Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems

Boyi Liu[1,3], Lujia Wang[1] and Ming Liu[2]

Abstract— **This paper was motivated by the problem of how to make robots fuse and transfer their experience so that they can effectively use prior knowledge and quickly adapt to new environments. To address the problem, we present a learning architecture for navigation in cloud robotic systems: Lifelong Federated Reinforcement Learning (LFRL). In the work, we propose a knowledge fusion algorithm for upgrading a shared model deployed on the cloud. Then, effective transfer learning methods in LFRL are introduced. LFRL is consistent with human cognitive science and fits well in cloud robotic systems. Experiments show that LFRL greatly improves the efficiency of reinforcement learning for robot navigation. The cloud robotic system deployment also shows that LFRL is capable of fusing prior knowledge. In addition, we release a cloud robotic navigation-learning website to provide the service based on LFRL: *www.shared-robotics.com*.**

I. INTRODUCTION

Autonomous navigation is one of the core issues in mobile robotics. It is raised among various techniques of avoiding obstacles and reaching target position for mobile robotic navigation. Recently, reinforcement learning (RL) algorithms are widely used to tackle the task of navigation. RL is a kind of reactive navigation method, which is an important meaning to improve the real-time performance and adaptability of mobile robots in unknown environments. Nevertheless, there still exists a number of problems in the application of reinforcement learning in navigation such as reducing training time, storing data over long time, separating from computation, adapting rapidly to new environments etc [1].

In this paper, we address the problem of how to make robots learn efficiently in a new environment and extend their experience so that they can effectively use prior knowledge. We focus on cloud computing and cloud robotic technologies [2], which can enhance robotic systems by facilitating the process of sharing trajectories, control policies and outcomes of collective robot learning. Inspired by human congnitive science present in Fig.1, we propose a **L**ifelong **F**ederated **R**einforcement **L**earning (LFRL) architecture to realize the goal. With the scalable architecture and knowledge fusion algorithm, LFRL achieves exceptionally efficiency in

[1]Boyi liu, Lujia Wang are with Cloud Computing Lab of Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences.liuboyi17@mails.ucas.edu.cn; lj.wang1@siat.ac.cn; cz.xu@siat.ac.cn

[2]Ming liu, is with Department of ECE, Hong Kong University of Science and Technology. eelium@ust.hk

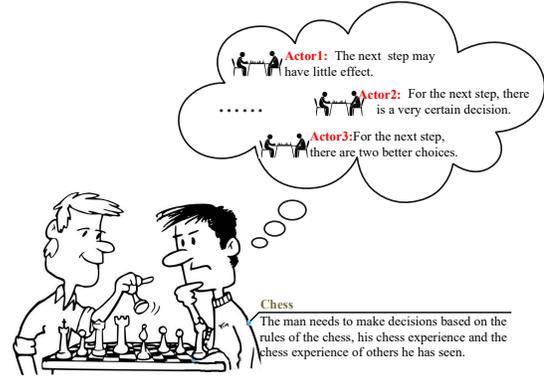[3]Boyi liu is also with the University of Chinese Academy of Sciences.

Fig. 1. The person on the right is considering where should the next step go. The chess he has played and the chess he has seen are the most two influential factors on making decisions. His memory fused into his policy model. So how can robots remember and make decisions like humans? Motivated by this human cognitive science, we propose the LFRL in cloud robot systems. LFRL makes the cloud remember what robots learned before like a human brain.

reinforcement learning for cloud robot navigation. LFRL enables robots to remember what they have learned and what other robots have learned with cloud robotic systems. LFRL contains both asynchronization and synchronization learning rather than limited to synchronous learning as A3C [3] or UNREAL [4]. To demonstrate the efficacy of LFRL, we test LFRL in some public and self-made training environments. Experimental result indicates that LFRL is capable of enabling robots effectively use prior knowledge and quickly adapt to new environments. Overall, this paper makes the following contributions:

- We present a Lifelong Federated Reinforcement Learning architecture based on human cognitive science. It makes robots perform lifelong learning of navigation in cloud robotic systems.
- We propose a knowledge fusion algorithm. It is able to fuse prior knowledge of robots and evolve the shared model in cloud robotic systems.
- Two effective transfer learning approaches are introduced to make robots quickly adapt to new environments.
- A cloud robotic navigation-learning website is built in the work: *www.shared-robotics.com*. It provides the service based on LFRL.

II. RELATED THEORY

A. Reinforcement learning for navigation

Eliminating the requirements for location, mapping or path planning procedures, several DRL works have been presented

that successful learning navigation policies can be achieved directly from raw sensor inputs: target-driven navigation [5], successor feature RL for transferring navigation policies [6], and using auxiliary tasks to boost DRL training [7]. Many follow-up works have also been proposed, such as embedding SLAM-like structure into DRL networks [8], or utilizing DRL for multi-robot collision avoidance [9]. Tai et al [10] successfully appplied DRL for mapless navigation by taking the sqarse 10-dimensional range findings and the target position , defining mobile robot coordinate frame as input and continuous steering commands as output. Zhu et al. [5] input both the first-person view and the image of the target object to the A3C model, formulating a target-driven navigation problem based on the universal value function approximators [11]. To make the robot learn to navigate, we adopt a reinforcement learning perspective, which is built on recent success of deep RL algorithms for solving challenging control tasks [12-15]. Zhang [16] presented a solution that can quickly adapt to new situations (e.g., changing navigation goals and environments). Making the robot quickly adapt to new situations is not enough, we also need to consider how to make robots capable of memory and evolution, which is similar to the main purpose of lifelong learning.

B. Lifelong machine learning

Lifelong machine learning, or LML [17], considers system that can learn many tasks from one or more domains over its lifetime. The goal is to sequentially store learned knowledge and to selectively transfer that knowledge when a robot learns a new task, so as to develop more accurate hypotheses or policies. Robots are confronted with different obstacles in different environments, including static and dynamic ones, which are similar to the multi-task learning in lifelong learning. Although learning tasks are the same, including reaching goals and avoiding obstacles, their obstacle types are different. There are static obstacles, dynamic obstacles, as well as different ways of movement in dynamic obstacles. Therefore, it can be regarded as a low-level multitasking learning.

A lifelong learning should be able to efficiently retain knowledge. This is typically done by sharing a representation among tasks, using distillation or a latent basis [18]. The agent should also learn to selectively use its past knowledge to solve new tasks efficiently. Most works have focused on a special transfer mechanism, i.e., they suggested learning differentiable weights are from a shared representation to the new tasks [4, 19]. In contrast, Brunskill and Li [20] suggested a temporal transfer mechanism, which identifies an optimal set of skills in new tasks. Finally, the agent should have a systematic approach that allows it to efficiently retain the knowledge of multiple tasks as well as an efficient mechanism to transfer knowledge for solving new tasks. Chen [21] proposed a lifelong learning system that has the ability to reuse and transfer knowledge from one task to another while efficiently retaining the previously learned knowledge-base in Minecraft. Although this method has achieved good results in Mincraft, there is a lack of multi-agent cooperative learning model. Learning different tasks in a same scene is similar but different for robot navigation learning.

C. Federated learning

LFRL realizes federated learning of multi robots through knowledge fusion. Federated learning was first proposed in [22], which showed its effectiveness through experiments on various datasets. In federated learning systems, the raw data is collected and stored at multiple edge nodes, and a machine learning model is trained from the distributed data without sending the raw data from the nodes to a central place [23, 24]. Different from the traditional joint learning method where multiple edges are learning at the same time, LFRL adopts the method of *first training then fusing* to reduce the dependence on the quality of communication[25, 26].

D. Cloud robotic system

LFRL fits well with cloud robotic system. Cloud robotic system usually relies on many other resources from a network to support its operation. Since the concept of the cloud robot was proposed by Dr. Kuffner of Carnegie Mellon University (now working at Google company) in 2010 [27], the research on cloud robots is rising gradually. At the beginning of 2011, the cloud robotic study program of RoboEarth [28] was initiated by the Eindhoven University of Technology. Google engineers have developed robot software based on the Android platform, which can be used for remote control based on the Lego mind-storms, iRobot Create and Vex Pro, etc. [29]. Wang et al. present a framework targeting near real-time MSDR, which grants asynchronous access to the cloud from the robots [30]. However, no specific navigation method for cloud robots has been proposed up to now. We believe that this is the first navigation learning architecture for cloud robotic systems.

Generally, this paper focuses on developing a reinforcement learning architecture for robot navigation, which is capable of lifelong federated learning and multi robots federated learning. This architecture is well fit in cloud robot systems.

III. METHODOLOGY

LFRL is capable of reducing training time without sacrificing accuracy of navigating decision in cloud robotic systems. LFRL uses Cloud-Robot-Environment setup to learn the navigation policy. LFRL consists of a cloud server, a set of environments, and one or more robots. We develop a federated learning algorithm to fuse private models into the shared model in the cloud. The cloud server fuses private models into the shared model, then evolves the shared model. As illustrated in Fig.2, LFRL is an implementation of lifelong learning for navigation in cloud robotic systems. Compared with A3C or UNREAL approaches which update parameters of the policy network at the same time, the proposed knowledge fusion approach is more suitable for the federated architecture of LFRL. The proposed approach is capable of fusing models with asynchronous evolution. The approach of updating parameters at the same time has certain requirements for environments, while the proposed knowledge fusion algorithm has no requirements for environments. Using generative network and dynamic weight labels are able to realize the integration of memory instead of A3C or UNREAL method, which only generates a decision model during learning and
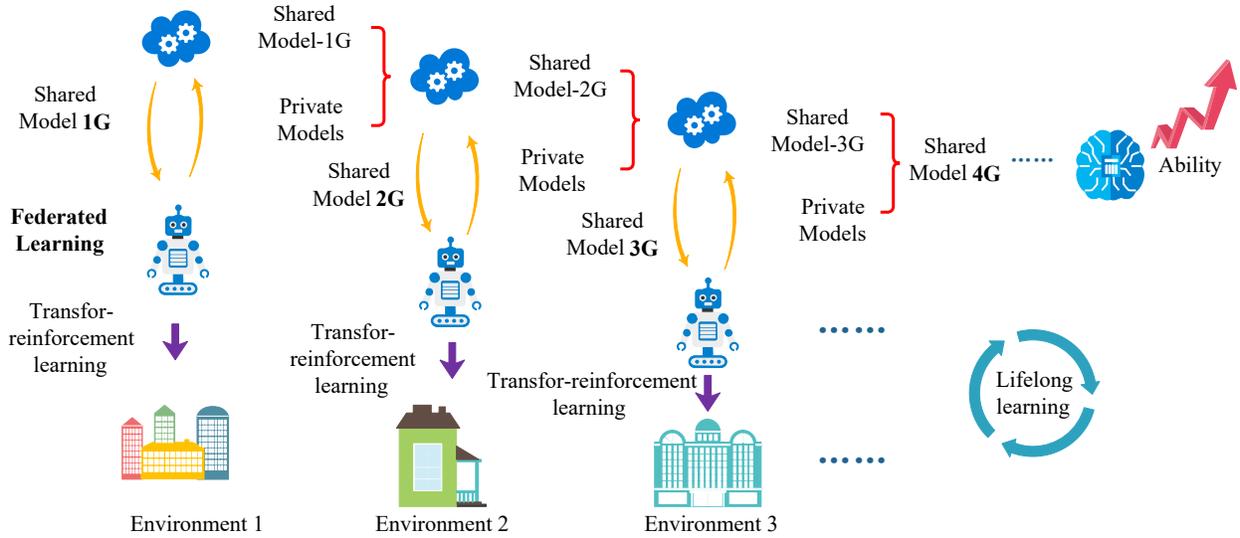
Fig. 2. Proposed Architecture. In Robot→Environment, the robot learns to avoid some new types of obstacles in the new environment through reinforcement learning and obtains the private Q-network model. Not only from one robot training in different environments, private models can also be resulted from multiple robots. It is a type of federated learning. After that, the private network will be uploaded to the cloud. The cloud server evolves the shared model by fusing private models to the shared model. In Cloud→Robot, inspired by transfer learning, successor features are used to transfer the strategy to unknown environment. We input the output of the shared model as added features to the Q-network in reinforcement learning, or simply transfer all parameters to the Q-network. Iterating this step, models on the cloud become increasingly powerful.
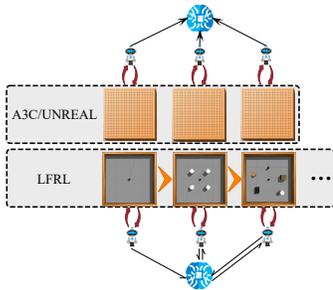

Fig. 3. LFRL compared with A3C or UNREAL

has no memory. As illustrated in Fig.3. In the algorithm of A3C or UNREAL, the training environment is constant. States of agents are countable. The central node only needs to fuse parameters, which can be performed at the same time. The two methods are capable of fusing parameters while training. Network structures of agents must be the same. However, in LFRL, the training environment is variable. State of agents are uncountable with more training environments uploading. In different agents, the structure of hiden layers of the policy network can be different. The cloud fuse training results. The robots are trained in new environments based on the shared model. Robots and the cloud have interactions in upload and download procedures. LFRL is more suitable for the cloud robot system where the environment is uncountable, especially in the lifelong learning framework.

A. Procedure of LFRL

This section displays a practical example of LFRL: there are 4 robots, 3 different environments and cloud servers. The first robot obtains its private strategy model Q1 through reinforcement learning in Environment 1 and upload it to the

cloud server as the shared model 1G. After a while, Robot 2 and Robot 3 desire to learn navigation by reinforcement learning in Environment 2 and Environment 3. In LFRL, Robot 2 and Robot 3 download the shared model 1G as the initial actor model in reinforcement learning. Then they can get their private networks Q2 and Q3 through reinforcement learning in Environment 2 and Environment 3. After completing the training, LFRL uploads Q2 and Q3 to the cloud. In the cloud, strategy models Q2 and Q3 will be fused into shared model 1G, and then shared model 2G will be generated. In the future, the shared model 2G can be used by other cloud robots. Other robots will also upload their private strategy models to the cloud server to promote the evolution of the shared model.

The more complicated tasks responded to more kinds of obstacles in robot navigation. The learning environment is gigantic in robot navigation learning. This case is different from the chess. So we borrow the idea of lifelong learning. Local robots will learn to avoid more kinds of obstacles and the cloud will fuse these skills. These skills will be used in more defined and undefined environments. For a cloud robotic system, the cloud generates a shared model for a time, which means an evolution in lifelong learning. The continuous evolution of the shared model in cloud is a lifelong learning pattern. In LFRL, the cloud server achieves the knowledge storage and fusion of a robot in different environments. Thus, the shared model becomes powerful through fusing the skills to avoid multi types of obstacles.

For an individual robot, when the robot downloads the cloud model, the initial Q-network has been defined. Therefore, the initial Q-network has the ability to reach the target and avoid some types of obstacles. It is conceivable that LFRL can reduce the training time for robots to learn navigation. Furthermore, there is a surprising experiment result that the

robot can get higher scores in navigation with LFRL. However,

---

Algorithm 1: Processing Algorithm in LFRL

Initialize action-value Q-network with random weights θ;
Input: θ_a: The parameters of the a-th shared model in
      cloud ; m: The number of private networks.
Output: The evolved θ_a
while *cloud server is running* **do**
    if *service_request=True* **then**
        Transfer θ_a to π;
        for $i = 1; i \leq m; i++$ **do**
            $\theta_i \leftarrow robot(i)$ perform reinforcement learning
               with π in environment.;
            Send θ_i to cloud;
        end
    end
    if *evolve time=True* **then**
        Generate $\theta_{a+1} = fuse(\theta_1, \theta_2, \cdots, \theta_m, \theta_a)$
        $\theta_a \leftarrow \theta_{a+1}$
    end
end

---

in actual operation, the cloud does not necessarily fuse models every time it receives a private network, but fuses at a fixed frequency rate. So, we present the processing flow of LFRL shown in Algorithm 1. Key algorithms in LFRL include knowledge function algorithm and transferring approaches, as introduced in the following.

B. Knowledge fusion algorithm in cloud

Inspired by images style transfer algorithm, we develop a knowledge fusion algorithm to evolve the shared model. This algorithm is based on generative networks and it is efficient to fuse parameters of networks trained from different robots or a robot in different environments. The algorithm deployed in the cloud server receives the privately transmitted network and upgrades the sharing network parameters. To address knowledge fusion, the algorithm generates a new shared model from private models and the shared model in cloud. This new shared model is the evolved model.

Fig.4 illustrates the process of generating a policy network. The structure of the policy network has the same input and output dimensions with the private policy network. The number of outputs is equal to the number of action types that robots can act. The dimensions of input also correspond with sensor data and human-made features. The training data of the network is randomly generated based on sensor data attributes. The label on each piece of data is dynamically weighted, which is based on the "confidence value" of each robot in each piece of data. We define robots as actors in reinforcement learning. Different robots or the same robot in different environments are different actors.

The "confidence value" motioned above of the actor is the degree of confirmation on which action the robot chooses to perform. For example, in a piece of sample from training data, the private Network 1 evaluates Q-values of different

actions to (85, 85, 84, 83, 86), but the evaluation of the k-G sharing network is (20, 20, 100, 10, 10). In this case, we are more confident on actor of k-G sharing network, because it has significant differentiation in the scoring process. On the contrary, the scores from actor of private Network 1 are confusing. Therefore, when generating the labels, the algorithm calculates the confidence value according to the score of different actors. Then the scores are weighted with confidence value and summed up. Finally, we obtain labels of training data by executing the above steps for each piece of data. There are several approaches to define confidence, such as variance, standard deviation, and information entropy. From the definition of the above statistical indicators we can infer that using variance to describe uncertainty will fail in some cases because it requires uniform distribution of data and ignores the occurrence of extreme events. The variance needs to meet the relevant premise to describe the uncertainty of the information. Entropy is more suitable for describing the uncertainty of information than variance, which comes from the definition of entropy. Uncertainty is the embodiment of confidence. So, In this work, we use information entropy to define confidence. Formula (1) is quantitative function of robotic confidence (information entropy):

Robot j "confidence":

$$c_j = -\frac{1}{\ln m} \sum_{i=1}^{m} \left(\frac{score_{ij}}{\sum_{i=1}^{m} score_{ij}} \cdot \ln \left(\frac{score_{ij}}{\sum_{i=1}^{m} score_{ij}} \right) \right) \quad (1)$$

m is the action size of robot, n is the number of private networks. Memory weight of robot j:

$$w_j = \frac{(1 - c_j)}{\sum_{j=1}^{n} (1 - c_j)} \quad (2)$$

Knowledge fusion function:

$$label_j = \ score \ \times (c_1, c_2, \cdots c_m)^T \quad (3)$$

It should be noted that Fig.4 only shows the process of one sample generating one label. Actually, we need to generate a large number of samples. For each data sample, the confidence values of the actors are different, so the weight of each actor is not the same. For example, when we generate 50,000 different pieces of data, there are nearly 50,000 kinds of different combinations of confidence. These changing weights can be incorporated into the data labels, and enable the generated network to dynamically adjust the weights on different sensor data. In conclusion, knowledge fusion algorithm in cloud can be defined as:

$$\omega_j = (1 - c_j) \div \sum_{j=1}^{n} (1 - c_j) \quad (4)$$

$$y_i = \sum_{j=1}^{num} c_j \cdot \omega_j \quad (5)$$

$$L(y, h_\theta(x_i)) = \frac{1}{N} \sum_{i=1}^{N} (y_i - h_\theta(x_i))^2 \quad (6)$$

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L(y_i \cdot h_\theta(x_i)) \quad (7)$$

Score Matrix (knowledge storage)

Generate large amounts of sample data

Random Sensor Data

Random data about target

Feature 1

Feature2

Feature x

Get human-made features from sensor data

Private Network 1

Score 11 | Score 21 | ⋯ | Score m1

Private Network n

Score 1n | Score 2n | ⋯ | Score mn

Sharing Network **kG**(in cloud)

Score 1(n+1) | Score 2(n+1) | ⋯ | Score m(n+1)

Score=

Score 11	⋯	Score 1n	Score 1(n+1)
Score 21		Score 2n	Score 2(n+1)
Score m1	⋯	Score mn	Score m(n+1)

Normalize each column

Knowledge Fusion based "Confidence" of robots

$$c_j = -\frac{1}{\ln m}\sum_{i=1}^{m}\left(\frac{score_{ij}}{\sum_{i=1}^{m} score_{ij}}\cdot \ln\left(\frac{score_{ij}}{\sum_{i=1}^{m} score_{ij}}\right)\right)$$

$$label_j = score \times \left[(1-\frac{c_1}{n})/\sum_{j=1}^{n} c_j,\cdots (1-c_n/\frac{c_1}{n})/\sum_{j=1}^{n} c_j\right]^{T}$$

| Action 1 | Action 2 | ⋯ | Action m |
| Score 1 | Score 2 | ⋯ | Score m |

Generative Networks Training

Label

Sharing Network **(k+1)G**(in cloud)
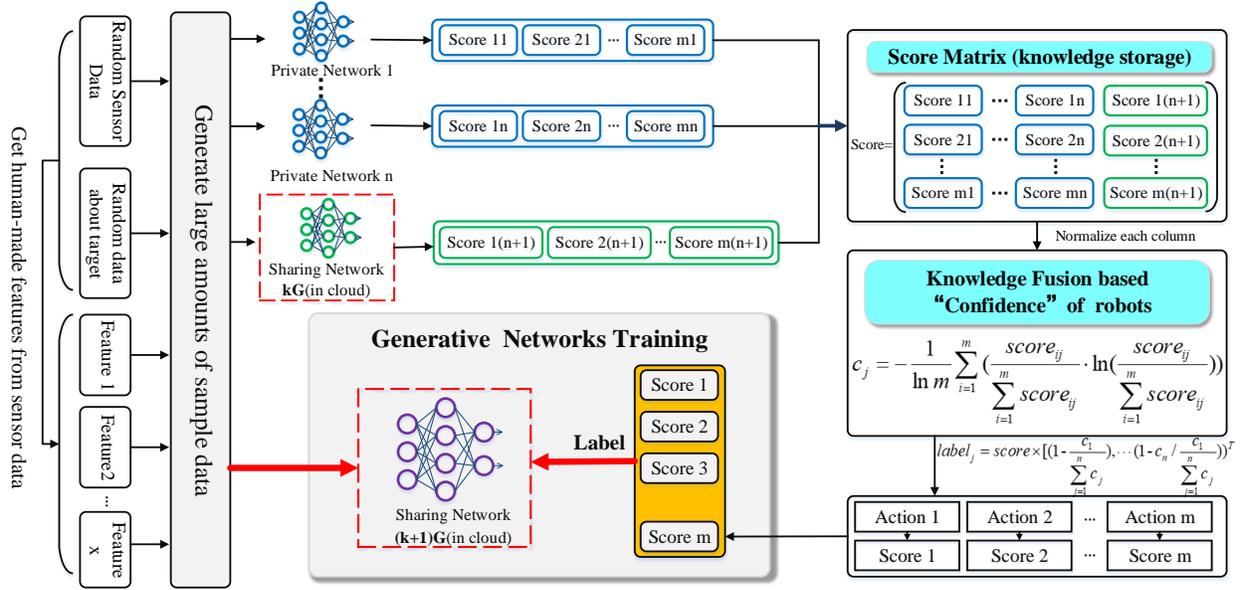
Score 1
Score 2
Score 3
⋮
Score m

Fig. 4. Knowledge Fusion Algorithm in LFRL: We generate a large amount of training data based on sensor data, target data, and human-defined features. Each training sample is added into the private network and the k-th generation sharing network, while different actors are scored for different actions. Then, we store the scores and calculate the confidence values of all actors in this training sample data. The "confidence value" is used as a weight, while the scores are weighted and summed to obtain the label of the current sample data. By analogy, all sample data labels are generated. Finally, a network is generated and fits the sample data as much as possible. The generated network is the (k+1)th generation. This step of fusion is finished.

Formula 4 takes the proportion of the confidence of robots as a weight. Formula 5 obtains the label of the sensor data by weighted summation. Formula 6 defines the error in the training process of generative network. Formula 7 is the goal of the training process. We discribe our approach in details in Algorithm 2. For a single robot, private network is obtained in different environments. Therefore, it can be regarded as asynchronous learning of the robot in LFRL. When there are multiple robots, we just need to treat them as the same robot in different environments. At this time, the evolution process is asynchronous, and multiple robots are synchronized.

It should be explained that the shared model in the cloud is not the final policy model of the local robot. We only use the shared model in the cloud as a pre-trained model or a feature extractor. The shared model maintained in the cloud is a cautious policy model but not the optimal for every robot. That is to say, the shared model in the cloud will not make serious mistakes in some private unstructured environments but the action is not the best. It is necessary for the robot to train its own policy model based on the shared model from the cloud, otherwise the error rate will be high in its private unstructured environment. As the saying goes, the older the person, the smaller the courage. In the process of lifelong learning, the cloud model will become more and more timid. In order to remove the error rate, we should transfer the shared model and train a new private model through reinforcement learning in a new environment. This responded to the transfer learning process in LFRL.

C. Transfer the shared model

Various approaches of transfer reinforcement learning have been proposed. In the specific task that a robot learns to navigate, we found that there are two applied approaches.

The first one is taking the shared model as initial actor network. While the other one is using the shared model as a feature extractor. If we adopt the first approach that takes the shared model as an initial actor network, abilities of avoiding obstacles and reaching targets can remain the same. In this approach, the robot deserves a good score at the beginning. The experimental data shows that the final score of the robot has been greatly improved at the same time. However, every coin has two sides, this approach is unstable. The training time depends on the adjustment of parameters in some extent. For example, we should accelerate updating speed, increase the punishment, reduce the probability of random action etc.

The shared model also can be used as a feature extractor in transfer reinforcement learning. As illustrated in Fig.5, this method increases the dimension of the features. So, it can improve the effect stably. One problem that needs to be solved in experiment is that there is a structural difference between input layer of the shared network and private network. The approach in LFRL is that the number of nodes in the input layer is consistent with the number of elements in the original feature vector, as shown in Fig.5. The features from transfer learning are not used as inputs to the shared network. They are just inputs of training private networks. This approach has high applicability, even though the shared model and private models have different network structures.

It is also worth noting that if the robot uses image sensors to acquire images as feature data. It is recommended to use the traditional transfer learning method that taking the output of some convolutional layers as features because the Q-network is a convolutional neural network. If a non-image sensor such as a laser radar is used, the Q-network is not a convolutional neural network, then we will use the output of the entire

Algorithm 2: Knowledge Fusion Algorithm

Initialize the shared network with random Parameters θ ;
Input: K: The number of data samples generated ; N:
The number of private networks; M: Action sizes of the robot;
Output: θ
for $i=1, i \leq N, i++$ **do**
    $\hat{x}_i \leftarrow$ Calculate indirct features from $\tilde{x}_i$;
    $x_i \leftarrow [\tilde{x}_i, \hat{x}_i]$;
    for $n=1, n \leq K, n++$ **do**
        $score_{in} \leftarrow f_n(x_i)$;
        $score_i \leftarrow$ score append $score_{in}$
    end
    for $n=1, n \leq K, n++$ **do**
        for $m=1, m \leq M, m++$ **do**
            $c_{in} \leftarrow$ Calculate the confidence value of the n-th private network in the i-th data based on formula (1)
        end
    end
    $label_i \leftarrow$ Calculate the $label_i$ based on formula (2) and (3);
    $label \leftarrow$ label append $label_i$;
end
$\theta \leftarrow$ training the shared network from (x,label);

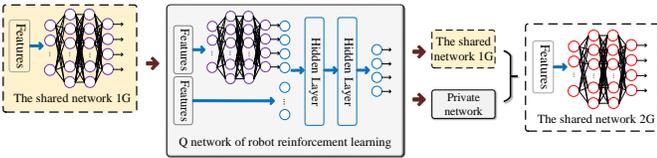network as additional features, as Fig.5 shows. The learning

Fig. 5. A transfer learning method of LFRL

process of the robot can roughly divided into two stages, the stage of avoiding obstacles and the stage of reaching the target. With the former transfer algorithm, it is possible to utilize the obstacle avoidance skills of the cloud model. In the latter transfer method, it is obvious that the evaluation of the directions by cloud model is valuable, which is useful features for navigation.

D. Explanation from human cognitive science

The design of the LFRL is inspired by the human decision-making process in cognitive science. For example, when playing chess, the chess player will make decisions based on the rules and his own experiences. The chess experiences include his own experiences and the experiences of other chess players he has seen. We can regard the chess player as a decision model. The quality of the decision model represents the performance level of the chess player. In general, this policy model will become increasingly excellent through experience accumulation, and the chess player's skill will be improved. This is the iteratively evolutionary process in LFRL. After each

chess player finishing playing chess, his chess level or policy model evolves, which is analogous to the process of knowledge fusion in LFRL. And these experiences will also be used in later chess player, which is analogous to the process of transfer learning in LFRL. Fig. 1 demonstrates a concrete example. The person on the right is considering where should the next step goes. The chess he has played and the chess he has seen are the most two influential factors on making decision. But his chess experiences may influence the next step differently. At this time, according to human cognitive science, the man will be more influenced by experiences with clear judgments. An experience with a clear judgment will have a higher weight in decision making. This procedure of humans makes decisions is analogous to knowledge fusion algorithm in LFRL. The influence of different chess experience is always dynamic in the decision of each step. The knowledge fusion algorithm in LFRL achieves this cognitive phenomenon by adaptively weighting the labels of training data. The chess player is a decision model that incorporates his own experiences. Corresponding to this opinion, LFRL integrates experience into one decision model by generating a network. This process is also analogous to the operation of human cognitive science.

IV. EXPERIMENTS

In this section, we intend to answer three questions: 1) Can LFRL help reduce training time without sacrifice accuracy of navigation in cloud robotic systems? 2) Does the knowledge fusion algorithm is effective to increase the shared model? 3) Are transfer learning approaches effective to transfer the shared model to specific task? To answer the former question, we conduct experiments to compare the performance of the generic approach and LFRL. To answer the second question, we conduct experiments to compare the performance of generic models and the shared model in transfer reinforcement learning. To answer the third question, we conduct experiments to compare the performance of the two transfer learning approaches and the native reinforcement learning.

A. Experimental setup

The training procedure of the LFRL was implemented in virtual environment simulated by gazebo. Four training environments were constructed to show the different consequence between the generic approach training from scratch and LFRL, as shown in Fig.6. There is no obstacle in Env-1 except the walls. There are four static cylindrical obstacles in Env-2, four moving cylindrical obstacles in Env-3. More complex static obstacles are in Env-4. In every environment, a *Turtlebot3* equipped with a laser range sensor is used as the robot platform. The scanning range is from 0.13m to 4m. The target is represented by a red square object. During the training process, the starting pose of the robot is the geometric center of the ground in the training environment. The target is randomly generated in pose where there are no obstacles in the environment. An episode is terminated after the agent either reaches the goal, collides with an obstacle, or after a maximum of 6000 steps during training and 1000 for testing. We calculated the average reward of the robot every two minutes. We end the training when the average reward of the

(a) Env-1   (b) Env-2   (c) Env-3   (d) Env-4

(e) Env-1 score   (f) Env-2 score   (g) Env-3 score   (h) Env-4 score
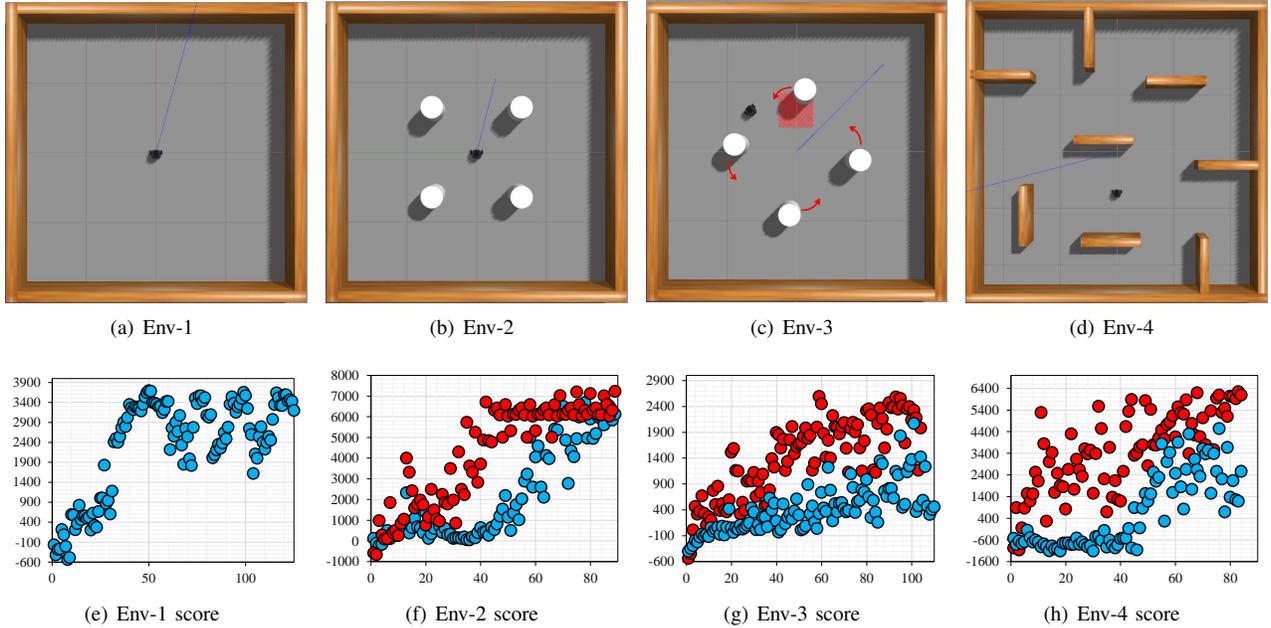
Fig. 6. We present both quantitative and compared results: Subfigure a to Subfigure d are the training environments. Subfigure e to h present scores of the generic approach (blue) compared with LFRL approach (red) in training process.In the training procedure of Env-1, LFRL has the same result with generic approaches. Because there is no antecedent shared models for the robot. In the training procedure of Env-2, LFRL obtained the shared model 1G, which made LFRL get higher reward in less time compared with the generic approach. In Env-3 and Env-4, LFRL evolve the shared model to 2G and 3G and obtained excellent result. From this figure, we demonstrate that LFRL can get higher reward in less time compared with the generic approach.

agent is enough and stable. We trained the model from scratch on a single Nvidia GeForce GTX 1070 GPU. The actor-critic network used two fully connected layers with 64 units. The output is used to produce the discrete action probabilities by a linear layer followed by a softmax, and the value function by a linear layer.

B. Evaluation for the architecture

To show the performance of LFRL, we tested it and compared with generic methods in the four environments. Then we started the training procedure of LFRL. As mentioned before, we initialized the shared model and evolved it as Algorithm 2 after training in Env-1. In the cloud robotic system, the robot downloaded the shared model 1G. Then, the robot performed reinforcement learning based on the shared model. The robot got a private model after training and it would be uploaded to the cloud server. The cloud server fused the private model and the shared model 1G to obstain the shared model 2G. With the same mode, follow-up evolutions would be performed. We constructed four environments, so the shared model upgraded to 4G. Performance of LFRL shown in Fig.6 where also shows generic methods performance. In Env2-Env4, LFRL increased accuracy of navigating decision and reduced training time in the cloud robotic system. From the last row of Fig.6, we can observe that the improvement are more efficient with the shared model. LFRL is highly effective for learning a policy over all considered obstacles. It improves the generalization capability of our trained model across commonly encountered environments. Experiments demonstrate that LFRL is capable of reducing training time without sacrificing accuracy of navigating decision in cloud robotic systems.

C. Evaluation for the knowledge fusion algorithm

In order to verify the effectiveness of the knowledge fusion algorithm, we conducted a comparative experiment. We created three new environments that were not present in the previous experiments. These environments are more similar to real situations: Static obstacles such as cardboard boxes, dustbin, cans are in the Test-Env-1. Moving obstacles such as stakes are in Test-Env-2. Test-Env-3 includes more complex static obstacles and moving cylindrical obstacles. We still used the *Turtlebot3* created by gazebo as the testing platform. In order to vertify the advancement of the shared model, we trained the navigation policy based on the generic model 1, model 2, model 3, model 4 in Test-Env-1, Test-Env-2 and Test-Env-3 respectively. The generic models are from the previous generic approaches experiments. These policy models are trained from one environment without knowledge fusion. According to the hyper-parameters and complexity of environments, the average score goal (5 consecutive times above a certain score) in Test-Env-1 is 4000, Test-Env-2 is 3000, Test-Env-3 is 2600.

In the following, we present compared results in Fig.7 and quantitative results in Table 1. The shared model steadily reduces training time. In particular, we can observe that the generic method models are only able to make excellent decisions in individual environments; while the shared model is able to make excellent decisions in plenty of different environments. So, the proposed knowledge fusion algorithm in this paper is effective.

(a) Env-1 (b) Env-2 (c) Env-3

(d) Env-1 stacked scores (e) Env-2 stacked scores (f) Env-3 stacked scores (sampling)
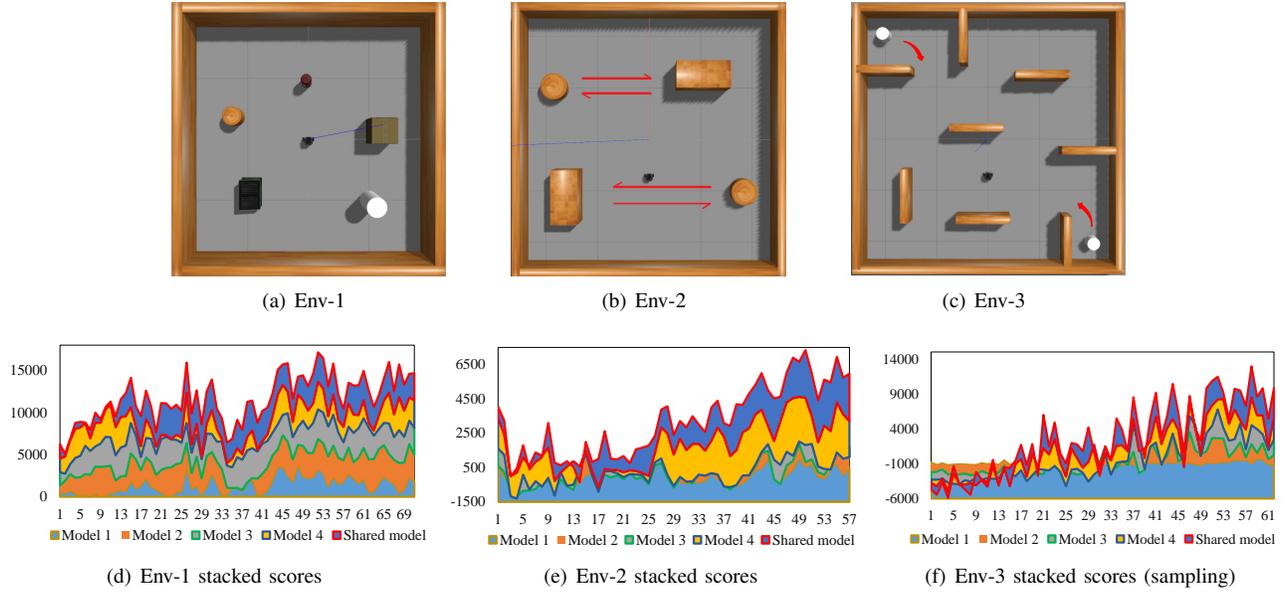
Fig. 7. We present both quantitative and compared results: Subfigure a to Subfigure c are the testing environments. Because of the large amount of data, we present the stacked figure to better repersent the comparison. From the subfigure d to subfigure f, it can seen that the reward of the shared model accounts for a larger proportion in all positive rewards (greater than 0) models in the later stage of training.

TABLE I

RESULTS OF THE CONTRAST EXPERIMENT

	Time to meet requirement			Averagge scores			Average of the last five scores		
	Test-Env-1	Test-Env-2	Test-Env-3	Test-Env-1	Test-Env-2	Test-Env-3	Test-Env-1	Test-Env-2	Test-Env-3
model 1	1h 32min	>3h	>6 h	1353.36	46.5	-953.56	1421.54	314.948	-914.16
model 2	33min	>3h	>6 h	2631.57	71.91	288.61	3516.34	794.02	-132.07
model 3	37min	>3h	5h 50min	2925.53	166.5	318.04	3097.64	-244.17	1919.12
model 4	4h 41min	2h 30min	5h 38min	1989.51	1557.24	1477.5	2483.18	2471.07	3087.83
Shared model	55min	2h 18min	4h 48min	2725.16	1327.76	1625.61	3497.66	2617.02	3670.92

The background color of the cell reflects the performance of the corresponding model. The darker the color, the better the performance.

D. Evaluation for the two transfer learning approaches

In order to verify and compare the two transfer learning approaches, we conducted a comparative experiment. The result is present in Fig.8. It can be seen from the figure that both transfer learning approaches can effectively improve the efficiency of reinforcement learning. Among them, the approach of parameter transferring has faster learning speed and the approach of feature extractor has higher stability.
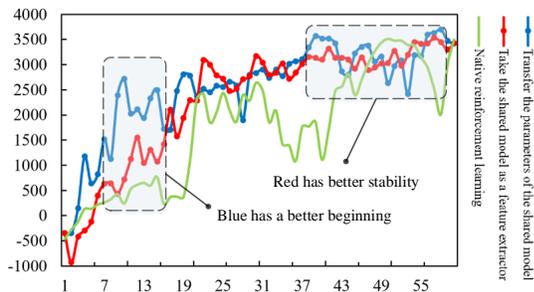

Fig. 8. Transfer learning approaches comparison
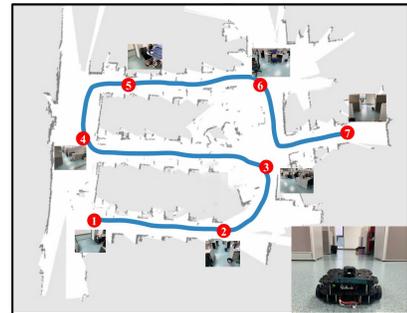
E. Real world experiments

Fig. 9. Trajectory tracking in the real environment

We also conducted real-world experiments to test the performance of our approach with different sensor noise. We use Turtlebot 3 platform, which is shown in Fig.9. The shared model 4G in the cloud was downloaded. Then we performed transferring reinforcement learning in Env-4 and got the policy finally. The Turtlebot navigates automatically in an indoor office environment as shown in Fig.9 under the

policy. The experiment indicates that the policy is reliable in real environment. The reference [10] also corroborates the conclusion.

V. CONCLUSION

We presented a learning architecture LFRL for navigation in cloud robotic systems. The architecture is able to make navigation-learning robots effectively use prior knowledge and quickly adapt to new environment. Additionally, we presented a knowleged fusion algorithm in LFRL and introduced transfer methods. Our approach is able to fuse models and asynchronously evolve the shared model. We validated our architecture and algorithmes in policy-learning experiments and realsed a website to provide the service.

The architecture has fixed requirements for the dimensions of input sensor signal and the dimensions of action. We leave it as future work to make LFRL flexible to deal with different input and output dimensions. The more flexible LFRL will offer a wider range of services in cloud robotic systems.

VI. ACKNOWLEDGEMENT

REFERENCES

[1] Y. Li, Deep Reinforcement Learning, arXiv preprint arXiv:1810.16339, 2018.
[2] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, A Survey of Research on Cloud Robotics and Automation, IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, pp. 398-409, 2015.
[3] V. Mnih et al., Asynchronous methods for deep reinforcement learning, in International conference on machine learning (ICML), 2016, pp. 1928-1937.
[4] M. Jaderberg et al., Reinforcement learning with unsupervised auxiliary tasks, in International Conference on Learning Representations (ICLR), 2017.
[5] Y. Zhu et al., Target-driven visual navigation in indoor scenes using deep reinforcement learning, in IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3357-3364.
[6] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, Deep reinforcement learning with successor features for navigation across similar environments, in IEEE International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2371-2378.
[7] P. Mirowski et al., Learning to Navigate in Complex Environments, in International Conference on Learning Representations (ICLR), 2017.
[8] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, Neural SLAM: Learning to Explore with External Memory, arXiv preprint arXiv:1706.09520, 2017.
[9] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, in IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 6252-6259.
[10] L. Tai, G. Paolo, and M. Liu, Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation, in IEEE International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 31-36.
[11] T. Schaul, D. Horgan, K. Gregor, and D. Silver, Universal Value Function Approximators, in International Conference on Machine Learning (ICML), 2015, pp. 1312-1320.
[12] V. Mnih et al., Human-level control through deep reinforcement learning, Nature, vol. 518, no. 7540, pp. 529-533, 2015.
[13] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, Optimal and Autonomous Control Using Reinforcement Learning: A Survey, IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2042-2062, 2018.
[14] H. Xu, Y. Gao, F. Yu, and T. Darrell, End-to-end learning of driving models from large-scale video datasets, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2174-2182.
[15] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation, Robotics and Autonomous Systems, vol. 112, pp. 72-83, 2019.
[16] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi, Lifelong Machine Learning: Second Edition, 2nd ed. Morgan and Claypool, 2018.
[17] A. A. Rusu et al., Policy Distillation, in International Conference on Learning Representations (ICLR), 2016.
[18] H. B. Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, Online Multi-Task Learning for Policy Gradient Methods, in International Conference on Machine Learning (ICML), 2014, pp. 1206-1214.
[19] A. A. Rusu et al., Progressive Neural Networks, in Conference and Workshop on Neural Information Processing Systems (NIPS), 2016.
[20] E. Brunskill and L. Li, PAC-inspired Option Discovery in Lifelong Reinforcement Learning, International Conference on Machine Learning (ICML), pp. 316-324, 2014.
[21] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, A Deep Hierarchical Approach to Lifelong Learning in Minecraft., in AAAI, 2017, vol. 3, pp. 1553-1561.
[22] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, Federated Learning of Deep Networks using Model Averaging, arXiv preprint arXiv:1602.05629, 2016.
[23] M. Nasr, R. Shokri, and A. Houmansadr, Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks, in IEEE Symposium on Security and Privacy, 2018, pp. 1-15.
[24] A. Hard et al., Federated Learning for Mobile Keyboard Prediction, arXiv preprint arXiv:1811.03604, 2018.
[25] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning, arXiv preprint arXiv:1809.07857, 2018.
[26] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, Distributed Federated Learning for Ultra-Reliable Low-Latency Vehicular Communications, arXiv preprint arXiv:1807.08127, 2018.
[27] J.Kuffner, Cloud-enabled Robots, in International Conference on Humanoid Robot, 2010.
[28] M. Waibel et al., RoboEarth - A World Wide Web for Robots, IEEE Robotics and Automation Magazine, vol. 18, no. 2, pp. 69-82, 2011.
[29] M. Srinivasan, K. Sarukesi, N. Ravi, and M. T, Design and Implementation of VOD (Video on Demand) SaaS Framework for Android Platform on Cloud Environment, in IEEE International Conference on Mobile Data Management, 2013, pp. 171-176.
[30] Wang L , Liu M , Meng Q H. Real-Time Multisensor Data Retrieval for Cloud Robotic Systems, IEEE Transactions on Automation Science and Engineering, 2015, Vol. 12, no. 2, pp. 507-518.