 m13 s	0.2303 GB
Lab 2	Output correction	14	41 s	0.2310 GB
	Input expansion	14	52 s	0.2307 GB

6 Conclusion

In this paper, we propose a cloud-aided edge reinforcement learning framework and introduce two edge knowledge fusion strategies. As shown in the experiments, the proposed approach can accelerate personalized service provision while do not increase the memory occupation obviously. We present a case study on applying the method in providing personalized illuminance services. The proposed framework and strategies are also suitable for other applications. In our future work, we will focus on identifying and describing the features of different edge environments, which would better enhance the inference accuracy.

References

1. Al-Jaroodi, J., Mohamed, N., Jawhar, I., Mahmoud, S.: CoTWare: a cloud of things middleware. In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 214–219. IEEE (2017)
2. Amento, B., Balasubramanian, B., Hall, R.J., Joshi, K., Jung, G., Purdy, K.H.: FocusStack: orchestrating edge clouds using location-based focus of attention. In: 2016 IEEE/ACM Symposium on Edge Computing (SEC), pp. 179–191. IEEE (2016)
3. Atzori, L., Iera, A., Morabito, G.: Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. Ad Hoc Netw. **56**, 122–140 (2017)
4. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog computing: a platform for Internet of Things and analytics. In: Bessis, N., Dobre, C. (eds.) Big Data and Internet of Things: A Roadmap for Smart Environments. SCI, vol. 546, pp. 169–186. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05029-4_7
5. Chen, J., Ran, X.: Deep learning with edge computing: a review. Proc. IEEE **107**(8), 1655–1674 (2019)
6. Cheng, Z., Zhao, Q., Wang, F., Jiang, Y., Xia, L., Ding, J.: Satisfaction based q-learning for integrated lighting and blind control. Energy Build. **127**, 43–55 (2016)
7. Cheuque, C., Baeza, F., Marquez, G., Calderon, J.: Towards to responsive web services for smart home led control with Raspberry Pi: a first approach. In: 2015 34th International Conference of the Chilean Computer Science Society (SCCC), pp. 1–4. IEEE (2015)
8. Corcoran, P., Datta, S.K.: Mobile-edge computing and the Internet of Things for consumers: extending cloud computing and services to the edge of the network. IEEE Consum. Electron. Mag. **5**(4), 73–74 (2016)
9. Fu, Q., Hu, L., Wu, H., Hu, F., Hu, W., Chen, J.: A Sarsa-based adaptive controller for building energy conservation. J. Comput. Methods Sci. Eng. **18**(2), 329–338 (2018)
10. Hardy, C., Le Merrer, E., Sericola, B.: Distributed deep learning on edge-devices: feasibility via adaptive compression. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), pp. 1–8. IEEE (2017)
11. Jang, M., Schwan, K., Bhardwaj, K., Gavrilovska, A., Avasthi, A.: Personal clouds: sharing and integrating networked resources to enhance end user experiences. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 2220–2228. IEEE (2014)
12. Kairouz, P., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
13. Kandasamy, N.K., Karunagaran, G., Spanos, C., Tseng, K.J., Soong, B.H.: Smart lighting system using ANN-IMC for personalized lighting control and daylight harvesting. Build. Environ. **139**, 170–180 (2018)
14. Lee, J., Tang, H., Park, J.: Energy efficient canny edge detector for advanced mobile vision applications. IEEE Trans. Circ. Syst. Video Technol. **28**(4), 1037–1046 (2016)
15. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Netw. **32**(1), 96–101 (2018)
16. Li, H., Wei, T., Ren, A., Zhu, Q., Wang, Y.: Deep reinforcement learning: framework, applications, and embedded implementations. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 847–854. IEEE (2017)

17. Liu, B., Wang, L., Liu, M.: Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. IEEE Robot. Autom. Lett. **4**(4), 4555–4562 (2019)
18. Lu, S., Yao, Y., Shi, W.: Collaborative learning on the edges: a case study on connected vehicles. In: 2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19) (2019)
19. Mahdavinejad, M.S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., Sheth, A.P.: Machine learning for Internet of Things data analysis: a survey. Digit. Commun. Netw. **4**(3), 161–175 (2018)
20. Mattern, F., Floerkemeier, C.: From the Internet of computers to the Internet of Things. In: Sachs, K., Petrov, I., Guerrero, P. (eds.) From Active Data Management to Event-Based Systems and More. LNCS, vol. 6462, pp. 242–259. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17226-7_15
21. Mortazavi, S.H., Salehe, M., Gomes, C.S., Phillips, C., de Lara, E.: Cloudpath: a multi-tier cloud computing framework. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–13 (2017)
22. Nadiger, C., Kumar, A., Abdelhak, S.: Federated reinforcement learning for fast personalization. In: 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), pp. 123–127. IEEE (2019)
23. Osia, S.A., et al.: A hybrid deep learning architecture for privacy-preserving mobile analytics. IEEE Internet Things J. **7**, 4505–4518 (2020)
24. Paulauskaite-Taraseviciene, A., Morkevicius, N., Janaviciute, A., Liutkevicius, A., Vrubliauskas, A., Kazanavicius, E.: The usage of artificial neural networks for intelligent lighting control based on resident's behavioural pattern. Elektronika ir Elektrotechnika **21**(2), 72–79 (2015)
25. Ravindra, P., Khochare, A., Reddy, S.P., Sharma, S., Varshney, P., Simmhan, Y.: ECHO: an adaptive orchestration Platform for hybrid Dataflows across cloud and edge. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 395–410. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_28
26. Sajjad, H.P., Danniswara, K., Al-Shishtawy, A., Vlassov, V.: SpanEdge: towards unifying stream processing over central and near-the-edge data centers. In: 2016 IEEE/ACM Symposium on Edge Computing (SEC), pp. 168–178. IEEE (2016)
27. Samie, F., Bauer, L., Henkel, J.: From cloud down to things: an overview of machine learning in Internet of Things. IEEE Internet Things J. **6**(3), 4921–4934 (2019)
28. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
29. Stojkoska, B.L.R., Trivodaliev, K.V.: A review of Internet of Things for smart home: challenges and solutions. J. Clean. Prod. **140**, 1454–1464 (2017)
30. Wei, T., Wang, Y., Zhu, Q.: Deep reinforcement learning for building HVAC control. In: Proceedings of the 54th Annual Design Automation Conference 2017, pp. 1–6 (2017)
31. Xu, Q., Varadarajan, S., Chakrabarti, C., Karam, L.J.: A distributed canny edge detector: algorithm and FPGA implementation. IEEE Trans. Image Process. **23**(7), 2944–2960 (2014)
32. Yazici, M.T., Basurra, S., Gaber, M.M.: Edge machine learning: enabling smart Internet of Things applications. Big Data Cogn. Comput. **2**(3), 26 (2018)
33. Yu, T., Kuki, Y., Matsushita, G., Maehara, D., Sampei, S., Sakaguchi, K.: Design and implementation of lighting control system using battery-less wireless human detection sensor networks. IEICE Trans. Commun. **E100-B**(6), 974–985 (2016)

34. Zhang, Q., Zhang, Q., Shi, W., Zhong, H.: Distributed collaborative execution on the edges and its application to amber alerts. IEEE Internet Things J. **5**(5), 3580–3593 (2018)
35. Zhang, T., He, Z., Lee, R.B.: Privacy-preserving machine learning through data obfuscation. arXiv preprint arXiv:1807.01860 (2018)
36. Zhang, X., Qiao, M., Liu, L., Xu, Y., Shi, W.: Collaborative cloud-edge computation for personalized driving behavior modeling. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, pp. 209–221 (2019)
37. Zhang, Y., Ma, X., Zhang, J., Hossain, M.S., Muhammad, G., Amin, S.U.: Edge intelligence in the cognitive Internet of Things: improving sensitivity and interactivity. IEEE Netw. **33**(3), 58–64 (2019)
38. Zhuo, H.H., Feng, W., Xu, Q., Yang, Q., Lin, Y.: Federated reinforcement learning. arXiv:1901.08277 (2019)