Intelligence & Robotics

Review

Open Access

Federated reinforcement learning: techniques, applications, and open challenges

Jiaju Qi[1], Qihao Zhou[2], Lei Lei[1], Kan Zheng[2]

[1]School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada.
[2]Intelligent Computing and Communications (IC^2) Lab, Beijing University of Posts and Telecommunications, Beijing 100876, China.

Correspondence to: Dr. Lei Lei, School of Engineering, University of Guelph, 50 Stone Road East, Guelph, ON N1G 2W1, Canada. E-mail: leil@uoguelph.ca

How to cite this article: Qi J, Zhou Q, Lei L, Zheng K. Federated reinforcement learning: techniques, applications, and open challenges. *Intell Robot* 2021;1(1):18-57. http://dx.doi.org/10.20517/ir.2021.02

Received: 24 Aug 2021 **First Decision:** 14 Sep 2021 **Revised:** 21 Sep 2021 **Accepted:** 22 Sep 2021 **Published:** 12 Oct 2021

Academic Editor: Simon X. Yang **Copy Editor:** Xi-Jun Chen **Production Editor:** Xi-Jun Chen

Abstract

This paper presents a comprehensive survey of federated reinforcement learning (FRL), an emerging and promising field in reinforcement learning (RL). Starting with a tutorial of federated learning (FL) and RL, we then focus on the introduction of FRL as a new method with great potential by leveraging the basic idea of FL to improve the performance of RL while preserving data-privacy. According to the distribution characteristics of the agents in the framework, FRL algorithms can be divided into two categories, *i.e.*, horizontal federated reinforcement learning and vertical federated reinforcement learning (VFRL). We provide the detailed definitions of each category by formulas, investigate the evolution of FRL from a technical perspective, and highlight its advantages over previous RL algorithms. In addition, the existing works on FRL are summarized by application fields, including edge computing, communication, control optimization, and attack detection. Finally, we describe and discuss several key research directions that are crucial to solving the open problems within FRL.

Keywords: Federated learning, reinforcement learning, federated reinforcement learning

1. INTRODUCTION

As machine learning (ML) develops, it becomes capable of solving increasingly complex problems, such as image recognition, speech recognition, and semantic understanding. Despite the effectiveness of traditional machine learning algorithms in several areas, the researchers found that scenes involving many parties are still difficult to resolve, especially when privacy is concerned. Federated learning (FL), in these cases, has attracted increasing interest among ML researchers. Technically, the FL is a decentralized collaborative approach that allows multiple partners to train data respectively and build a shared model while maintaining privacy. With its innovative learning architecture and concepts, FL provides safer experience exchange services and enhances capabilities of ML in distributed scenarios.

In ML, reinforcement learning (RL) is one of the branches that focuses on how individuals, *i.e.*, agents, interact with their environment and maximize some portion of the cumulative reward. The process allows agents to learn to improve their behavior in a trial and error manner. Through a set of policies, they take actions to explore the environment and expect to be rewarded. Research on RL has been hot in recent years, and it has shown great potential in various applications, including games, robotics, communication, and so on.

However, there are still many problems in the implementation of RL in practical scenarios. For example, considering that in the case of large action space and state space, the performance of agents is vulnerable to collected samples since it is nearly impossible to explore all sampling spaces. In addition, many RL algorithms have the problem of learning efficiency caused by low sample efficiency. Therefore, through information exchange between agents, learning speed can be greatly accelerated. Although distributed RL and parallel RL algorithms[1–3] can be used to address the above problems, they usually need to collect all the data, parameters, or gradients from each agent in a central server for model training. However, one of the important issues is that some tasks need to prevent agent information leakage and protect agent privacy during the application of RL. Agents' distrust of the central server and the risk of eavesdropping on the transmission of raw data has become a major bottleneck for such RL applications. FL can not only complete information exchange while avoiding privacy disclosure, but also adapt various agents to their different environments. Another problem of RL is how to bridge the simulation-reality gap. Many RL algorithms require pre-training in simulated environments as a prerequisite for application deployment, but one problem is that the simulated environments cannot accurately reflect the environments of the real world. FL can aggregate information from both environments and thus bridge the gap between them. Finally, in some cases, only partial features can be observed by each agent in RL. However, these features, no matter observations or rewards, are not enough to obtain sufficient information required to make decisions. At this time, FL makes it possible to integrate this information through aggregation.

Thus, the above challenges give rise to the idea of federated reinforcement learning (FRL). As FRL can be considered as an integration of FL and RL under privacy protection, several elements of RL can be presented in FL frameworks to deals with sequential decision-making tasks. For example, these three dimensions of sample, feature and label in FL can be replaced by environment, state and action respectively in FRL. Since FL can be divided into several categories according to the distribution characteristics of data, including horizontal federated learning (HFL) and vertical federated learning (VFL), we can similarly categorize FRL algorithms into horizontal federated reinforcement learning (HFRL) and vertical federated reinforcement learning (VFRL).

Though a few survey papers on FL[4–6] have been published, to the best of our knowledge, there are currently no relevant survey papers focused on FRL. Due to the fact that FRL is a relatively new technique, most researchers may be unfamiliar with it to some extent. We hope to identify achievements from current studies and serve as a stepping stone to further research. In summary, this paper sheds light on the following aspects.

1. *Systematic tutorial on FRL methodology*. As a review focusing on FRL, this paper tries to explain the knowledge about FRL to researchers systematically and in detail. The definition and categories of FRL are introduced firstly, including system model, algorithm process, *etc*. In order to explain the framework of HFRL and VFRL and the difference between them clearly, two specific cases are introduced, *i.e.*, autonomous driving and smart grid. Moreover, we comprehensively introduce the existing research on FRL's algorithm

design.

2. *Comprehensive summary for FRL applications*. This paper collects a large number of references in the field of FRL, and provides a comprehensive and detailed investigation of the FRL applications in various areas, including edge computing, communications, control optimization, attack detection, and some other applications. For each reference, we discuss the authors' research ideas and methods, and summarize how the researchers combine the FRL algorithm with the specific practical problems.
3. *Open issues for future research*. This paper identifies several open issues for FRL as a guide for further research. The scope covers communication, privacy and security, join and exit mechanisms design, learning convergence and some other issues. We hope that they can broaden the thinking of interested researchers and provide help for further research on FRL.

The organization of this paper is as follows. To quickly gain a comprehensive understanding of FRL, the paper starts with FL and RL in Section 2 and Section 3, respectively, and extends the discussion further to FRL in Section 4. The existing applications of FRL are summarized in Section 5. In addition, a few open issues and future research directions for FRL are highlighted in Section 6. Finally, the conclusion is given in Section 7.

2. FEDERATED LEARNING

2.1. Federated learning definition and basics

In general, FL is a ML algorithmic framework that allows multiple parties to perform ML under the requirements of privacy protection, data security, and regulations[7]. In FL architecture, model construction includes two processes: model training and model inference. It is possible to exchange information about the model between parties during training, but not the data itself, so that data privacy will not be compromised in any way. An individual party or multiple parties can possess and maintain the trained model. In the process of model aggregation, more data instances collected from various parties contribute to updating the model. As the last step, a fair value-distribution mechanism should be used to share the profits obtained by the collaborative model[8]. The well-designed mechanism enables the federation sustainability. Aiming to build a joint ML model without sharing local data, FL involves technologies from different research fields such as distributed systems, information communication, ML and cryptography[9]. FL has the following characteristics as a result of these techniques, *i.e.*,

- Distribution. There are two or more parties that hope to jointly build a model to tackle similar tasks. Each party holds independent data and would like to use it for model training.
- Data protection. The data held by each party does not need to be sent to the other during the training of the model. The learned profits or experiences are conveyed through model parameters that do not involve privacy.
- Secure communication. The model is able to be transmitted between parties with the support of an encryption scheme. The original data cannot be inferred even if it is eavesdropped during transmission.
- Generality. It is possible to apply FL to different data structures and institutions without regard to domains or algorithms.
- Guaranteed performance. The performance of the resulting model is very close to that of the ideal model established with all data transferred to one centralized party.
- Status equality. To ensure the fairness of cooperation, all participating parties are on an equal footing. The shared model can be used by each party to improve its local models when needed.

A formal definition of FL is presented as follows. Consider that there are N parties $\{\mathcal{F}_i\}_{i=1}^N$ interested in establishing and training a cooperative ML model. Each party has their respective datasets $\mathcal{D}_i$. Traditional ML approaches consist of collecting all data $\{\mathcal{D}_i\}_{i=1}^N$ together to form a centralized dataset $\mathbb{R}$ at one data server. The expected model $\mathcal{M}_{SUM}$ is trained by using the dataset $\mathbb{R}$. On the other hand, FL is a reform of ML process in which the participants $\mathcal{F}_i$ with data $\mathcal{D}_i$ jointly train a target model $\mathcal{M}_{FED}$ without aggregating their data. Respective data $\mathcal{D}_i$ is stored on the owner $\mathcal{F}_i$ and not exposed to others. In addition, the performance measure of the federated model $\mathcal{M}_{FED}$ is denoted as $\mathcal{V}_{FED}$, including accuracy, recall, and F1-score, *etc*, which should be a good approximation of the performance of the expected model $\mathcal{M}_{SUM}$, *i.e.*, $\mathcal{V}_{SUM}$. In order to quantify differences in performance, let δ be a non-negative real number and define the federated learning model $\mathcal{M}_{FED}$ has δ performance loss if

$$|\mathcal{V}_{SUM} - \mathcal{V}_{FED}| < \delta.$$

Specifically, the FL model hold by each party is basically the same as the ML model, and it also includes a set of parameters w_i which is learned based on the respective training dataset $\mathcal{D}_i$[10]. A training sample j typically contains both the input of FL model and the expected output. For example, in the case of image recognition, the input is the pixel of the image, and the expected output is the correct label. The learning process is facilitated by defining a loss function on parameter vector w for every data sample j. The loss function represents the error of the model in relation to the training data. For each dataset $\mathcal{D}_i$ at party $\mathcal{F}_i$, the loss function on the collection of training samples can be defined as follow[11],

$$F_i(w) = \frac{1}{|\mathcal{D}_i|} \sum_{j \in \mathcal{D}_i} f_j(w),$$

where $f_j(w)$ denotes the loss function of the sample j with the given model parameter vector w and $|\cdot|$ represents the size of the set. In FL, it is important to define the global loss function since multiple parties are jointly training a global statistical model without sharing a dataset. The common global loss function on all the distributed datasets is given by,

$$F_g(w) = \sum_{i=1}^{N} \eta_i F_i(w),$$

where η_i indicates the relative impact of each party on the global model. In addition, $\eta_i > 0$ and $\sum_{i=1}^{N} \eta_i = 1$. This term η can be flexibly defined to improve training efficiency. The natural setting is averaging between parties, *i.e.*, $\eta = 1/N$. The goal of the learning problem is to find the optimal parameter that minimizes the global loss function $F_g(w)$. It is presented in formula form,

$$w^* = \arg\min_{w} F_g(w).$$

Considering that FL is designed to adapt to various scenarios, the objective function may be appropriate depending on the application. However, a closed-form solution is almost impossible to find with most FL models due to their inherent complexity. A canonical federated averaging algorithm (FedAvg) based on gradient-descent techniques is presented in the study from McMahan *et al.*[12], which is widely used in FL systems. In general, the coordinator has the initial FL model and is responsible for aggregation. Distributed participants know the optimizer settings and can upload information that does not affect privacy. The specific architecture of FL will be discussed in the next subsection. Each participant uses their local data to perform one step (or multiple steps) of gradient descent on the current model parameter $\bar{w}(t)$ according to the following formula,

$$\forall i, w_i(t+1) = \bar{w}(t) - \gamma \nabla F_i(\bar{w}_i(t)),$$

where γ denotes a fixed learning rate of each gradient descent. After receiving the local parameters from participants, the central coordinator updates the global model using a weighted average, *i.e.*,

$$\bar{w}_g(t+1) = \sum_{i=1}^{N} \frac{n_i}{n} w_i(t+1),$$

where n_i indicates the number of training data samples of the i-th participant has and n denotes the total number of samples contained in all the datasets. Finally, the coordinator sends the aggregated model weights $\bar{w}_g\ (t+1)$ back to the participants. The aggregation process is performed at a predetermined interval or iteration round. Additionally, FL leverages privacy-preserving techniques to prevent the leakage of gradients or model weights. For example, the existing encryption algorithms are added on top of the original FedAvg to provide secure FL[13,14].

2.2. Architecture of federated learning

According to the application characteristics, the architecture of FL can be divided into two types[7], *i.e.*, client-server model and peer-to-peer model.

As shown in Figure 1, there are two major components in the client-server model, *i.e.*, participants and coordinators. The participants are the data owners and can perform local model training and updates. In different scenarios, the participants are made up of different devices, the vehicles in the internet of vehicles (IoV), or the smart devices in the IoT. In addition, participants usually possess at least two characteristics. Firstly, each participant has a certain level of hardware performance, including computation power, communication and storage. The hardware capabilities ensure that the FL algorithm operates normally. Secondly, participants are independent of one another and located in a wide geographic area. In the client-server model, coordinator can be considered as a central aggregation server, which can initialize a model and aggregate model updates from participants[12]. As participants train both based on local data sets concurrently and share their experience through the coordinator with the model aggregation mechanism, it will greatly enhance the efficiency of the training and enhance the performance of the model. However, since participants won't be able to communicate directly, the coordinator must perform well to train the global model and maintain communication with all participants. Therefore, the model has security challenges such as a single point of failure. If the coordinator fails to complete the model aggregation task, the local model of participant has difficulty meeting target performance. The basic workflow of the client-server model can be summarized in the following five steps. The process continues to repeat the steps from 2 to 5 until the model converges, or until the maximum number of iterations is reached.

- Step 1: In the process of setting up a client-server-based learning system, the coordinator creates an initial model and sends it to each participant. Those participants who join later can access the latest global model.
- Step 2: Each participant trains a local model based on their respective dataset.
- Step 3: Updates of model parameters are sent to the central coordinator.
- Step 4: The coordinator combines the model updates using specific aggregation algorithms.
- Step 5: The combined model is sent back to the corresponding participant.

The peer-to-peer based FL architecture does not require a coordinator as illustrated in Figure 2. Participants can directly communicate with each other without relying on a third party. Therefore, each participant in the architecture is equal and can initiate a model exchange request with anyone else. As there is no central server, participants must agree in advance on the order in which model should be sent and received. Common transfer modes are cyclic transfer and random transfer. The peer-to-peer model is suitable and important for specific scenarios. For example, multiple banks jointly develop an ML-based attack detection model. With FL, there is no need to establish a central authority between banks to manage and store all attack patterns. The attack record is only held at the server of the attacked bank, but the detection experience can be shared with other participants through model parameters. The FL procedure of the peer-to-peer model is simpler than that of the client-server model.

- Step 1: Each participant initializes their local model depending on its needs.
- Step 2: Train the local model based on the respective dataset.
- Step 3: Create a model exchange request to other participants and send local model parameters.
- Step 4: Aggregate the model received from other participants into the local model.

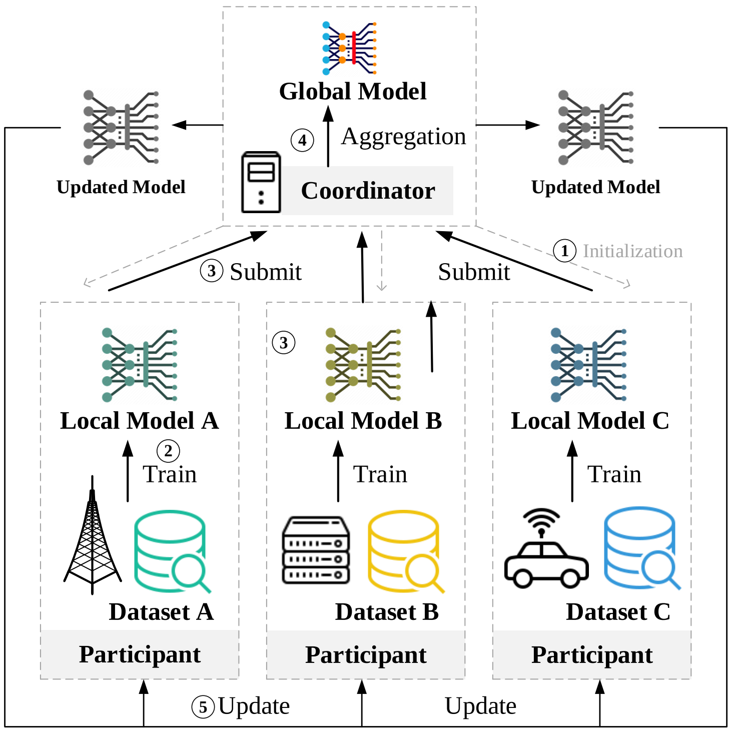

Figure 1. An example of federated learning architecture: Client-Server Model.

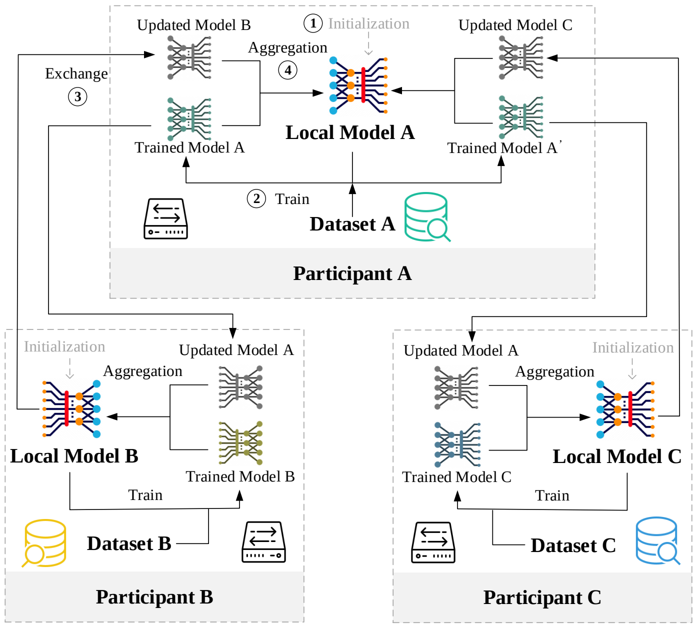

Figure 2. An example of federated learning architecture: Peer-to-Peer Model.

The termination conditions of the process can be designed by participants according to their needs. This architecture further guarantees security since there is no centralized server. However, it requires more communication resources and potentially increased computation for more messages.

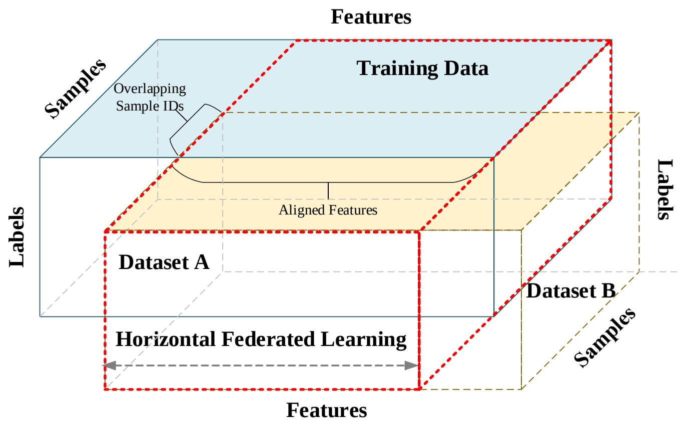

Figure 3. Illustration of horizontal federated learning.

2.3. Categories of federated learning

Based on the way data is partitioned within a feature and sample space, FL may be classified as HFL, VFL, or federated transfer learning (FTL)[8]. In Figure 3, Figure 4, and Figure 5, these three federated learning categories for a two-party scenario are illustrated. In order to define each category more clearly, some parameters are formalized. We suppose that the i-th participant has its own dataset $\mathcal{D}_i$. The dataset includes three types of data, *i.e.*, the feature space $\mathcal{X}_i$, the label space $\mathcal{Y}_i$ and the sample ID space $\mathcal{I}_i$. In particular, the feature space $\mathcal{X}_i$ is a high-dimensional abstraction of the variables within each pattern sample. Various features are used to characterize data held by the participant. All categories of association between input and task target are collected in the label space $\mathcal{Y}_i$. The sample ID space $\mathcal{I}_i$ is added in consideration of actual application requirements. The identification can facilitate the discovery of possible connections among different features of the same individual.

HFL indicates the case in which participants have their dataset with a small sample overlap, while most of the data features are aligned. The word "horizontal" is derived from the term "horizontal partition". This is similar to the situation where data is horizontally partitioned inside the traditional tabular view of a database. As shown in Figure 3, the training data of two participants with the aligned features is horizontally partitioned for HFL. A cuboid with a red border represents the training data required in learning. Especially, a row corresponds to complete data features collected from a sampling ID. Columns correspond to different sampling IDs. The overlapping part means there can be more than one participant sampling the same ID. In addition, HFL is also known as feature-aligned FL, sample-partitioned FL, or example-partitioned FL. Formally, the conditions for HFL can be summarized as

$$\mathcal{X}_i = \mathcal{X}_j, \mathcal{Y}_i = \mathcal{Y}_j, \mathcal{I}_i \neq \mathcal{I}_j, \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j,$$

where $\mathcal{D}_i$ and $\mathcal{D}_j$ denote the datasets of participant i and participant j respectively. In both datasets, the feature space $\mathcal{X}$ and label space $\mathcal{Y}$ are assumed to be the same, but the sampling ID space $\mathcal{I}$ is assumed to be different. The objective of HFL is to increase the amount of data with similar features, while keeping the original data from being transmitted, thus improving the performance of the training model. Participants can still perform feature extraction and classification if new samples appear. HFL can be applied in various fields because it benefits from privacy protection and experience sharing[15]. For example, regional hospitals may receive different patients, and the clinical manifestations of patients with the same disease are similar. It is imperative to protect the patient's privacy, so data about patients cannot be shared. HFL provides a good way to jointly build a ML model for identifying diseases between hospitals.

VFL refers to the case where different participants with various targets usually have datasets that have different

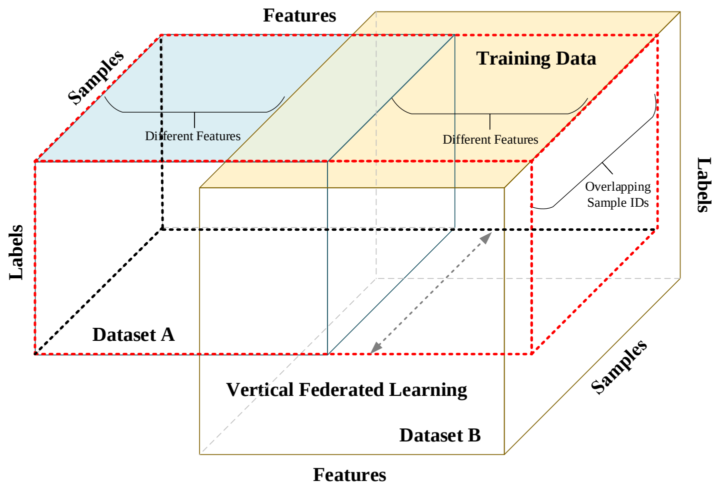

Figure 4. Illustration of vertical federated learning.

feature spaces, but those participants may serve a large number of common users. The heterogeneous feature spaces of distributed datasets can be used to build more general and accurate models without releasing the private data. The word "vertical" derives from the term "vertical partition", which is also widely used in reference to the traditional tabular view. Different from HFL, the training data of each participant are divided vertically. Figure 4 shows an example of VFL in a two-party scenario. The important step in VFL is to align samples, *i.e.*, determine which samples are common to the participants. Although the features of the data are different, the sampled identity can be verified with the same ID. Therefore, VFL is also called sample-aligned FL or feature-partitioned FL. Multiple features are vertically divided into one or more columns. The common samples exposed to different participants can be marked by different labels. The formal definition of VFL's applicable scenario is given.

$$\mathcal{X}_i \neq \mathcal{X}_j, \mathcal{Y}_i \neq \mathcal{Y}_j, \mathcal{I}_i = \mathcal{I}_j, \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j,$$

where $\mathcal{D}_i$ and $\mathcal{D}_i$ represent the dataset held by different participants, and the data feature space pair $(\mathcal{X}_i, \mathcal{X}_j)$ and label space pair $(\mathcal{Y}_i, \mathcal{Y}_j)$ are assumed to be different. The sample ID space $\mathcal{I}_i$ and $\mathcal{I}_j$ are assumed to be the same. It is the objective of VFL to collaborate in building a shared ML model by exploiting all features collected by each participant. The fusion and analysis of existing features can even infer new features. An example of the application of VFL is the evaluation of trust. Banks and e-commerce companies can create a ML model for trust evaluation for users. The credit card record held at the bank and the purchasing history held at the e-commerce company for the set of same users can be used as training data to improve the evaluation model.

FTL applies to a more general case where the datasets of participants are not aligned with each other in terms of samples or features. FTL involves finding the invariant between a resource-rich source domain and a resource-scarce target domain, and exploiting that invariant to transfer knowledge. In comparison with traditional transfer learning[16], FTL focuses on privacy-preserving issues and addresses distributed challenges. An example of FTL is shown in Figure 5. The training data required by FTL may include all data owned by multiply parties for comprehensive information extraction. In order to predict labels for unlabeled new samples, a prediction model is built using additional feature representations for mixed samples from participants A and B. More formally, FTL is applicable for the following scenarios:

$$\mathcal{X}_i \neq \mathcal{X}_j, \mathcal{Y}_i \neq \mathcal{Y}_j, \mathcal{I}_i \neq \mathcal{I}_j, \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j,$$

In datasets $\mathcal{D}_i$ and $\mathcal{D}_j$, there is no duplication or similarity in terms of features, labels and samples. The objective of FTL is to generate as accurate a label prediction as possible for newly incoming samples or unlabeled samples already present. Another benefit of FTL is that it is capable of overcoming the absence of data or labels.

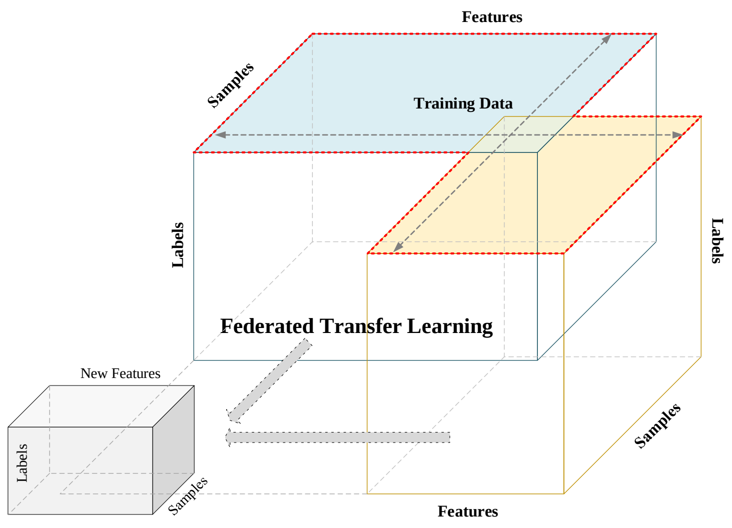

Figure 5. Illustration of federated transfer learning.

For example, a bank and an e-commerce company in two different countries want to build a shared ML model for user risk assessment. In light of geographical restrictions, the user groups of these two organizations have limited overlap. Due to the fact that businesses are different, only a small number of data features are the same. It is important in this case to introduce FTL to solve the problem of small unilateral data and fewer sample labels, and improve the model performance.

3. REINFORCEMENT LEARNING

3.1. Reinforcement learning definition and basics

Generally, the field of ML includes supervised learning, unsupervised learning, RL, *etc*[17]. While supervised and unsupervised learning attempt to make the agent copy the data set, *i.e*., learning from the pre-provided samples, RL is to make the agent gradually stronger in the interaction with the environment, *i.e*., generating samples to learn by itself[18]. RL is a very hot research direction in the field of ML in recent years, which has made great progress in many applications, such as IoT[19–22], autonomous driving[23,24], and game design[25]. For example, the AlphaGo program developed by DeepMind is a good example to reflect the thinking of RL[26]. The agent gradually accumulates the intelligent judgment on the sub-environment of each move by playing game by game with different opponents, so as to continuously improve its level.

The RL problem can be defined as a model of the agent-environment interaction, which is represented in Figure 6. The basic model of RL contains several important concepts, *i.e*.,

- **Environment** and **agent**: Agents are a part of a RL model that exists in an external environment, such as the player in the environment of chess. Agents can improve their behavior by interacting with the environment. Specifically, they take a series of actions to the environment through a set of policies and expect to get a high payoff or achieve a certain goal.
- **Time step**: The whole process of RL can be discretized into different time steps. At every time step, the environment and the agent interact accordingly.
- **State**: The state reflects agents' observations of the environment. When agents take action, the state will also change. In other words, the environment will move to the next state.
- **Actions**: Agents can assess the environment, make decisions and finally take certain actions. These actions are imposed on the environment.
- **Reward**: After receiving the action of the agent, the environment will give the agent the state of the current

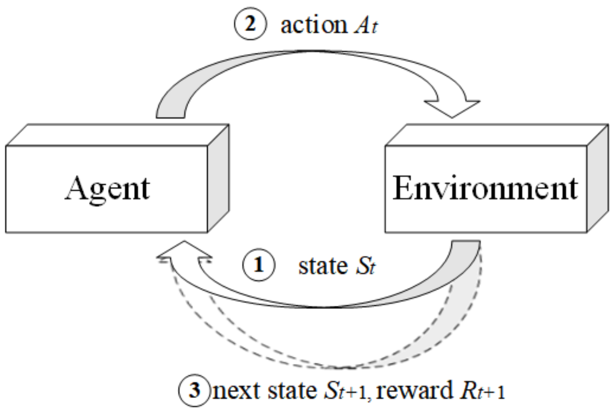

Figure 6. The agent-environment interaction of the basic reinforcement learning model.

environment and the reward due to the previous action. Reward represents an assessment of the action taken by agents.

More formally, we assume that there are a series of time steps $t = 0,1,2,...$ in a basic RL model. At a certain time step t, the agent will receive a state signal S_t of the environment. In each step, the agent will select one of the actions allowed by the state to take an action A_t. After the environment receives the action signal A_t, the environment will feed back to the agent the corresponding status signal S_{t+1} at the next step $t + 1$ and the immediate reward R_{t+1}. The set of all possible states, *i.e.*, the state space, is denoted as $\mathcal{S}$. Similarly, the action space is denoted as $\mathcal{A}$. Since our goal is to maximize the total reward, we can quantify this total reward, usually referred to as return with

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T,$$

where T is the last step, *i.e.*, S_T as the termination state. An **episode** is completed when the agent completes the termination action.

In addition to this type of episodic task, there is another type of task that does not have a termination state, in other words, it can in principle run forever. This type of task is called a continuing task. For continuous tasks, since there is no termination state, the above definition of return may be divergent. Thus, another way to calculate return is introduced, which is called discounted return, *i.e.*,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where the discount factor γ satisfies $0 \leqslant \gamma \leqslant 1$. When $\gamma = 1$, the agent can obtain the full value of all future steps, while when $\gamma = 0$, the agent can only see the current reward. As γ changes from 0 to 1, the agent will gradually become forward-looking, looking not only at current interests, but also for its own future.

The value function is the agent's prediction of future rewards, which is used to evaluate the quality of the state and select actions. The difference between the value function and rewards is that the latter is defined as evaluating an immediate sense for interaction while the former is defined as the average return of actions over a long period of time. In other words, the value function of the current state $S_t = s$ is its long-term expected return. There are two significant value functions in the field of RL, *i.e.*, state value function $V_\pi(s)$ and action value function $Q_\pi(s, a)$. The function $V_\pi(s)$ represents the expected return obtained if the agent continues to follow strategy π all the time after reaching a certain state S_t, while the function $Q_\pi(s, a)$ represents the expected return obtained if action $A_t = a$ is taken after reaching the current state $S_t = s$ and the following actions are taken according to the strategy π. The two functions are specifically defined as follows, *i.e.*,

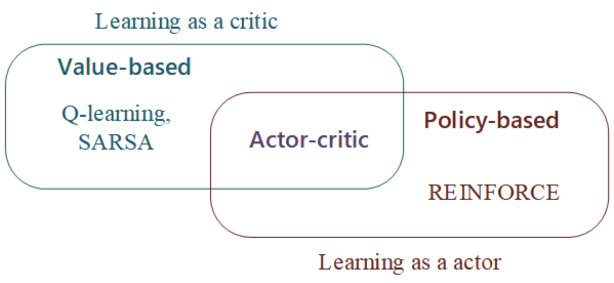

Figure 7. The categories and representative algorithms of reinforcement learning.

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s], \forall s \in \mathcal{S}$$

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a], \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

The results of RL are action decisions, called as the policy. The policy gives agents the action a that should be taken for each state s. It is noted as $=\pi(A_t = a|S_t = s)$, which represents the probability of taking action $A_t = a$ in state $S_t = s$. The goal of RL is to learn the optimal policy that can maximize the value function by interacting with the environment. Our purpose is not to get the maximum reward after a single action in the short term, but to get more reward in the long term. Therefore, the policy can be figured out as,

$$\pi^* = arg\max_\pi V_\pi(s), \forall s \in \mathcal{S}.$$

3.2. Categories of reinforcement learning

In RL, there are several categories of algorithms. One is value-based and the other is policy-based. In addition, there is also an actor-critic algorithm that can be obtained by combining the two, as shown in Figure 7.

3.2.1. Value-based methods

Recursively expand the formulas of the action value function, the corresponding Bellman equation is obtained, which describes the recursive relationship between the value function of the current state and subsequent state. The recursive expansion formula of the action value function $Q_\pi(s,a)$ is

$$Q_\pi(s,a) = \sum_{s',r} p\left(s',r|s,a\right)\left[r + \gamma \sum_{a'} \pi\left(a'|s'\right) Q_\pi\left(s',a'\right)\right],$$

where the function $p\left(s',r|s,a\right) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$ defines the trajectory probability to characterize the environment's dynamics. $R_t = r$ indicates the reward obtained by the agent taking action $A_{t-1} = a$ in state $S_{t-1} = s$. Besides, $S_t = s'$ and $A_t = a'$ respectively represent the state and the action taken by the agent at the next moment t.

In the value-based algorithms, the above value function $Q_\pi(s,a)$ is calculated iteratively, and the strategy is then improved based on this value function. If the value of every action in a given state is known, the agent can select an action to perform. In this way, if the optimal $Q_\pi(s, a = a^*)$ can be figured out, the best action a^* will be found. There are many classical value-based algorithms, including Q-learning[27], state–action–reward–state–action (SARSA)[28], *etc*.

Q-learning is a typical widely-used value-based RL algorithm. It is also a model-free algorithm, which means that it does not need to know the model of the environment but directly estimates the Q value of each executed action in each encountered state through interacting with the environment[27]. Then, the optimal strategy is formulated by selecting the action with the highest Q value in each state. This strategy maximizes the expected return for all subsequent actions from the current state. The most important part of Q-learning is the update of Q value. It uses a table, *i.e.*, Q-table, to store all Q value functions. Q-table uses state as row and action as column. Each (s, a) pair corresponds to a Q value, *i.e.*, $Q(s, a)$, in the Q-table, which is updated as follows,

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q\left(s^{'}, a^{'}\right) - Q(s, a) \right]$$

where r is the reward given by taking action a under state s at the current time step. s' and a' indicate the state and the action taken by the agent at the next time step respectively. α is the learning rate to determine how much error needs to be learned, and γ is the attenuation of future reward. If the agent continuously accesses all state-action pairs, the Q-learning algorithm will converge to the optimal Q function. Q-learning is suitable for simple problems, *i.e.*, small state space, or a small number of actions. It has high data utilization and stable convergence.

3.2.2. Policy-based methods

The above value-based method is an indirect approach to policy selection, and has trouble handling an infinite number of actions. Therefore, we want to be able to model the policy directly. Different from the value-based method, the policy-based algorithm does not need to estimate the value function, but directly fits the policy function, updates the policy parameters through training, and directly generates the best policy. In policy-based methods, we input a state and output the corresponding action directly, rather than the value $V(s)$ or Q value $Q(s, a)$ of the state. One of the most representative algorithms is strategy gradient, which is also the most basic policy-based algorithm.

Policy gradient chooses to optimize the policy directly and update the parameters of the policy network by calculating the gradient of expected reward[29]. Therefore, its objective function $J(\theta)$ is directly designed as expected cumulative rewards, *i.e.*,

$$J(\theta) = \mathbb{E}_{\tau\ \_\theta(\tau)}[r(\tau)] = \int_{\tau\ \pi(\tau)} r(\tau)\,\pi_\theta(\tau)\,d\tau \quad .$$

By taking the derivative of $J(\theta)$, we get

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\ \pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (A_t|S_t) \sum_{t=1}^{T} r(S_t, A_t)\right].$$

The above formula consists of two parts. One is $\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (A_t|S_t)$ which denotes the probability of the gradient in the current trace. The other is $\sum_{t=1}^{T} r(S_t, A_t)$ which represents the return of the current trace. Since the return is total rewards and can only be obtained after one episode, the policy gradient algorithm can only be updated for each episode, not for each time step.

The expected value can be expressed in a variety of ways, corresponding to different ways of calculating the loss function. The advantage of the strategy gradient algorithm is that it can be applied in the continuous action space. In addition, the change of the action probability is smoother, and the convergence is better guaranteed.

REINFORCE algorithm is a classic policy gradient algorithm[30]. Since the expected value of the cumulative reward cannot be calculated directly, the Monte Carlo method is applied to approximate the average value of multiple samples. REINFORCE updates the unbiased estimate of the gradient by using Monte Carlo sampling.

Each sampling generates a trajectory, which runs iteratively. After obtaining a large number of trajectories, the cumulative reward can be calculated by using certain transformations and approximations as the loss function for gradient update. However, the variance of this algorithm is large since it needs to interact with the environment until the terminate state. The reward for each interaction is a random variable, so each variance will add up when the variance is calculated. In particular, the REINFORCE algorithm has three steps:

- Step 1: sample τ_i from $\pi_\theta\left(A_t|S_t\right)$
- Step 2: $\nabla_\theta J\left(\theta\right) \approx \sum_i \left[\sum_{t=1}^T \nabla_\theta \log \pi_\theta\left(A_t^i|S_t^i\right) \sum_{t=1}^T r\left(S_t^i, A_t^i\right)\right]$
- Step 3: $theta \leftarrow \theta + \alpha \nabla_\theta J\left(\theta\right)$

The two algorithms, value-based and policy-based methods, both have their own characteristics and disadvantages. Firstly, the disadvantages of the value-based methods are that the output of the action cannot be obtained directly, and it is difficult to extend to the continuous action space. The value-based methods can also lead to the problem of high bias, *i.e.*, it is difficult to eliminate the error between the estimated value function and the actual value function. For the policy-based methods, a large number of trajectories must be sampled, and the difference between each trajectory may be huge. As a result, high variance and large gradient noise are introduced. It leads to the instability of training and the difficulty of policy convergence.

3.2.3. Actor-critic methods

The actor-critic architecture combines the characteristics of the value-based and policy-based algorithms, and to a certain extent solves their respective weaknesses, as well as the contradictions between high variance and high bias. The constructed agent can not only directly output policies, but also evaluate the performance of the current policies through the value function. Specifically, the actor-critic architecture consists of an actor which is responsible for generating the policy and a critic to evaluate this policy. When the actor is performing, the critic should evaluate its performance, both of which are constantly being updated[31]. This complementary training is generally more effective than a policy-based method or value-based method.

In specific, the input of actor is state S_t, and the output is action A_t. The role of actor is to approximate the policy model $\pi_\theta\left(A_t|S_t\right)$. Critic uses the value function Q as the output to evaluate the value of the policy, and this Q value $Q\left(S_t, A_t\right)$ can be directly applied to calculate the loss function of actor. The gradient of the expected revenue function $J\left(\theta\right)$ in the action-critic framework is developed from the basic policy gradient algorithm. The policy gradient algorithm can only implement the update of each episode, and it is difficult to accurately feedback the reward. Therefore, it has poor training efficiency. Instead, the actor-critic algorithm replaces $\sum_{t=1}^T r\left(S_t^i, A_t^i\right)$ with $Q\left(S_t, A_t\right)$ to evaluate the expected returns of state-action tuple $\{S_t, A_t\}$ in the current time step t. The gradient of $J\left(\theta\right)$ can be expressed as

$$\nabla_\theta J\left(\theta\right) = \mathbb{E}_{\tau\ \pi_\theta(\tau)}\left[\sum_{t=1}^T \nabla_\theta \log \pi_\theta\left(A_t|S_t\right) Q\left(S_t, A_t\right)\right].$$

3.3. Deep reinforcement learning

With the continuous expansion of the application of deep learning, its wave also swept into the RL field, resulting in deep reinforcement learning (DRL), *i.e.*, using a multi-layer deep neural network to approximate value function or policy function in the RL algorithm[32,33]. DRL mainly solves the curse-of-dimensionality problem in real-world RL applications with large or continuous state and/or action space, where the traditional tabular RL algorithms cannot store and extract a large amount of feature information[17,34].

Q-learning, as a very classical algorithm in RL, is a good example to understand the purpose of DRL. The big issue with Q-learning falls into the tabular method, which means that when state and action spaces are very large, it cannot build a very large Q table to store a large number of Q values[35]. Besides, it counts and iterates

Table 1. Taxonomy of representative algorithms for DRL.

Types		Representative algorithms
Value-based		Deep Q-Network (DQN)[37], Double Deep Q-Network (DDQN)[39], DDQN with proportional prioritization[40]
Policy-based		REINFORCE[30], Q-prop[41]
Actor-critic		Soft Actor-Critic (SAC)[42], Asynchronous Advantage Actor Critic (A3C)[43], Deep Deterministic Policy Gradient (DDPG)[44], Distributed Distributional Deep Deterministic Policy Radients (D4PG)[45], Twin Delayed Deep Deterministic (TD3)[46], Trust Region Policy Optimization (TRPO)[47], Proximal Policy Optimization (PPO)[48]
Advanced	POMDP	Deep Belief Q-Network (DBQN)[49], Deep Recurrent Q-Network (DRQN)[50], Recurrent Deterministic Policy Gradients (RDPG)[51]
	Multi-agents	Multi-Agent Importance Sampling (MAIS)[52], Coordinated Multi-agent DQN[53], Multi-agent Fingerprints (MAF)[52], Counterfactual Multiagent Policy Gradient (COMAPG)[54], Multi-Agent DDPG (MADDPG)[55]

Q values based on past states. Therefore, on the one hand, the applicable state and action space of Q-learning is very small. On the other hand, if a state never appears, Q-learning cannot deal with it[36]. In other words, Q-learning has no prediction ability and generalization ability at this point.

In order to make Q-learning with prediction ability, considering that neural network can extract feature information well, deep Q network (DQN) is proposed by applying deep neural network to simulate Q value function. In specific, DQN is the continuation of Q-learning algorithm in continuous or large state space to approximate Q value function by replacing Q table with neural networks[37].

In addition to the value-based DRL algorithm such as DQN, we summarize a variety of classical DRL algorithms according to algorithm types by referring to some DRL related surveys[38] in Table 1, including not only the policy-based and actor-critic DRL algorithms, but also the advanced DRL algorithms of partially observable markov decision process (POMDP) and multi-agents.

4. FEDERATED REINFORCEMENT LEARNING

In this section, the detailed background and categories of FRL will be discussed.

4.1. Federated reinforcement learning background

Despite the excellent performance that RL and DRL have achieved in many areas, they still face several important technical and non-technical challenges in solving real-world problems. The successful application of FL in supervised learning tasks arouses interest in exploiting similar ideas in RL, *i.e.*, FRL. On the other hand, although FL is useful in some specific situations, it fails to deal with cooperative control and optimal decision-making in dynamic environments[10]. FRL not only provides the experience for agents to learn to make good decisions in an unknown environment, but also ensures that the privately collected data during the agent's exploration does not have to be shared with others. A forward-looking and interesting research direction is how to conduct RL under the premise of protecting privacy. Therefore, it is proposed to use FL framework to enhance the security of RL and define FRL as a security-enhanced distributed RL framework to accelerate the learning process, protect agent privacy and handle not independent and identically distributed (Non-IID) data[8]. Apart from improving the security and privacy of RL, we believe that FRL has a wider and larger potential in helping RL to achieve better performance in various aspects, which will be elaborated in the following subsections.

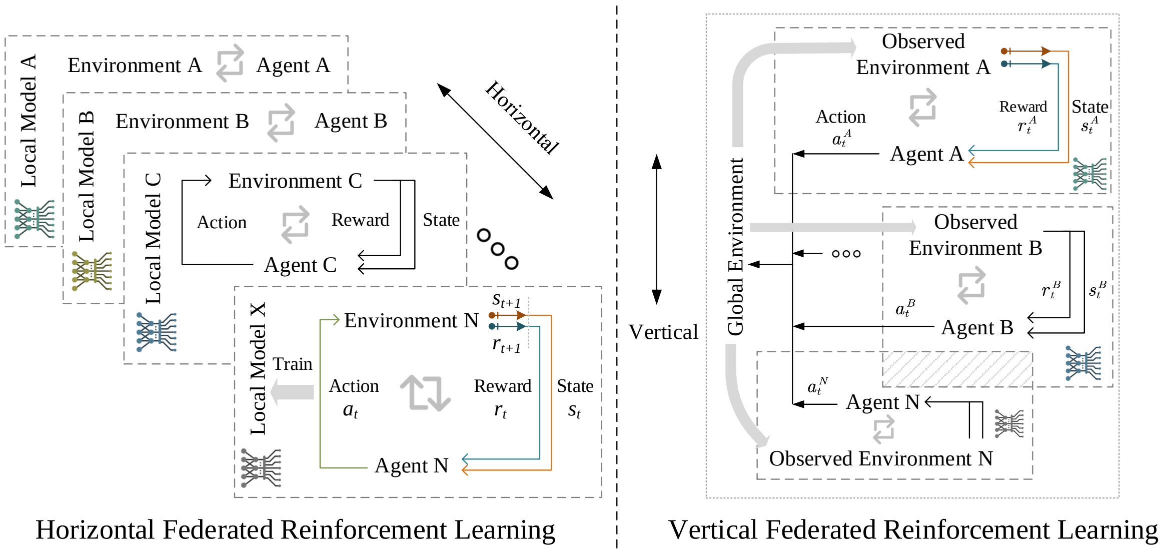

Figure 8. Comparison of horizontal federated reinforcement learning and vertical federated reinforcement learning.

In order to facilitate understanding and maintain consistency with FL, FRL is divided into two categories depending on environment partition[7], *i.e.*, HFRL and VFRL. Figure 8 gives the comparison between HFRL and VFRL. In HFRL, the environment that each agent interacts with is independent of the others, while the state space and action space of different agents are aligned to solve similar problems. The action of each agent only affects its own environment and results in corresponding rewards. As an agent can hardly explore all states of its environment, multiple agents interacting with their own copy of the environment can accelerate training and improve model performance by sharing experience. Therefore, horizontal agents use server-client model or peer-to-peer model to transmit and exchange the gradients or parameters of their policy models (actors) and/or value function models (critics). In VFRL, multiple agents interact with the same global environment, but each can only observe limited state information in the scope of its view. Agents can perform different actions depending on the observed environment and receive local reward or even no reward. Based on the actual scenario, there may be some observation overlap between agents. In addition, all agents' actions affect the global environment dynamics and total rewards. As opposed to the horizontal arrangement of independent environments in HFRL, the vertical arrangement of observations in VFRL poses a more complex problem and is less studied in the existing literature.

4.2. Horizontal federated reinforcement learning

HFRL can be applied in scenarios in which the agents may be distributed geographically, but they face similar decision-making tasks and have very little interaction with each other in the observed environments. Each participating agent independently executes decision-making actions based on the current state of environment and obtains positive or negative rewards for evaluation. Since the environment explored by one agent is limited and each agent is unwilling to share the collected data, multiple agents try to train the policy and/or value model together to improve model performance and increase learning efficiency. The purpose of HFRL is to alleviate the sample-efficiency problem in RL, and help each agent quickly obtain the optimal policy which can maximize the expected cumulative reward for specific tasks, while considering privacy protection.

In the HFRL problem, the environment, state space, and action space can replace the data set, feature space, and label space of basic FL. More formally, we assume that N agents $\{\mathcal{F}_i\}_{i=1}^N$ can observe the environment $\{\mathcal{E}_i\}_{i=1}^N$ within their field of vision. $\mathcal{G}$ denotes the collection of all environments. The environment $\mathcal{E}_i$ where the i-th agent is located has a similar model, *i.e.*, state transition probability and reward function compared to other environments. Note that the environment $\mathcal{E}_i$ is independent of the other environments, in that the state

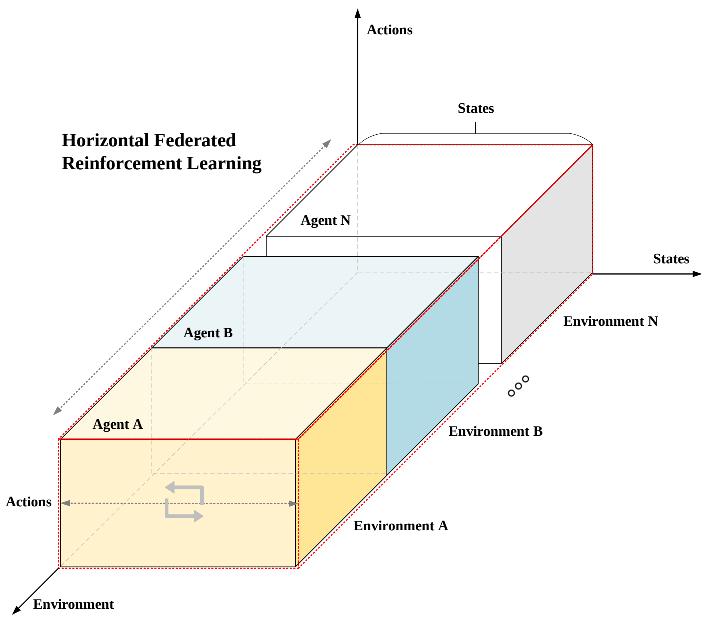

Figure 9. Illustration of horizontal federated reinforcement learning.

transition and reward model of $\mathcal{E}_i$ do not depend on the states and actions of the other environments. Each agent $\mathcal{F}_i$ interacts with its own environment $\mathcal{E}_i$ to learn an optimal policy. Therefore, the conditions for HFRL are presented as follows, *i.e.*,

$$\mathcal{S}_i = \mathcal{S}_j, \mathcal{A}_i = \mathcal{A}_j, \mathcal{E}_i \neq \mathcal{E}_j, \forall i, j \in \{1,2,...,N\}, \mathcal{E}_i, \mathcal{E}_j \in \mathcal{G}, i \neq j,$$

where $\mathcal{S}_i$ and $\mathcal{S}_j$ denote the similar state space encountered by the i-th agent and j-th agent, respectively. $\mathcal{A}_i$ and $\mathcal{A}_j$ denote the similar action space of the i-th agent and j-th agent, respectively The observed environment $\mathcal{E}_i$ and $\mathcal{E}_j$ are two different environments that are assumed to be independent and ideally identically distributed.

Figure 9 shows the HFRL in graphic form. Each agent is represented by a cuboid. The axes of the cuboid denote three dimensions of information, *i.e.*, the environment, state space, and action space. We can intuitively see that all environments are arranged horizontally, and multiple agents have aligned state and action spaces. In other words, each agent explores independently in its respective environment, and needs to obtain optimal strategies for similar tasks. In HFRL, agents share their experiences by exchanging masked models to enhance sample efficiency and accelerate the learning process.

A typical example of HFRL is the autonomous driving system in IoV. As vehicles drive on roads throughout the city and country, they can collect various environmental information and train the autonomous driving models locally. Due to driving regulations, weather conditions, driving routes, and other factors, one vehicle cannot be exposed to every possible situation in the environment. Moreover, the vehicles have basically the same operations, including braking, acceleration, steering, *etc*. Therefore, vehicles driving on different roads, different cities, or even different countries could share their learned experience with each other by FRL without revealing their driving data according to the premise of privacy protection. In this case, even if other vehicles have never encountered a situation, they can still perform the best action by using the shared model. The exploration of multiple vehicles together also creates an increased chance of learning rare cases to ensure the reliability of the model.

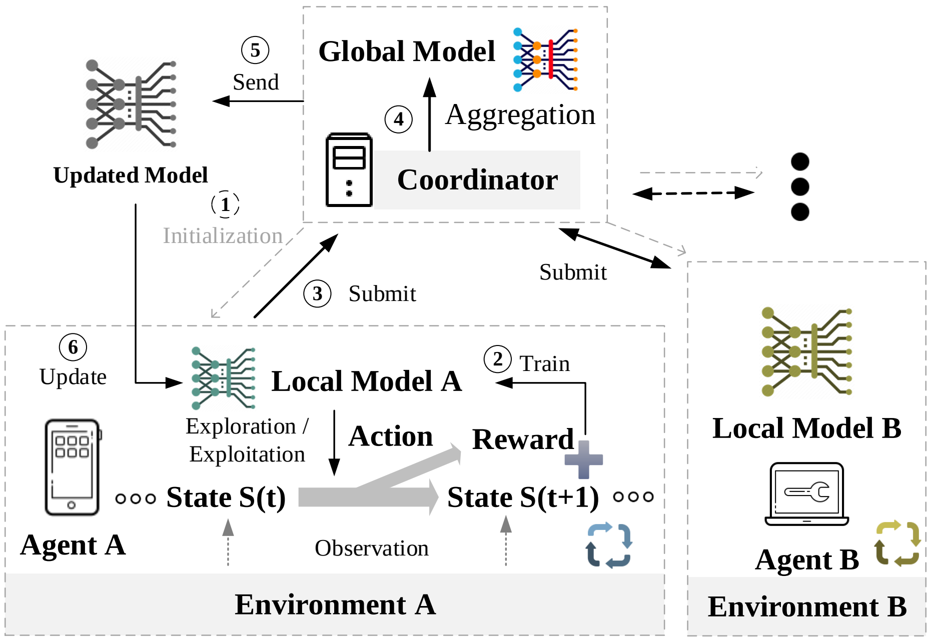

Figure 10. An example of horizontal federated reinforcement learning architecture.

For a better understanding of HFRL, Figure 10 shows an example of HFRL architecture using the server-client model. The coordinator is responsible for establishing encrypted communication with agents and implementing aggregation of shared models. The multiple parallel agents may be composed of heterogeneous equipment (*e.g.*, IoT devices, smart phone and computers, *etc.*) and distributed geographically. It is worth noting that there is no specific requirement for the number of agents, and agents are free to choose to join or leave. The basic procedure for conducting HFRL can be summarized as follows.

- Step 1: The initialization/join process can be divided into two cases, one is when the agent has no model locally, and the other is when the agent has a model locally. For the first case, the agent can directly download the shared global model from a coordinator. For the second case, the agent needs to confirm the model type and parameters with the central coordinator.
- Step 2: Each agent independently observes the state of the environment and determines the private strategy based on the local model. The selected action is evaluated by the next state and received reward. All agents train respective models in state-action-reward-state (SARS) cycles.
- Step 3: Local model parameters are encrypted and transmitted to the coordinator. Agents may submit local models at any time as long as the trigger conditions are met.
- Step 4: The coordinator conducts the specific aggregation algorithm to evolve the global federated model. Actually, there is no need to wait for submissions from all agents, and appropriate aggregation conditions can be formulated depending on communication resources.
- Step 5: The coordinator sends back the aggregated model to the agents.
- Step 6: The agents improve their respective models by fusing the federated model.

Following the above architecture and process, applications suitable for HFRL should meet the following characteristics. First, agents have similar tasks to make decisions under dynamic environments. Different from the FL setting, the goal of the HFRL-based application is to find the optimal strategy to maximize reward in the future. For the agent to accomplish the task requirements, the optimal strategy directs them to perform certain actions, such as control, scheduling, navigation, *etc*. Second, distributed agents maintain independent observations. Each agent can only observe the environment within its field of view, but does not ensure that the collected data follows the same distribution. Third, it is important to protect the data that each agent collects and explores. Agents are presumed to be honest but curious, *i.e*., they honestly follow the learning mechanism but are curious about private information held by other agents. Due to this, the data used for training is only stored at the owner and is not transferred to the coordinator. HFRL provides an implementation method for sharing experiences under the constraints of privacy protection. Additionally, various reasons limit the agent's ability to explore the environment in a balanced manner. Participating agents may include heterogeneous devices. The amount of data collected by each agent is unbalanced due to mobility, observation, energy and other factors. However, all participants have sufficient computing, storage, and communication capabilities. These capabilities assist the agent in completing model training, merging, and other basic processes. Finally, the environment observed by a agent may change dynamically, causing differences in data distribution. The participating agents need to update the model in time to quickly adapt to environmental changes and construct a personalized local model.

In existing RL studies, some applications that meet the above characteristics can be classified as HFRL. Nadiger *et al*.[56] presents a typical HFRL architecture, which includes the grouping policy, the learning policy, and the federation policy. In this work, RL is used to show the applicability of granular personalization and FL is used to reduce training time. To demonstrate the effectiveness of the proposed architecture, a non-player character in the Atari game Pong is implemented and evaluated. In the study from Liu *et al*.[57], the authors propose the lifelong federated reinforcement learning (LFRL) for navigation in cloud robotic systems. It enables the robot to learn efficiently in a new environment and use prior knowledge to quickly adapt to the changes in the environment. Each robot trains a local model according to its own navigation task, and the centralized cloud server implements a knowledge fusion algorithm for upgrading a shared model. In considering that the local model and the shared model might have different network structures, this paper proposes to apply transfer learning to improve the performance and efficiency of the shared model. Further, researchers also focus on HFRL-based applications in the IoT due to the high demand for privacy protection. Ren *et al*.[58] suggest deploying the FL architecture between edge nodes and IoT devices for computation offloading tasks. IoT devices can download RL model from edge nodes and train the local model using own data, including the remained energy resources and the workload of IoT device, *etc*. The edge node aggregates the updated private model into the shared model. Although this method considers privacy protection issues, it requires further evaluation regarding the cost of communication resources by the model exchange. In addition, the work[59] proposes a federated deep-reinforcement-learning-based framework (FADE) for edge caching. Edge devices, including base stations (BSs), can cooperatively learn a predictive model using the first round of training parameters for local learning, and then upload the local parameters tuned to the next round of global training. By keeping the training on local devices, the FADE can enable fast training and decouple the learning process between the cloud and data owner in a distributed-centralized manner. More HFRL-based applications will be classified and summarized in the next section.

Prior to HFRL, a variety of distributed RL algorithms have been extensively investigated, which are closely related to HFRL. In general, distributed RL algorithms can be divided into two types: synchronized and asynchronous. In synchronous RL algorithms, such as Sync-Opt synchronous stochastic optimization (Sync-Opt)[60] and parallel advantage actor critic (PAAC)[3], the agents explore their own environments separately, and after a number of samples are collected, the global parameters are updated synchronously. On the contrary, the coordinator will update the global model immediately after receiving the gradient from an arbitrary agent in asynchronous RL algorithms, rather than waiting for other agents. Several asynchronous RL algorithms are presented, including A3C[61], Impala[62], Ape-X[63] and general reinforcement learning architecture (Gorila)[1]. From the perspective of technology development, HFRL can also be considered security-enhanced parallel RL. In parallel RL, multiple agents interact with a stochastic environment to seek the optimal policy for the same task[1,2]. By building a closed loop of data and knowledge in parallel systems, parallel RL helps determine the next course of action for each agent. The state and action representations are fed into a designed neural network to approximate the action value function[64]. However, parallel RL typically transfers the experience of agent without considering privacy protection issues[7]. In the implementation of HFRL, further restrictions accompany privacy protection and communication consumption to adapt to special scenarios, such as IoT applications[59]. In addition, another point to consider is Non-IID data. In order to ensure convergence of the RL model, it is generally assumed in parallel RL that the states transitions in the environment follow the same distribution, *i.e*., the environments of different agents are IID. But in actual scenarios, the situation faced by agents may differ slightly, so that the models of environments for different agents are not identically distributed. Therefore, HFRL needs to improve the generalization ability of the model compared with parallel RL to meet the challenges posed by Non-IID data.

Based on the potential issues faced by the current RL technology, the advantages of HFRL can be summarized as follows.

- Enhancing training speed. In the case of a similar target task, multiple agents sharing training experiences gained from different environments can expedite the learning process. The local model rapidly evolves through aggregation and update algorithms to assess the unexplored environment. Moreover, the data obtained by different agents are independent, reducing correlations between the observed data. Furthermore, this also helps to solve the issue of unbalanced data caused by various restrictions.
- Improving the reliability of model. When the dimensions of the state of the environment are enormous or even uncountable, it is difficult for a single agent to train an optimal strategy for situations with extremely low occurrence probabilities. Horizontal agents are exploring independently while building a cooperative model to improve the local model's performance on rare states.
- Mitigating the problems of devices heterogeneity. Different devices deploying RL agents in the HFRL architecture may have different computational and communication capabilities. Some devices may not meet the basic requirements for training, but strategies are needed to guide actions. HFRL makes it possible for all agents to obtain the shared model equally for the target task.
- Addressing the issue of non-identical environment. Considering the differences in the environment dynamics for the different agents, the assumption of IID data may be broken. Under the HFRL architecture, agents in not identically-distributed environment models can still cooperate to learn a federated model. In order to address the difference in environment dynamics, a personalized update algorithm of local model could be designed to minimize the impact of this issue.
- Increasing the flexibility of the system. The agent can decide when to participate in the cooperative system at any time, because HFRL allows asynchronous requests and aggregation of shared models. In the existing HFRL-based application, new agents also can apply for membership and benefit from downloading the shared model.

4.3. Vertical federated reinforcement learning

In VFL, samples of multiple data sets have different feature spaces but these samples may belong to the same groups or common users. The training data of each participant are divided vertically according to their features. More general and accurate models can be generated by building heterogeneous feature spaces without releasing private information. VFRL applies the methodology of VFL to RL and is suitable for POMDP scenarios where different RL agents are in the same environment but have different interactions with the environment. Specifically, different agents could have different observations that are only part of the global state. They could take actions from different action spaces and observe different rewards, or some agents even take no actions or cannot observe any rewards. Since the observation range of a single agent to the environment is limited, multiple agents cooperate to collect enough knowledge needed for decision making. The role of FL in VFRL is to aggregate the partial features observed by various agents. Especially for those agents without rewards, the aggregation effect of FL greatly enhances the value of such agents in their interactions with the environment, and ultimately helps with the strategy optimization. It is worth noting that in VFRL the issue of privacy protection needs to be considered, *i.e*., private data collected by some agents do not have to be shared with others. Instead, agents can transmit encrypted model parameters, gradients, or direct mid-product to each other. In

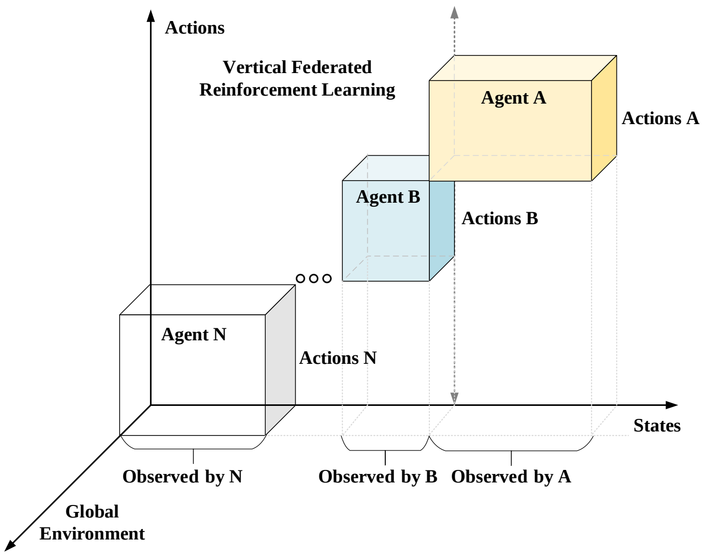

Figure 11. Illustration of vertical federated reinforcement learning.

short, the goal of VFRL is for agents interacting with the same environment to improve the performance of their policies and the effectiveness in learning them by sharing experiences without compromising the privacy.

More formally, we denote $\{\mathcal{F}_i\}_{i=1}^N$ as N agents in VFRL, which interact with a global environment $\mathcal{E}$. The i-th agent $\mathcal{F}_i$ is located in the environment $\mathcal{E}_i = \mathcal{E}$, obtains the local partial observation $\mathcal{O}_i$, and can perform the set of actions $\mathcal{A}_i$. Different from HFRL, the state/observation and action spaces of two agents $\mathcal{F}_i$ and $\mathcal{F}_j$ may be not identical, but the aggregation of the state/observation spaces and action spaces of all the agents constitutes the global state and action spaces of the global environment $\mathcal{E}$. The conditions for VFRL can be defined as *i.e.*,

$$\mathcal{O}_i \neq \mathcal{O}_j, \mathcal{A}_i \neq \mathcal{A}_j, \mathcal{E}_i = \mathcal{E}_j = \mathcal{E}, \bigcup_{i=1}^{N} \mathcal{O}_i = \mathcal{S}, \bigcup_{i=1}^{N} \mathcal{A}_i = \mathcal{A}, \forall i, j \in \{1,2,...,N\}, i \neq j,$$

where $\mathcal{S}$ and $\mathcal{A}$ denote the global state space and action space of all participant agents respectively. It can be seen that all the observations of the N agents together constitute the global state space $\mathcal{S}$ of the environment $\mathcal{E}$. Besides, the environments $\mathcal{E}_i$ and $\mathcal{E}_j$ are the same environment $\mathcal{E}$. In most cases, there is a great difference between the observations of two agents $\mathcal{F}_i$ and $\mathcal{F}_j$.

Figure 11 shows the architecture of VFRL. The dataset and feature space in VFL are converted to the environment and state space respectively. VFL divides the dataset vertically according to the features of samples, and VFRL divides agents based on the state spaces observed from the global environment. Generally speaking, every agent has its local state which can be different from that of the other agents and the aggregation of these local partial states corresponds to the entire environment state[65]. In addition, after interacting with the environment, agents may generate their local actions which correspond to the labels in VFL.

Two types of agents can be defined for VFRL, *i.e.*, decision-oriented agents and support-oriented agents. Decision-oriented agents $\{\mathcal{F}_i\}_{i=1}^K$ can interact with the environment $\mathcal{E}$ based on their local state $\{\mathcal{S}_i\}_{i=1}^K$ and action $\{\mathcal{A}_i\}_{i=1}^K$. Meanwhile, support-oriented agents $\{\mathcal{F}_i\}_{i=K+1}^N$ take no actions and receive no rewards but only the observations of the environment, *i.e.*, their local states $\{\mathcal{S}_i\}_{i=K+1}^N$. In general, the following six steps, as shown in Figure 12, are the basic procedure for VFRL, *i.e.*,

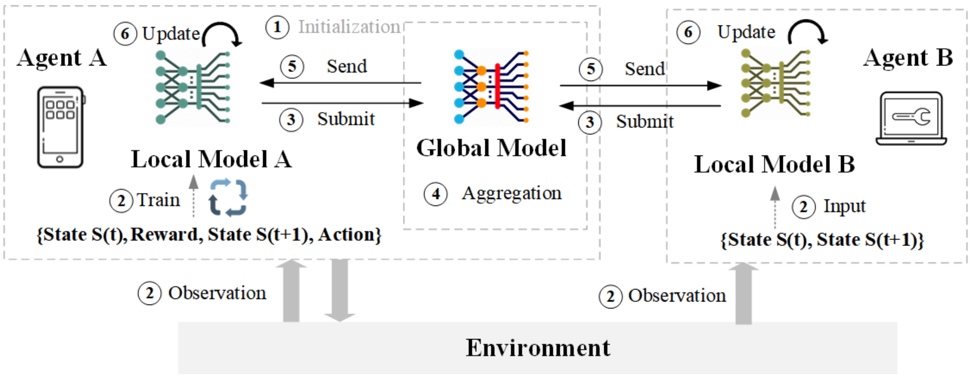

Figure 12. An example of vertical federated reinforcement learning architecture.

- Step 1: Initialization is performed for all agent models.
- Step 2: Agents obtain states from the environment. For decision-oriented agents, actions are obtained based on the local models, and feedbacks are obtained through interactions with the environment, *i.e*., the states of the next time step and rewards. The data tuple of state-action-reward-state (SARS) is used to train the local models.
- Step 3: All agents calculate the mid-products of the local models and then transmit the encrypted mid-products to the federated model.
- Step 4: The federated model performs the aggregation calculation for mid-products and trains the federated model based on the aggregation results.
- Step 5: Federated model encrypts model parameters such as weight and gradient and passes them back to other agents.
- Step 6: All agents update their local models based on the received encrypted parameters.

As an example of VFRL, consider a microgrid (MG) system including household users, the power company, and the photovoltaic (PV) management company as the agents. All the agents observe the same MG environment while their local state spaces are quite different. The global states of the MG system generally consist of several dimensions/features, *i.e*., state-of-charge (SOC) of the batteries, load consumption of the household users, power generation from PV, etc. The household agents can obtain the SOC of their own batteries and their own load consumption, the power company can know the load consumption of all the users, and PV management company can know the power generation of PV. As to the action, the power company needs to make decisions on the power dispatch of the diesel generators (DG), and the household users can make decisions to manage their electrical utilities with demand response. Finally, the power company can observe rewards such as the cost of DG power generation, the balance between power generation and consumption, and the household users can observe rewards such as their electricity bill that is related to their power consumption. In order to learn the optimal policies, these agents need to communicate with each other to share their observations. However, PV managers do not want to expose their data to other companies, and household users also want to keep their consumption data private. In this way, VFRL is suitable to achieve this goal and can help improve policy decisions without exposing specific data.

Compared with HFRL, there are currently few works on VFRL. Zhuo *et al*.[65] present the federated deep reinforcement learning (FedRL) framework. The purpose of this paper is to solve the challenge where the feature space of states is small and the training data are limited. Transfer learning approaches in DRL are also solutions for this case. However, when considering the privacy-aware applications, directly transferring data or models should not be used. Hence, FedRL combines the advantage of FL with RL, which is suitable for the case when agents need to consider their privacy. FedRL framework assumes agents cannot share their partial observations of the environment and some agents are unable to receive rewards. It builds a shared value network, *i.e.*, multiLayer perceptron (MLP), and takes its own Q-network output and encryption value as input to calculate a global Q-network output. Based on the output of global Q-network, the shared value network and self Q-network are updated. Two agents are used in the FedRL algorithm, *i.e.*, agent α and β, which interact with the same environment. However, agent β cannot build its own policies and rewards. Finally, FedRL is applied in two different games, *i.e.*, Grid-World and Text2Action, and achieves better results than the other baselines. Although the VFRL model in this paper only contains two agents, and the structure of the aggregated neural network model is relatively simple, we believe that it is a great attempt to first implement VFRL and verify its effectiveness.

Multi-agent RL (MARL) is very closely related to VFRL. As the name implies, MARL takes into account the existence of multiple agents in the RL system. However, the empirical evaluation shows that applying the simple single-agent RL algorithms directly to scenarios of multiple agents cannot converge to the optimal solution, since the environment is no longer static from the perspective of each agent[66]. In specific, the action of each agent will affect the next state, thus affecting all agents in the future time step[67]. Besides, the actions performed by one certain agent will yield different rewards depending on the actions taken by other agents. This means that agents in MARL correlate with each other, rather than being independent of each other. This challenge, called as the non-stationarity of the environment, is the main problem to be solved in the development of an efficient MARL algorithm[68].

MARL and VFRL both study the problem of multiple agents learning concurrently how to solve a task by interacting with the same environment[69]. Since MARL and VFRL have a large range of similarities, the review of MARL's related works is a very useful guide to help researchers summarize the research focus and better understand VFRL. There is abundant literature related to MARL. However, most MARL research[70–73] is based on a fully observed markov decision process (MDP), where each agent is assumed to have the global observation of the system state[68]. These MARL algorithms are not applicable to the case of POMDP where the observations of individual agents are often only a part of the overall environment[74]. Partial observability is a crucial consideration for the development of algorithms that can be applied to real-world problems[75]. Since VFRL is mainly oriented towards POMDP scenarios, it is more important to analyze the related works of MARL based on POMDP as the guidance of VFRL.

Agents in the above scenarios partially observe the system state and make decisions at each step to maximize the overall rewards for all agents, which can be formalized as a decentralized partially observable markov decision process (Dec-POMDP)[76]. Optimally addressing a Dec-POMDP model is well known to be a very challenging problem. In the early works, Omidshafiei *et al.*[77] proposes a two-phase MT-MARL approach that concludes the methods of cautiously-optimistic learners for action-value approximation and concurrent experience replay trajectories (CERTs) as the experience replay targeting sample-efficient and stable MARL. The authors also apply the recursive neural network (RNN) to estimate the non-observed state and hysteretic Q-learning to address the problem of non-stationarity in Dec-POMDP. Han *et al.*[78] designs a neural network architecture, IPOMDP-net, which extends QMDP-net planning algorithm[79] to MARL settings under POMDP. Besides, Mao *et al.*[80] introduces the concept of information state embedding to compress agents' histories and proposes an RNN model combining the state embedding. Their method, *i.e.*, embed-then-learn pipeline, is universal since the embedding can be fed into any existing partially observable MARL algorithm as the black-box. In the study from Mao *et al.*[81], the proposed Attention MADDPG (ATT-MADDPG) has several critic networks for various agents under POMDP. A centralized critic is adopted to collect the observations and actions of the teammate agents. Specifically, the attention mechanism is applied to enhance the centralized critic. The final introduced work is from Lee *et al.*[82]. They present an augmenting MARL algorithm based on pretraining to address the challenge in disaster response. It is interesting that they use behavioral cloning (BC), a supervised learning method where agents learn their policy from demonstration samples, as the approach to pretrain the neural network. BC can generate a feasible Dec-POMDP policy from demonstration samples, which offers advantages over plain MARL in terms of solution quality and computation time.

Some MARL algorithms also concentrate on the communication issue of POMDP. In the study from Sukhbaatar *et al.*[83], communication between the agents is performed for a number of rounds before their action is selected. The communication protocol is learned concurrently with the optimal policy. Foerster *et al.*[84] proposes a deep recursive network architecture, *i.e.*, deep distributed recurrent Q-network (DDRQN), to address the communication problem in a multi-agent partially-observable setting. This work makes three fundamental modifications to previous algorithms. The first one is last-action inputs, which let each agent access its previous action as an input for the next time-step. Besides, inter-agent weight sharing allows diverse behavior between agents, as the agents receive different observations and thus evolve in different hidden states. The final one is disabling experience replay, which is because the non-stationarity of the environment renders old experiences obsolete or misleading. Foerster *et al.*[84] considers the communication task of fully cooperative, partially observable, sequential multi-agent decision-making problems. In their system model, each agent can receive a private observation and take actions that affect the environment. In addition, the agent can also communicate with its fellow agents via a discrete limited-bandwidth channel. Despite the partial observability and limited channel capacity, authors achieved the task that the two agents could discover a communication protocol that enables them to coordinate their behavior based on the approach of deep recurrent Q-networks.

While there are some similarities between MARL and VFRL, several important differences have to be paid attention to, *i.e.*,

- VFRL and some MARL algorithms are able to address similar problems, *e.g.*, the issues of POMDP. However, there are differences between the solution ideas between two algorithms. Since VFRL is the product of applying VFL to RL, the FL component of VFRL focuses more on the aggregation of partial features, including states and rewards, observed by different agents since VFRL inception. Security is also an essential issue in VFRL. On the contrary, MARL may arise as the most natural way of adding more than one agent in a RL system[85]. In MARL, agents not only interact with the environment, but also have complex interactive relationships with other agents, which creates a great obstacle to the solution of policy optimization. Therefore, the original intentions of two algorithms are different.
- Two algorithms are slightly different in terms of the structure. The agents in MARL must surely have the reward even some of them may not have their own local actions. However, in some cases, the agents in VFRL are not able to generate a corresponding operation policy, so in these cases, some agents have no actions and rewards[65]. Therefore, VFRL can solve more extensive problems that MARL is not capable of solving.
- Both two algorithms involve the communication problem between agents. In MARL, information such as the states of other agents and model parameters can be directly and freely propagated among agents. During communication, some MARL methods such as DDRQN in the work of Foerster *et al.*[84] consider the previous action as an input for the next time-step state. Weight sharing is also allowed between agents. However, VFRL assumes states cannot be shared among agents. Since these agents do not exchange experience and data directly, VFRL focuses more on security and privacy issues of communication between agents, as well as how to process mid-products transferred by other agents and aggregate federated models.

In summary, as a potential and notable algorithm, VFRL has several advantages as follows, *i.e.*,

- Excellent privacy protection. VFRL inherits the FL algorithm's idea of data privacy protection, so for the task of multiple agents cooperation in the same environment, information interaction can be carried out confidently to enhance the learning efficiency of RL model. In this process, each participant does not have to worry about any leakage of raw real-time data.
- Wide application scenarios. With appropriate knowledge extraction methods, including algorithm design and system modeling, VFRL can solve more real-world problems compared with MARL algorithms. This

is because VFRL can consider some agents that cannot generate rewards into the system model, so as to integrate their partial observation information of the environment based on FL while protecting privacy, train a more robust RL agent, and further improve learning efficiency.

4.4. Other types of FRL

The above HFRL or VFRL algorithms borrow ideas from FL for federation between RL agents. Meanwhile, there are also some existing works on FRL that are less affected by FL. Hence, they do not belong to either HFRL or VFRL, but federation between agents is also implemented.

The study from Hu *et al.*[86] is a typical example, which proposes a reward shaping based general FRL algorithm, called federated reward shaping (FRS). It uses reward shaping to share federated information to improve policy quality and training speed. FRS adopts the server-client architecture. The server includes the federated model, while each client completes its own tasks based on the local model. This algorithm can be combined with different kinds of RL algorithms. However, it should be noted that FRS focuses on reward shaping, this algorithm cannot be used when there is no reward in some agents in VFRL. In addition, FRS performs knowledge aggregation by sharing high-level information such as reward shaping value or embedding between client and server instead of sharing experience or policy directly. The convergence of FRS is also guaranteed since only minor changes are made during the learning process, which is the modification of the reward in the replay buffer.

As another example, Anwar *et al.*[87] achieves federation between agents by smoothing the average weight. This work analyzes the Multi-task FRL algorithms (MT-FedRL) with adversaries. Agents only interact and make observations in their environment, which can be featured by different MDPs. Different from HFRL, the state and action spaces do not need to be the same in these environments. The goal of MT-FedRL is to learn a unified policy, which is jointly optimized across all of the environments. MT-FedRL adopts policy gradient methods for RL. In other words, policy parameter is needed to learn the optimal policy. The server-client architecture is also applied and all agents should share their own information with a centralized server. The role of non-negative smoothing average weights is to achieve a consensus among the agents' parameters. As a result, they can help to incorporate the knowledge from other agents as the process of federation.

5. APPLICATIONS OF FRL

In this section, we provide an extensive discussion of the application of FRL in a variety of tasks, such as edge computing, communications, control optimization, attack detection, *etc*. This section is aimed at enabling readers to understand the applicable scenarios and research status of FRL.

5.1. FRL for edge computing

In recent years, edge equipment, such as BSs and road side units (RSUs), has been equipped with increasingly advanced communication, computing and storage capabilities. As a result, edge computing is proposed to delegating more tasks to edge equipment in order to reduce the communication load and reduce the corresponding delay. However, the issue of privacy protection remains challenging since it may be untrustworthy for the data owner to hand off their private information to a third-party edge server[4]. FRL offers a potential solution for achieving privacy-protected intelligent edge computing, especially in decision-making tasks like caching and offloading. Additionally, the multi-layer processing architecture of edge computing is also suitable for implementing FRL through the server-client model. Therefore, many researchers have focused on applying FRL to edge computing.

The distributed data of large-scale edge computing architecture makes it possible for FRL to provide distributed intelligent solutions to achieve resource optimization at the edge. For mobile edge networks, a potential FRL framework is presented for edge system[88], named as "In-Edge AI", to address optimization of mobile edge computing, caching, and communication problems. The authors also propose some ideas and paradigms for solving these problems by using DRL and Distributed DRL. To carry out dynamic system-level optimization and reduce the unnecessary transmission load, "In-Edge AI" framework takes advantage of the collaboration among edge nodes to exchange learning parameters for better training and inference of models. It has been evaluated that the framework has high performance and relatively low learning overhead, while the mobile communication system is cognitive and adaptive to the environment. The paper provides good prospects for the application of FRL to edge computing, but there are still many challenges to overcome, including the adaptive improvement of the algorithm, and the training time of the model from scratch *etc*.

Edge caching has been considered a promising technique for edge computing to meet the growing demands for next-generation mobile networks and beyond. Addressing the adaptability and collaboration challenges of the dynamic network environment, Wang *et al.*[89] proposes a device-to-device (D2D)-assisted heterogeneous collaborative edge caching framework. User equipment (UE) in a mobile network uses the local DQN model to make node selection and cache replacement decisions based on network status and historical information. In other words, UE decides where to fetch content and which content should be replaced in its cache list. The BS calculates aggregation weights based on the training evaluation indicators from UE. To solve the long-term mixed-integer linear programming problem, the attention-weighted federated deep reinforcement learning (AWFDRL) is presented, which optimizes the aggregation weights to avoid the imbalance of the local model quality and improve the learning efficiency of the DQN. The convergence of the proposed algorithm is verified and simulation results show that the AWFDRL framework can perform well on average delay, hit rate, and offload traffic.

A federated solution for cooperative edge caching management in fog radio access networks (F-RANs) is proposed[90]. Both edge computing and fog computing involve bringing intelligence and processing to the origins of data. The key difference between the two architectures is where the computing node is positioned. A dueling deep Q-network based cooperative edge caching method is proposed to overcome the dimensionality curse of RL problem and improve caching performance. Agents are developed in fog access points (F-APs) and allowed to build a local caching model for optimal caching decisions based on the user content request and the popularity of content. HFRL is applied to aggregate the local models into a global model in the cloud server. The proposed method outperforms three classical content caching methods and two RL algorithms in terms of reducing content request delays and increasing cache hit rates.

For edge-enabled IoT, Majidi *et al.*[91] proposes a dynamic cooperative caching method based on hierarchical federated deep reinforcement learning (HFDRL), which is used to determine which content should be cached or evicted by predicting future user requests. Edge devices that have a strong relationship are grouped into a cluster and one head is selected for this cluster. The BS trains the Q-value based local model by using BS states, content states, and request states. The head has enough processing and caching capabilities to deal with model aggregation in the cluster. By categorizing edge devices hierarchically, HFDRL improves the response time delay to keeps both small and large clusters from experiencing the disadvantages they could encounter. Storage partitioning allows content to be stored in clusters at different levels using the storage space of each device. The simulation results show the proposed method using MovieLens datasets improves the average content access delay and the hit rate.

Considering the low latency requirements and privacy protection issue of IoV, the study of efficient and secure caching methods has attracted many researchers. An FRL-empowered task caching problem with IoV has been analyzed by Zhao *et al.*[92]. The work proposes a novel cooperative caching algorithm (CoCaRL) for vehicular networks with multi-level FRL to dynamically determine which contents should be replaced and where the content requests should be served. This paper develops a two-level aggregation mechanism for federated learning to speed up the convergence rate and reduces communication overhead, while DRL task is employed to optimize the cooperative caching policy between RSUs of vehicular networks. Simulation results show that the proposed algorithm can achieve a high hit rate, good adaptability and fast convergence in a complex environment.

Apart from caching services, FRL has demonstrated its strong ability to facilitate resource allocation in edge computing. In the study from Zhu *et al.*[93], the authors specifically focus on the data offloading task for mobile edge computing (MEC) systems. To achieve joint collaboration, the heterogeneous multi-agent actor-critic (H-MAAC) framework is proposed, in which edge devices independently learn the interactive strategies through their own observations. The problem is formulated as a multi-agent MDP for modeling edge devices' data allocation strategies, *i.e.*, moving the data, locally executing or offloading to a cloud server. The corresponding joint cooperation algorithm that combines the edge federated model with the multi-agent actor-critic RL is also presented. Dual lightweight neural networks are built, comprising original actor/critic networks and target actor/critic networks.

Blockchain technology has also attracted lot attention from researchers in edge computing fields since it is able to provide reliable data management within the massive distributed edge nodes. In the study from Yu *et al.*[94], the intelligent ultra-dense edge computing (I-UDEC) framework is proposed, integrating with blockchain and RL technologies into 5G ultra-dense edge computing networks. In order to achieve low overhead computation offloading decisions and resource allocation strategies, authors design a two-timescale deep reinforcement learning (2Ts-DRL) approach, which consists of a fast-timescale and a slow-timescale learning process. The target model can be trained in a distributed manner via FL architecture, protecting the privacy of edge devices.

Additionally, to deal with the different types of optimization tasks, variants of FRL are being studied. Zhu *et al.*[95] presents a resource allocation method for edge computing systems, called concurrent federated reinforcement learning (CFRL). The edge node continuously receives tasks from serviced IoT devices and stores those tasks in a queue. Depending on its own resource allocation status, the node determines the scheduling strategy so that all tasks are completed as soon as possible. In case the edge host does not have enough available resources for the task, the task can be offloaded to the server. Contrary to the definition of the central server in the basic FRL, the aim of central server in CFRL is to complete the tasks that the edge nodes cannot handle instead of aggregating local models. Therefore, the server needs to train a special resource allocation model based on its own resource status, forwarded tasks and unique rewards. The main idea of CFRL is that edge nodes and the server cooperatively participate in all task processing in order to reduce total computing time and provide a degree of privacy protection.

5.2. FRL for communication networks

In parallel with the continuous evolution of communication technology, a number of heterogeneous communication systems are also being developed to adapt to different scenarios. Many researchers are also working toward intelligent management of communication systems. The traditional ML-based management methods are often inefficient due to their centralized data processing architecture and the risk of privacy leakage[5]. FRL can play an important role in services slicing and access controlling to replace centralized ML methods.

In communication network services, network function virtualization (NFV) is a critical component of achieving scalability and flexibility. Huang *et al.*[96] proposes a novel scalable service function chains orchestration (SSCO) scheme for NFV-enabled networks via FRL. In the work, a federated-learning-based framework for training global learning, along with a time-variant local model exploration, is designed for scalable SFC orchestration. It prevents data sharing among stakeholders and enables quick convergence of the global model. To reduce communication costs, SSCO allows the parameters of local models to be updated just at the beginning and end of each episode through distributed clients and the cloud server. A DRL approach is used to map virtual network functions (VNFs) into networks with local knowledge of resources and instantiation cost. In addition, the authors also propose a loss-weight-based mechanism for generation and exploitation of reference samples for training in replay buffers, avoiding the strong relevance of each sample. Simulation results demonstrate that SSCO can significantly reduce placement errors and improve resource utilization ratios to place time-variant VNFs, as well as achieving desirable scalability.

Network slicing (NS) is also a form of virtual network architecture to support divergent requirements sustainably. The work from Liu *et al*.[97] proposes a device association scheme (such as access control and handover management) for radio access network (RAN) slicing by exploiting a hybrid federated deep reinforcement learning (HDRL) framework. In view of the large state-action space and variety of services, HDRL is designed with two layers of model aggregations. Horizontal aggregation deployed on BSs is used for the same type of service. Generally, data samples collected by different devices within the same service have similar features. The discrete-action DRL algorithm, *i.e*., DDQN, is employed to train the local model on individual smart devices. BS is able to aggregate model parameters and establish a cooperative global model. Vertical aggregation developed on the third encrypted party is responsible for the services of different types. In order to promote collaboration between devices with different tasks, authors aggregate local access features to form a global access feature, in which the data from different flows is strongly correlated since different data flows are competing for radio resources with each other. Furthermore, the Shapley value[98], which represents the average marginal contribution of a specific feature across all possible feature combinations, is used to reduce communication cost in vertical aggregation based on the global access feature. Simulation results show that HDRL can improve network throughput and communication efficiency.

The open radio access network (O-RAN) has emerged as a paradigm for supporting multi-class wireless services in 5G and beyond networks. To deal with the two critical issues of load balance and handover control, Cao *et al*.[99] proposes a federated DRL-based scheme to train the model for user access control in the O-RAN. Due to the mobility of UEs and the high cost of the handover between BSs, it is necessary for each UE to access the appropriate BS to optimize its throughput performance. As independent agents, UEs make access decisions with assistance from a global model server, which updates global DQN parameters by averaging DQN parameters of selected UEs. Further, the scheme proposes only partially exchanging DQN parameters to reduce communication overheads, and using the dueling structure to allow convergence for independent agents. Simulation results demonstrate that the scheme increases long-term throughput while avoiding frequent handovers of users with limited signaling overheads.

The issue of optimizing user access is important in wireless communication systems. FRL can provide interesting solutions for enabling efficient and privacy-enhanced management of access control. Zhang *et al*.[100] studies the problem of multi-user access in WIFI networks. In order to mitigate collision events on channel access, an enhanced multiple access mechanism based on FRL is proposed for user-dense scenarios. In particular, distributed stations train their local q-learning networks through channel state, access history and feedback from central access point (AP). AP uses the central aggregation algorithm to update the global model every period of time and broadcast it to all stations. In addition, a monte carlo (MC) reward estimation method for the training phase of local model is introduced, which allocates more weight to the reward of that current state by reducing the previous cumulative reward.

FRL is also studied for intelligent cyber-physical systems (ICPS), which aims to meet the requirements of intelligent applications for high-precision, low-latency analysis of big data. In light of the heterogeneity brought by multiple agents, the central RL-based resource allocation scheme has non-stationary issues and does not consider privacy issues. Therefore, the work from Xu *et al*.[101] proposes a multi-agent FRL (MA-FRL) mechanism which synthesizes a good inferential global policy from encrypted local policies of agents without revealing private information. The data resource allocation and secure communication problems are formulated as a

Stackelberg game with multiple participants, including near devices (NDs), far devices (FDs) and relay devices (RDs). Take into account the limited scope of the heterogeneous devices, the authors model this multi-agent system as a POMDP. Furthermore, it is proved that MA-FRL is μ-strongly convex and β-smooth and derives its convergence speed in expectation.

Zhang *et al.*[102] pays attention to the challenges in cellular vehicle-to-everything (V2X) communication for future vehicular applications. A joint optimization problem of selecting the transmission mode and allocating the resources is presented. This paper proposes a decentralized DRL algorithm for maximizing the amount of available vehicle-to-infrastructure capacity while meeting the latency and reliability requirements of vehicle-to-vehicle (V2V) pairs. Considering limited local training data at vehicles, the federated learning algorithm is conducted on a small timescale. On the other hand, the graph theory-based vehicle clustering algorithm is conducted on a large timescale.

The development of communication technologies in extreme environments is important, including deep underwater exploration. The architecture and philosophy of FRL are applied to smart ocean applications in the study of Kwon[103]. To deal with the nonstationary environment and unreliable channels of underwater wireless networks, the authors propose a multi-agent DRL-based algorithm that can realize FL computation with internet-of-underwater-things (IoUT) devices in the ocean environment. The cooperative model is trained by MADDPG for cell association and resource allocation problems. As for downlink throughput, it is found that the proposed MADDPG-based algorithm performed 80% and 41% better than the standard actor-critic and DDPG algorithms, respectively.

5.3. FRL for control optimization

Reinforcement learning based control schemes are considered as one of the most effective ways to learn a nonlinear control strategy in complex scenarios, such as robotics. Individual agent's exploration of the environment is limited by its own field of vision and usually needs a great deal of training to obtain the optimal strategy. The FRL-based approach has emerged as an appealing way to realize control optimization without exposing agent data or compromising privacy.

Automated control of robots is a typical example of control optimization problems. Liu *et al.*[57] discusses robot navigation scenarios and focuses on how to make robots transfer their experience so that they can make use of prior knowledge and quickly adapt to changing environments. As a solution, a cooperative learning architecture, called LFRL, is proposed for navigation in cloud robotic systems. Under the FRL-based architecture, the authors propose a corresponding knowledge fusion algorithm to upgrade the shared model deployed on the cloud. In addition, the paper also discusses the problems and feasibility of applying transfer learning algorithms to different tasks and network structures between the shared model and the local model.

FRL is combined with autonomous driving of robotic vehicles in the study of Liang *et al.*[104]. To reach rapid training from a simulation environment to a real-world environment, Liang *et al.*[104] presents a federated transfer reinforcement learning (FTRL) framework for knowledge extraction where all the vehicles make corresponding actions with the knowledge learned by others. The framework can potentially be used to train more powerful tasks by pooling the resources of multiple entities without revealing raw data information in real-life scenarios. To evaluate the feasibility of the proposed framework, authors perform real-life experiments on steering control tasks for collision avoidance of autonomous driving robotic cars and it is demonstrated that the framework has superior performance to the non-federated local training process. Note that the framework can be considered an extension of HFRL, because the target tasks to be accomplished are highly-relative and all observation data are pre-aligned.

FRL also appears as an attractive approach for enabling intelligent control of IoT devices without revealing private information. Lim *et al.*[105] proposes a FRL architecture which allows agents working on independent IoT devices to share their learning experiences with each other, and transfer the policy model parameters to other agents. The aim is to effectively control multiple IoT devices of the same type but with slightly different dynamics. Whenever an agent meets the predefined criteria, its mature model will be shared by the server with all other agents in training. The agents continue training based on the shared model until the local model converges in the respective environment. The actor-critical proximal policy optimization (Actor-Critic PPO) algorithm is integrated into the control of multiple rotary inverted pendulum (RIP) devices. The results show that the proposed architecture facilitates the learning process and if more agents participate the learning speed can be improved. In addition, Lim *et al.*[106] uses FRL architecture based on a multi-agent environment to solve the problems and limitations of RL for applications to the real-world problems. The proposed federation policy allows multiple agents to share their learning experiences to get better learning efficacy. The proposed scheme adopts Actor-Critic PPO algorithm for four types of RL simulation environments from OpenAI Gym as well as RIP in real control systems. Compared to a previous real-environment study, the scheme enhances learning performance by approximately 1.2 times.

5.4. FRL for attack detection

With the heterogeneity of services and the sophistication of threats, it is challenging to detect these attacks using traditional methods or centralized ML-based methods, which have a high false alarm rate and do not take privacy into account. FRL offers a powerful alternative to detecting attacks and provides support for network defense in different scenarios.

Because of various constraints, IoT applications have become a primary target for malicious adversaries that can disrupt normal operations or steal confidential information. In order to address the security issues in flying ad-hoc network (FANET), Mowla *et al.*[107] proposes an adaptive FRL-based jamming attack defense strategy for unmanned aerial vehicles (UAVs). A model-free Q-learning mechanism is developed and deployed on distributed UAVs to cooperatively learn detection models for jamming attacks. According to the results, the average accuracy of the federated jamming detection mechanism, employed in the proposed defense strategy, is 39.9% higher than the distributed mechanism when verified with the CRAWDAD standard and the ns-3 simulated FANET jamming attack dataset.

An efficient traffic monitoring framework, known as DeepMonitor, is presented in the study of Nguyen *et al.*[108] to provide fine-grained traffic analysis capability at the edge of software defined network (SDN) based IoT networks. The agents deployed in edge nodes consider the different granularity-level requirements and their maximum flow-table capacity to achieve the optimal flow rule match-field strategy. The control optimization problem is formulated as the MDP and a federated DDQN algorithm is developed to improve the learning performance of agents. The results show that the proposed monitoring framework can produce reliable traffic granularity at all levels of traffic granularity and substantially mitigate the issue of flow-table overflows. In addition, the distributed denial of service (DDoS) attack detection performance of an intrusion detection system can be enhanced by up to 22.83% by using DeepMonitor instead of FlowStat.

In order to reduce manufacturing costs and improve production efficiency, the industrial internet of things (IIoT) is proposed as a potentially promising research direction. It is a challenge to implement anomaly detection mechanisms in IIoT applications with data privacy protection. Wang *et al.*[109] proposes a reliable anomaly detection strategy for IIoT using FRL techniques. In the system framework, there are four entities involved in establishing the detection model, *i.e.*, the Global Anomaly Detection Center (GADC), the Local Anomaly Detection Center (LADC), the Regional Anomaly Detection Center (RADC), and the users. The anomaly detection is suggested to be implemented in two phases, including anomaly detection for RADC and users. Especially, the GADC can build global RADC anomaly detection models based on local models trained by LADCs. Different from RADC anomaly detection based on action deviations, user anomaly detection is mainly concerned with privacy leakage and is employed by RADC and GADC. Note that the DDPG algorithm is applied for local anomaly detection model training.

5.5. FRL for other applications

Due to the outstanding performance of training efficiency and privacy protection, many researchers are exploring the possible applications of FRL.

FL has been applied to realize distributed energy management in IoT applications. In the revolution of smart home, smart meters are deployed in the advanced metering infrastructure (AMI) to monitor and analyze the energy consumption of users in real-time. As an example[110], the FRL-based approach is proposed for the energy management of multiple smart homes with solar PVs, home appliances, and energy storage. Multiple local home energy management systems (LHEMSs) and a global server (GS) make up FRL architecture of the smart home. DRL agents for LHEMSs construct and upload local models to the GS by using energy consumption data. The GS updates the global model based on local models of LHEMSs using the federated stochastic gradient descent (FedSGD) algorithm. Under heterogeneous home environments, simulation results indicate that the proposed approach outperforms others when it comes to convergence speed, appliance energy consumption, and the number of agents.

Moreover, FRL offers an alternative to share information with low latency and privacy preservation. The collaborative perception of vehicles provided by IoV can greatly enhance the ability to sense things beyond their line of sight, which is important for autonomous driving. Region quadtrees have been proposed as a storage and communication resource-saving solution for sharing perception information[111]. It is challenging to tailor the number and resolution of transmitted quadtree blocks to bandwidth availability. In the framework of FRL, Mohamed *et al.*[112] presents a quadtree-based point cloud compression mechanism to select cooperative perception messages. Specifically, over a period of time, each vehicle covered by an RSU transfers its latest network weights with the RSU, which then averages all of the received model parameters and broadcasts the result back to the vehicles. Optimal sensory information transmission (*i.e.*, quadtree blocks) and appropriate resolution levels for a given vehicle pair are the main objectives of a vehicle. The dueling and branching concepts are also applied to overcome the vast action space inherent in the formulation of the RL problem. Simulation results show that the learned policies achieve higher vehicular satisfaction and the training process is enhanced by FRL.

5.6. Lessons Learned

In the following, we summarize the major lessons learned from this survey in order to provide a comprehensive understanding of current research on FRL applications.

5.6.1. Lessons learned from the aggregation algorithms

The existing FRL literature usually uses classical DRL algorithms, such as DQN and DDPG, at the participants, while the gradients or parameters of the critic and/or actor networks are periodically reported synchronously or asynchronously by the participants to the coordinator. The coordinator then aggregates the parameters or gradients and sends the updated values to the participants. In order to meet the challenges presented by different scenarios, the aggregation algorithms have been designed as a key feature of FRL. In the original FedAvg algorithm[12], the number of samples in a participant's dataset determines its influence on the global model. In accordance with this idea, several papers propose different methods to calculate the weights in the aggregation algorithms according to the requirement of application. In the study from Lim *et al.*[106], the aggregation weight is derived from the average of the cumulative rewards of the last ten episodes. Greater weights are placed on the models of those participants with higher rewards. In contrast to the positive correlation of reward, Huang *et al.*[96] takes the error rate of action as an essential factor to assign weights for participating in the global model training. In D2D -assisted edge caching, Wang *et al.*[89] uses the reward and some device-related indicators as the measurement to evaluate the local model's contribution to the global model. Moreover, the existing FRL methods based on offline DRL algorithms, such DQN and DDPG, usually use experience replay. Sampling random batch from replay memory can break correlations of continuous transition tuples and accelerate the training process. To arrive at an accurate evaluation of the participants, the paper[102] calculates the aggregation weight based on the size of the training batch in each iteration.

The above aggregation methods can effectively deal with the issue of data imbalance and performance discrepancy between participants, but it is hard for participants to cope with subtle environmental differences. According to the paper[105], as soon as a participant reaches the predefined criteria in its own environment, it should stop learning and send its model parameters as a reference to the remaining individuals. Exchanging mature network models (satisfying terminal conditions) can help other participants complete their training quickly. Participants in other similar environments can continue to use FRL for further updating their parameters to achieve the desired model performance according to their individual environments. Liu *et al.*[57] also suggests that the sharing global model in the cloud is not the final policy model for local participants. An effective transfer learning should be applied to resolve the structural difference between the shared network and private network.

5.6.2. Lessons learned from the relationship between FL and RL

In most of the literature on FRL, FL is used to improve the performance of RL. With FL, the learning experience can be shared among decentralized multiple parties while ensuring privacy and scalability without requiring direct data offloading to servers or third parties. Therefore, FL can expand the scope and enhance the security of RL. Among the applications of FRL, most researchers focus on the communication network system due to its robust security requirements, advanced distributed architecture, and a variety of decision-making tasks. Data offloading[93] and caching[89] solutions powered by distributed AI are available from FRL. In addition, with the ability to detect a wide range of attacks and support defense solutions, FRL has emerged as a strong alternative for performing distributed learning for security-sensitive scenarios. Enabled by the privacy-enhancing and cooperative features, detection and defense solutions can be learned quickly where multiple participants join to build a federated model[107,109]. FRL can also provide viable solutions to realize intelligence for control systems with many applied domains such as robotics[57] and autonomous driving[104] without data exchange and privacy leakage. The data owners (robot or vehicle) may not trust the third-party server and therefore hesitate to upload their private information to potentially insecure learning systems. Each participant of FRL runs a separate RL model for determining its own control policy and gains experience by sharing model parameters, gradients or losses.

Meanwhile, RL may have the potential to optimize FL schemes and improve the efficiency of training. Due to the unstable network connectivity, it is not practical for FL to update and aggregate models simultaneously across all participants. Therefore, Wang *et al.*[113] proposes a RL-based control framework that intelligently chooses the participants to participate in each round of FL with the aim to speed up convergence. Similarly, Zhang *et al.*[114] applies RL to pre-select a set of candidate edge participants, and then determine reliable edge participants through social attribute perception. In IoT or IoV scenarios, due to the heterogeneous nature of participating devices, different computing and communication resources are available to them. RL can speed up training by coordinating the allocation of resources between participants. Zhan *et al.*[115] defines the L4L (Learning for Learning) concept, *i.e.*, use RL to improve FL. Using the heterogeneity of participants and dynamic network connections, this paper investigates a computational resource control problem for FL that simultaneously considers learning time and energy efficiency. An experience-driven resource control approach based on RL is presented to derive the near-optimal strategy with only the participants' bandwidth information in the previous training rounds. In addition, as with any other ML algorithm, FL algorithms are vulnerable to malicious attacks. RL has been studied to defend against attacks in various scenarios, and it can also enhance the security of FL. The paper[116] proposes a reputation-aware RL (RA-RL) based selection method to ensure that FL is not disrupted. The participating devices' attributes, including computing resources and trust values, *etc*, are used as part of the environment in RL. In the aggregation of the global model, devices with high reputation levels will have a greater chance of being considered to reduce the effects of malicious devices mixed into FL.

5.6.3. Lessons learned from categories of FRL

As discussed above, FRL can be divided into two main categories, *i.e*., HFRL and VFRL. Currently, most of the existing research is focused on HFRL, while little attention is devoted to VFRL. The reason for this is that HFRL has obvious application scenarios, where multiple participants have similar decision-making tasks with individual environments, such as caching allocation[59], offloading optimization[58], and attack monitoring[108]. The participants and coordinator only need to train a similar model with the same state and action spaces. Consequently, the algorithm design can be implemented and the training convergence can be verified relatively easily. On the other hand, even though VFRL has a higher degree of technical difficulty at the algorithm design level, it also has a wide range of possible applications. In a multi-agent scenario, for example, a single agent is limited by its ability to observe only part of the environment, whereas the transition of the environment is determined by the behavior of all the agents. Zhuo *et al.*[65] assumes agents cannot share their partial observations of the environment and some agents are unable to receive rewards. The federated Q-network aggregation algorithm between two agents is proposed for VFRL. The paper[97] specifically applies both HFRL and VFRL for radio access network slicing. For the same type of services, similar data samples are trained locally at participating devices, and BSs perform horizontal aggregation to integrate a cooperative access model by adopting an iterative approach. The terminal device also can optimize the selection of base stations and network slices based on the global model of VFRL, which aggregates access features generated by different types of services on the third encrypted party. The method improves the device's ability to select the appropriate access points when initiating different types of service requests under restrictions regarding privacy protection. The feasible implementation of VFRL also provides guidance for future research.

6. OPEN ISSUES AND FUTURE RESEARCH DIRECTIONS

As we presented in the previous section, FRL serves an increasingly important role as an enabler of various applications. While the FRL-based approach possesses many advantages, there are a number of critical open issues to consider for future implementation. Therefore, this section focuses on several key challenges, including those inherited from FL such as security and communication issues, as well as those unique to FRL. Research on tackling these issues offers interesting directions for the future.

6.1. Learning convergence in HFRL

In realistic HFRL scenarios, while the agents perform similar tasks, the inherent dynamics for the different environments in which the agents reside are usually not exactly identically distributed. The slight difference in the stochastic properties of the transition models for multiple agents could cause the learning convergence issue. One possible method to address this problem is by adjusting the frequency of global aggregation, *i.e*., after each global aggregation, a period of time is left for each agent to fine-tune its local parameters according to its own environment. Apart from the non-identical environment problem, another interesting and important problem is how to leverage FL to make RL algorithms converge better and faster. It is well-known that DRL algorithms could be unstable and diverge, especially when off-policy training is combined with function approximation and bootstrapping. In FRL, the training curves of some agents could diverge while others converge although the agents are trained in the exact replicas of the same environment. By leveraging FL, it is envisioned that we could expedite the training process as well as increase the stability. For example, we could selectively aggregate the parameters of a subset of agents with a larger potential for convergence, and later transfer the converged parameters to all the agents. To tackle the above problems, several possible solutions proposed for FL algorithm contains certain reference significance. For example, server operators could account for heterogeneity inherent in partial information by adding a proximal term[117]. The local updates submitted by agents are constrained by the tunable term and have a different effect on the global parameters. In addition, a probabilistic agent selection scheme can be implemented to select the agents whose local FL models have significant effects on the global model to minimize the FL convergence time and the FL training loss[118]. Another problem is theoretical analysis of the convergence bounds. Although some existing studies have been directed at this problem[119], the convergence can be guaranteed since the loss function is convex. How to analyze and evaluate the non-convex loss functions in HFRL is also an important research topic in the future.

6.2. Agents without rewards in VFRL

In most existing works, all the RL agents have the ability to take part in full interaction with the environment and can generate their own actions and rewards. Even though some MARL agents may not participate in the policy decision, they still generate their own reward for evaluation. In some scenarios, special agents in VFRL take the role of providing assistance to other agents. They can only observe the environment and pass on the knowledge of their observation, so as to help other agents make more effective decisions. Therefore, such agents do not have their own actions and rewards. The traditional RL models cannot effectively deal with this thorny problem. Many algorithms either directly use the states of such agents as public knowledge in the system model or design corresponding action and reward for such agents, which may be only for convenience of calculation and have no practical significance. These approaches cannot fundamentally overcome the challenge, especially when privacy protection is also an essential objective to be complied with. Although the FedRL algorithm[65] is proposed to deal with the above problem, which has demonstrated good performance, there are still some limitations. First of all, the number of agents used in experiments and algorithms is limited to two, which means the scalability of this algorithm is not high and VFRL algorithms for a large number of agents need to be designed. Secondly, this algorithm uses Q-network as the federated model, which is a relatively simple algorithm. Therefore, how to design VFRL models based on other more complex and changeable networks remains an open issue.

6.3. Communications

In FRL, the agents need to exchange the model parameters, gradients, intermediate results, etc., between themselves or with a central server. Due to the limited communication resources and battery capacity, the communication cost is an important consideration when implementing these applications. With an increased number of participants, the coordinator has to bear more network workload within the client-server FRL model[120]. This is because each participant needs to upload and download model updates through the coordinator. Although the distributed peer-to-peer model does not require a central coordinator, each agent may have to exchange information with other participants more frequently. In current research for distributed models, there are no effective model exchange protocols to determine when to share experiences with which agents. In addition, DRL involves updating parameters in deep neural networks. Several popular DRL algorithms, such as DQN[121] and DDPG[122], consist of multiple layers or multiple networks. Model updates contain millions of parameters, which isn't feasible for scenarios with limited communication resources. The research directions for the above issues can be divided into three categories. First, it is necessary to design a dynamic update mechanism for participants to optimize the number of model exchanges. A second research direction is to use model compression algorithms to reduce the amount of communication data. Finally, aggregation algorithms that allow participants to only submit the important parts of local model should be studied further.

6.4. Privacy and Security

Although FL provides privacy protection that allows the agents to exchange information in a secure manner during the learning process, it still has several privacy and security vulnerabilities associated with communication and attack[123]. As FRL is implemented based on FL algorithms, these problems also exist in FRL in the same or variant form. It is important to note that the data poisoning attack is a different attack mode between FL and FRL. In the existing classification tasks of FL, each piece of training data in the dataset corresponds to a respective label. The attacker flips the labels on training examples in one category onto another while the features of the examples are kept unchanged, misguiding the establishment of a target model[124]. However, in the decision-making task of FRL, the training data is continuously generated from the interaction between the agent and the environment. As a result, the data poisoning attack is implemented in another way. For example, the malicious agent tampers with the reward, which causes the evaluative function to shift. An option is to conduct regular safety assessments for all participants. Participants whose evaluation indicator falls below the threshold are punished to reduce the impact on the global model[125]. Apart form the insider attacks which are launched by the agents in the FRL system, there may be various outsider attacks which are launched by intruders or eavesdroppers. Intruders may hide in the environment where the agent is and manipulate the transitions of environment to achieve specific goals. In addition, by listening to the communication between the coordinator and the agent, the eavesdropper may infer sensitive information from exchanging parameters and gradients[126]. Therefore, the development of technology that detects and protects against attacks and privacy threats does have great potential and is urgently needed.

6.5. Join and exit mechanisms design

One overlooked aspect of FRL-based research is the join and exit process of participants. In practice, the management of participants is essential to the normal progression of cooperation. As mentioned earlier in the security issue, the penetration of malicious participants severely impacts the performance of the cooperative model and the speed of training. The joining mechanism provides participants with the legal status to engage in federated cooperation. It is the first line of defense against malicious attackers. In contrast, the exit mechanism signifies the cancellation of the permission for cooperation. Participant-driven or enforced exit mechanisms are both possible. In particular, for synchronous algorithms, ignoring the exit mechanism can negatively impact learning efficiency. This is because the coordinator needs to wait for all participants to submit their information. In the event that any participant is offline or compromised and unable to upload, the time for one round of training will be increased indefinitely. To address the bottleneck, a few studies consider updating the global model using the selected models from a subset of participants[113,127]. Unfortunately, there is no comprehensive consideration of the exit mechanism, and the communication of participants is typically assumed to be reliable. Therefore, research gaps of FRL still exist in joining and exiting mechanisms. It is expected that the coordinator or monitoring system, upon discovering a failure, disconnection, or malicious participant, will use the exit mechanism to reduce its impact on the global model or even eliminate it.

6.6. Incentive mechanisms

For most studies, the agents taking part in the FRL process are assumed to be honest and voluntary. Each agent provides assistance for the establishment of the cooperation model following the rules and freely shares the masked experience through encrypted parameters or gradients. An agent's motivation for participation may come from regulation or incentive mechanisms. The FRL process within an organization is usually governed by regulations. For example, BSs belonging to the same company establish a joint model for offloading and caching. Nevertheless, because participants may be members of different organizations or use disparate equipment, it is difficult for regulation to force all parties to share information learned from their own data in the same manner. If there are no regulatory measures, participants prone to selfish behavior will only benefit from the cooperation model but not submit local updates. Therefore, the cooperation of multiple parties, organizations, or individuals requires a fair and efficient incentive mechanism to encourage their active participation. In this way, agents providing more contributions can benefit more and selfish agents unwilling to share there learning experience will receive less benefit. As an example, Google Keyboard[128] users can choose whether or not to allow Google to use their data, but if they do, they can benefit from more accurate word prediction. Although an incentive mechanism in a context-aware manner among data owners is proposed in the study from Yu *et al*.[129], it is not suitable for the RL problems. There is still no clear plan of action regarding how the FRL-based application can be designed to create a reasonable incentive mechanism for inspiring agents to participate in collaborative learning. To be successful, future research needs to propose a quantitative standard for evaluating the contribution of agents in FRL.

6.7. Peer-to-peer cooperation

FRL applications have the option of choosing between a central server-client model as well as a distributed peer-to-peer model. A distributed model can not only eliminate the single point of failure, but it can also improve energy efficiency significantly by allowing models to be exchanged directly between two agents. In a typical application, two adjacent cars share experience learned from road condition environment in the form of models with D2D communications to assist autonomous driving. However, the distributed cooperation increases the complexity of the learning system and imposes stricter requirements for application scenarios. This research should include, but not be limited to, the agent selection method for the exchange model, the mechanism for triggering the model exchange, the improvement of algorithm adaptability, and the convergence analysis of the aggregation algorithm.

7. CONCLUSION

As a new and potential branch of RL, FL can make learning safer and more efficient while leveraging the benefits of FL. We have discussed the basic definitions of FL and RL as well as our thoughts on their integration in this paper. In general, FRL algorithms can be classified into two categories, *i.e.*, HFRL and VFRL. Thus, the definition and general framework of these two categories have been given. Specifically, we have highlighted the difference between HFRL and VFRL. Then, a lot of existing FRL schemes have been summarized and analyzed according to different applications. Finally, the potential challenges in the development of FRL algorithms have been explored. Several open issues of FRL have been identified, which will encourage more efforts toward further research in FRL.

DECLARATIONS

Authors' contributions

Made substantial contributions to the research and investigation process, reviewed and summarized the literature, wrote and edited the original draft: Qi J, Zhou Q

Performed oversight and leadership responsibility for the research activity planning and execution, as well as developed ideas and evolution of overarching research aims: Lei L

Performed critical review, commentary and revision, as well as provided administrative, technical, and material support: Zheng K

Availability of data and materials

Not applicable.

Financial support and sponsorship

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Discovery Grant No. 401718) and the CARE-AI Seed Fund at the University of Guelph.

Conflicts of interest

The authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Nair A, Srinivasan P, Blackwell S, et al. Massively parallel methods for deep reinforcement learning. CoRR 2015;abs/1507.04296. Available from: http://arxiv.org/abs/1507.04296.
2. Grounds M, Kudenko D. Parallel reinforcement learning with linear function approximation. In: Tuyls K, Nowe A, Guessoum Z, Kudenko D, editors. Adaptive agents and multi-agent systems III. Adaptation and multi-agent learning. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. pp. 60–74.
3. Clemente AV, Martínez HNC, Chandra A. Efficient parallel methods for deep reinforcement learning. CoRR 2017;abs/1705.04862. Available from: http://arxiv.org/abs/1705.04862.
4. Lim WYB, Luong NC, Hoang DT, et al. Federated larning in mobile edge networks: A Comprehensive Survey. *IEEE Communications Surveys Tutorials* 2020;22:2031–63.
5. Nguyen DC, Ding M, Pathirana PN, et al. Federated learning for internet of things: a comprehensive survey. *IEEE Communications Surveys Tutorials* 2021;23:1622–58.
6. Khan LU, Saad W, Han Z, Hossain E, Hong CS. Federated learning for internet of things: recent advances, taxonomy, and open challenges. *IEEE Communications Surveys Tutorials* 2021;23:1759–99.
7. Yang Q, Liu Y, Cheng Y, et al. 1st ed. Morgan & Claypool; 2019.
8. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: concept and applications. *ACM T Intel Syst Tec* 2019;10:1–19.
9. Qinbin L, Zeyi W, Bingsheng H. Federated learning systems: vision, hype and reality for data privacy and protection. CoRR 2019;abs/1907.09693. Available from: http://arxiv.org/abs/1907.09693.
10. Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: challenges, methods, and future directions. *IEEE Signal Process Mag* 2020;37:50–60.
11. Wang S, Tuor T, Salonidis T, Leung KK, et al. Adaptive federated learning in resource constrained edge computing systems. *IEEE J Sel Area Comm* 2019;37:1205–21.
12. McMahan HB, Moore E, Ramage D, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. CoRR 2016;abs/1602.05629. Available from: http://arxiv.org/abs/1602.05629.
13. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE T Knowl Date En* 2018;13:1333–45.
14. Zhu H, Jin Y. Multi-objective evolutionary federated learning. *IEEE Transactions on Neural Networks and Learning Systems* 2020;31:1310–22.
15. Kairouz P, McMahan HB, Avent B, et al. Advances and open problems in federated learning. CoRR 2019;abs/1912.04977. Available from: http://arxiv.org/abs/1912.04977.
16. Pan SJ, Yang Q. A survey on transfer learning. *IEEE T Knowl Date En* 2010;22:1345–59.
17. Li Y. Deep reinforcement learning: an overview. CoRR 2017;abs/1701.07274. Available from: http://arxiv.org/abs/1701.07274.
18. Xu Z, Tang J, Meng J, et al. Experience-driven networking: a deep reinforcement learning based approach. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE; 2018. pp. 1871–79.
19. Mohammadi M, Al-Fuqaha A, Guizani M, Oh JS. Semisupervised deep reinforcement learning in support of IoT and smart city services. IEEE Internet of Things Journal 2018;5:624–35. [DOI: 10.1109/JIOT.2017.2712560]
20. Bu F, Wang X. A smart agriculture IoT system based on deep reinforcement learning. *Future Generation Com puter Systems* 2019;99:500–507. Available from: https://www.sciencedirect.com/science/article/pii/S0167739X19307277.
21. Xiong X, Zheng K, Lei L, Hou L. Resource allocation based on deep reinforcement learning in IoT edge computing. *IEEE J Sel Area Comm* 2020;38:1133–46.
22. Lei L, Qi J, Zheng K. Patent analytics based on feature vector space model: a case of IoT. *IEEE Access* 2019;7:45705–15.
23. Shalev-Shwartz S, Shammah S, Shashua A. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. CoRR 2016;abs/1610.03295. Available from: http://arxiv.org/abs/1610.03295.
24. Sallab AE, Abdou M, Perot E, Yogamani S. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*

2017;2017:70–76.
25. Taylor ME. Teaching reinforcement learning with mario: an argument and case study. In: Second AAAI Symposium on Educational Advances in Artificial Intelligence; 2011. Available from: https://www.aaai.org/ocs/index.php/EAAI/EAAI11/paper/viewPaper/3515.
26. Holcomb SD, Porter WK, Ault SV, Mao G, Wang J. Overview on deepmind and its alphago zero ai. In: Proceedings of the 2018 international conference on big data and education; 2018. pp. 67–71.
27. Watkins CJ, Dayan P. Q-learning. *Mach Learn* 1992;8:279–92. Available from: https://link.springer.com/content/pdf/10.1007/BF00992698.pdf.
28. Thorpe TL. Vehicle traffic light control using sarsa. In: Online]. Available: citeseer. ist. psu. edu/thorpe97vehicle. html. Citeseer; 1997. Available from: https://citeseer.ist.psu.edu/thorpe97vehicle.html.
29. Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms. In: Xing EP, Jebara T, editors. Proceedings of the 31st International Conference on Machine Learning. vol. 32 of Proceedings of Machine Learning Research. Bejing, China: PMLR; 2014. pp. 387–95. Available from: https://proceedings.mlr.press/v32/silver14.html.
30. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn* 1992;8:229–56.
31. Konda VR, Tsitsiklis JN. Actor-critic algorithms. In: advances in neural information processing systems; 2000. pp. 1008–14. Available from: https://proceedings.neurips.cc/paper/1786-actor-critic-algorithms.pdf.
32. Henderson P, Islam R, Bachman P, et al. Deep reinforcement learning that matters. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32; 2018. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/11694.
33. Lei L, Tan Y, Dahlenburg G, Xiang W, Zheng K. Dynamic energy dispatch based on deep reinforcement learning in IoT-Driven smart isolated microgrids. *IEEE Internet Things* 2021;8:7938–53.
34. Lei L, Xu H, Xiong X, Zheng K, Xiang W, et al. Multiuser resource control with deep reinforcement learning in IoT edge computing. *IEEE Internet Things* 2019;6:10119–33.
35. Ohnishi S, Uchibe E, Yamaguchi Y, et al. Constrained deep q-learning gradually approaching ordinary q-learning. *Front Neurorobotics* 2019;13:103.
36. Peng J, Williams RJ. Incremental multi-step Q-learning. In: machine learning proceedings 1994. Elsevier; 1994. pp. 226–32.
37. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;518:529–33.
38. Lei L, Tan Y, Zheng K, et al. Deep reinforcement learning for autonomous internet of things: model, applications and challenges. *IEEE Communications Surveys Tutorials* 2020;22:1722–60.
39. Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 30; 2016. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/10295.
40. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. arXiv preprint arXiv:151105952 2015. Available from: https://arxiv.org/abs/1511.05952.
41. Gu S, Lillicrap TP, Ghahramani Z, Turner RE, Levine S. Q-Prop: sample-efficient policy gradient with an off-policy critic. CoRR 2016;abs/1611.02247. Available from: http://arxiv.org/abs/1611.02247.
42. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochas- tic actor. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. pp. 1861–70. Available from: https://proceedings.mlr.press/v80/haarnoja18b.html.
43. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of The 33rd International Conference on Machine Learning. vol. 48 of Proceedings of Machine Learning Research. New York, New York, USA: PMLR; 2016. pp. 1928–37. Available from: https://proceedings.mlr.press/v48/mniha1 6.html.
44. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971 2015. Available from: https://arxiv.org/abs/1509.02971.
45. Barth-Maron G, Hoffman MW, Budden D, et al. Distributed distributional deterministic policy gradients. CoRR 2018;abs/1804.08617. Available from: http://arxiv.org/abs/1804.08617.
46. Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. pp. 1587–96. Available from: https://proceedings.mlr.press/v80/fujimoto18a.html.
47. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning. vol. 37 of Proceedings of Machine Learning Research. Lille, France: PMLR; 2015. pp. 1889–97. Available from: https://proceedings.mlr.press/v37/schulman15.html.
48. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. CoRR 2017;abs/1707.06347. Available from: http://arxiv.org/abs/1707.06347.
49. Zhu P, Li X, Poupart P. On improving deep reinforcement learning for POMDPs. CoRR 2017;abs/1704.07978. Available from: http://arxiv.org/abs/1704.07978.
50. Hausknecht M, Stone P. Deep recurrent q-learning for partially observable mdps. In: 2015 aaai fall symposium series; 2015. Available from: https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/viewPaper/11673.
51. Heess N, Hunt JJ, Lillicrap TP, Silver D. Memory-based control with recurrent neural networks. CoRR 2015;abs/1512.04455. Available

from: http://arxiv.org/abs/1512.04455.

52. Foerster J, Nardelli N, Farquhar G, et al. Stabilising experience replay for deep multi-agent reinforcement learning. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning. vol. 70 of Proceedings of Machine Learning Research. PMLR; 2017. pp. 1146–55. Available from: https://proceedings.mlr.press/v70/foerster17b.html.
53. Van der Pol E, Oliehoek FA. Coordinated deep reinforcement learners for traffic light control. Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016) 2016. Available from: https://www.elisevanderpol.nl/papers/vanderpolNIPSMALIC2016.pdf.
54. Foerster J, Farquhar G, Afouras T, Nardelli N, Whiteson S. Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32; 2018. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/11794.
55. Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. CoRR 2017;abs/1706.02275. Available from: http://arxiv.org/abs/1706.02275.
56. 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); 2019. pp. 123–27.
57. Liu B, Wang L, Liu M, Xu C. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. CoRR 2019;abs/1901.06455. Available from: http://arxiv.org/abs/1901.06455.
58. Ren J, Wang H, Hou T, Zheng S, Tang C. Federated learning-based computation offloading optimization in edge computing-supported internet of things. *IEEE Access* 2019;7:69194–201.
59. Wang X, Wang C, Li X, Leung VCM, Taleb T. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet Things* 2020;7:9441–55.
60. Chen J, Monga R, Bengio S, Józefowicz R. Revisiting Distributed Synchronous SGD. CoRR 2016;abs/1604.00981. Available from: http://arxiv.org/abs/1604.00981.
61. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of The 33rd International Conference on Machine Learning. vol. 48 of Proceedings of Machine Learning Research. New York, New York, USA: PMLR; 2016. pp. 1928–37. Available from: https://proceedings.mlr.press/v48/mniha1 6.html.
62. Espeholt L, Soyer H, Munos R, et al. IMPALA: scalable distributed deep-RL with importance weighted actor- learner architectures. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. pp. 1407–16. Available from: http://proceedings.mlr.press/v80/espeholt18a.html.
63. Horgan D, Quan J, Budden D, et al. Distributed prioritized experience replay. CoRR 2018;abs/1803.00933. Available from: http://arxiv.org/abs/1803.00933.
64. Liu T, Tian B, Ai Y, et al. Parallel reinforcement learning: a framework and case study. *IEEE/CAA Journal of Automatica Sinica* 2018;5:827–35.
65. Zhuo HH, Feng W, Xu Q, Yang Q, Lin Y. Federated reinforcement learning. CoRR 2019;abs/1901.08277. Available from: http: //arxiv.org/abs/1901.08277.
66. Canese L, Cardarilli GC, Di Nunzio L, et al. Multi-agent reinforcement learning: a review of challenges and applications. *Applied Sciences* 2021;11:4948. Available from: https://doi.org/10.3390/app11114948.
67. Busoniu L, Babuska R, De Schutter B. A comprehensive survey of Multiagent Reinforcement Learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 2008;38:156–72.
68. Zhang K, Yang Z, Başar T. Multi-agent reinforcement learning: a selective overview of theories and algorithms. *Handbook of Rein forcement Learning and Control* 2021:321–84.
69. Stone P, Veloso M. Multiagent systems: A survey from a machine learning perspective. *Auton Robot* 2000;8:345–83.
70. Szepesvári C, Littman ML. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Comput* 1999;11:2017–60.
71. Littman ML. Value-function reinforcement learning in Markov games. *Cogn Syst Res* 2001;2:55–66.
72. Tan M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning; 1993. pp. 330–37.
73. Lauer M, Riedmiller M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: In Proceedings of the Seventeenth International Conference on Machine Learning. Citeseer; 2000. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary.
74. Monahan GE. State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms. *Manage Sci* 1982;28:1–16.
75. Oroojlooyjadid A, Hajinezhad D. A review of cooperative multi-agent deep reinforcement learning. CoRR 2019;abs/1908.03963. Available from: http://arxiv.org/abs/1908.03963.
76. Bernstein DS, Givan R, Immerman N, Zilberstein S. The complexity of decentralized control of Markov decision processes. *Math Oper Res* 2002;27:819–40.
77. Omidshafiei S, Pazis J, Amato C, How JP, Vian J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning. vol. 70 of

Proceedings of Machine Learning Research. PMLR; 2017. pp. 2681–90. Available from: https://proceedings.mlr.press/v70/omidshafiei17a.html.

78. Han Y, Gmytrasiewicz P. Ipomdp-net: a deep neural network for partially observable multi-agent planning using interactive pomdps. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33; 2019. pp. 6062–69.

79. Karkus P, Hsu D, Lee WS. QMDP-Net: deep learning for planning under partial observability; 2017. Available from: https://arxiv.org/abs/1703.06692.

80. Mao W, Zhang K, Miehling E, Başar T. Information state embedding in partially observable cooperative multi-agent reinforcement learning. In: 2020 59th IEEE Conference on Decision and Control (CDC); 2020. pp. 6124–31.

81. Mao H, Zhang Z, Xiao Z, Gong Z. Modelling the dynamic joint policy of teammates with attention multi-agent DDPG. CoRR 2018;abs/1811.07029. Available from: http://arxiv.org/abs/1811.07029.

82. Lee HR, Lee T. Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response. *Eur J Oper Res* 2021;291:296–308.

83. Sukhbaatar S, szlam a, Fergus R. Learning multiagent communication with backpropagation. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc.; 2016. Available from: https://proceedings.neurips.cc/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf.

84. Foerster JN, Assael YM, de Freitas N, Whiteson S. Learning to communicate with deep multi-agent reinforcement learning. CoRR 2016;abs/1605.06676. Available from: http://arxiv.org/abs/1605.06676.

85. Buşoniu L, Babuška R, De Schutter B. Multi-agent reinforcement learning: an overview. *Innovations in multiagent systems and applications1* 2010:183–221.

86. Hu Y, Hua Y, Liu W, Zhu J. Reward shaping based federated reinforcement learning. *IEEE Access* 2021;9:67259–67.

87. Anwar A, Raychowdhury A. Multi-task federated reinforcement learning with adversaries. CoRR 2021;abs/2103.06473. Available from: https://arxiv.org/abs/2103.06473.

88. Wang X, Han Y, Wang C, et al. In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network* 2019;33:156–65.

89. Wang X, Li R, Wang C, et al. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE J Sel Area Comm* 2021;39:154–69.

90. Zhang M, Jiang Y, Zheng FC, Bennis M, You X. Cooperative edge caching via federated deep reinforcement learning in fog-RANs. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops); 2021. pp. 1–6.

91. Majidi F, Khayyambashi MR, Barekatain B. HFDRL: an intelligent dynamic cooperate cashing method based on hierarchical federated deep reinforcement learning in edge-enabled IoT. *IEEE Internet Things* 2021:1-1.

92. Zhao L, Ran Y, Wang H, Wang J, Luo J. Towards cooperative caching for vehicular networks with multi-level federated reinforcement learning. In: ICC 2021 - IEEE International Conference on Communications; 2021. pp. 1–6.

93. Zhu Z, Wan S, Fan P, Letaief KB. Federated multi-agent actor-critic learning for age sensitive mobile edge computing. *IEEE Internet Things* 2021:1-1.

94. Yu S, Chen X, Zhou Z, Gong X, Wu D. When deep reinforcement learning meets federated learning: intelligent multi-timescale resource management for multi-access edge computing in 5G ultra dense network. arXiv:200910601 [cs] 2020 Sep. ArXiv: 2009.10601. Available from: http://arxiv.org/abs/2009.10601.

95. Tianqing Z, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things* 2021:1–1.

96. Huang H, Zeng C, Zhao Y, et al. Scalable orchestration of service function chains in NFV-Enabled networks: a federated reinforcement learning approach. *IEEE J Sel Area Comm* 2021;39:2558–71.

97. Liu YJ, Feng G, Sun Y, Qin S, Liang YC. Device association for RAN slicing based on hybrid federated deep reinforcement learning. *IEEE T Veh Technol* 2020;69:15731–45.

98. Wang G, Dang CX, Zhou Z. Measure contribution of participants in federated learning. In: 2019 IEEE International Conference on Big Data (Big Data); 2019. pp. 2597–604.

99. Cao Y, Lien SY, Liang YC, Chen KC. Federated deep reinforcement learning for user access control in open radio access networks. In: ICC 2021 - IEEE International Conference on Communications; 2021. pp. 1–6.

100. Zhang L, Yin H, Zhou Z, Roy S, Sun Y. Enhancing WiFi multiple access performance with federated deep reinforcement learning. In: 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall); 2020. pp. 1–6.

101. Xu M, Peng J, Gupta BB, et al. Multi-agent federated reinforcement learning for secure incentive mechanism in intelligent cyber-physical systems. *IEEE Internet Things* 2021:1–1.

102. Zhang X, Peng M, Yan S, Sun Y. Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications. *IEEE Internet Things* 2020;7:6380–91.

103. Kwon D, Jeon J, Park S, Kim J, Cho S. Multiagent DDPG-based deep learning for smart ocean federated learning IoT networks. *IEEE Internet Things* 2020;7:9895–903.

104. Liang X, Liu Y, Chen T, Liu M, Yang Q. Federated transfer reinforcement learning for autonomous driving. arXiv:191006001 [cs] 2019 Oct. ArXiv: 1910.06001. Available from: http://arxiv.org/abs/1910.06001.
105. Lim HK, Kim JB, Heo JS, Han YH. Federated reinforcement learning for training control policies on multiple IoT devices. Sensors 2020 Mar;20:1359. Available from: https://www.mdpi.com/1424-8220/20/5/1359.
106. Lim HK, Kim JB, Ullah I, Heo JS, Han YH. Federated reinforcement learning acceleration method for precise control of multiple devices. *IEEE Access* 2021;9:76296–306.
107. Mowla NI, Tran NH, Doh I, Chae K. AFRL: adaptive federated reinforcement learning for intelligent jamming defense in FANET. *Journal of Communications and Networks* 2020;22:244–58.
108. Nguyen TG, Phan TV, Hoang DT, Nguyen TN, So-In C. Federated deep reinforcement learning for traffic monitoring in SDN-Based IoT networks. *IEEE T Cogn Commun* 2021:1–1.
109. Wang X, Garg S, Lin H, et al. Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning. *IEEE Internet Things* 2021:1–1.
110. Lee S, Choi DH. Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources. *IEEE T Ind Inform* 2020:1–1.
111. Samet H. The quadtree and related hierarchical data structures. *ACM Comput Surv* 1984;16:187–260. Available from: https: //doi.org/10.1145/356924.356930.
112. Abdel-Aziz MK, Samarakoon S, Perfecto C, Bennis M. Cooperative perception in vehicular networks using multi-agent reinforcement learning. In: 2020 54th Asilomar Conference on Signals, Systems, and Computers; 2020. pp. 408–12.
113. Wang H, Kaplan Z, Niu D, Li B. Optimizing federated learning on non-IID data with reinforcement learning. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. Toronto, ON, Canada: IEEE; 2020. pp. 1698–707. Available from: https: //ieeexplore.ieee.org/document/9155494/.
114. Zhang P, Gan P, Aujla GS, Batth RS. Reinforcement learning for edge device selection using social attribute perception in industry 4.0. *IEEE Internet Things* 2021:1–1.
115. Zhan Y, Li P, Leijie W, Guo S. L4L: Experience-driven computational resource control in federated learning. *IEEE T Comput* 2021:1–1.
116. Dong Y, Gan P, Aujla GS, Zhang P. RA-RL: reputation-aware edge device selection method based on reinforcement learning. In: 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM); 2021. pp. 348–53.
117. Sahu AK, Li T, Sanjabi M, et al. On the convergence of federated optimization in heterogeneous networks. CoRR 2018;abs/1812.06127. Available from: http://arxiv.org/abs/1812.06127.
118. Chen M, Poor HV, Saad W, Cui S. Convergence time optimization for federated learning over wireless networks. *IEEE T Wirel Commun* 2021;20:2457–71.
119. Li X, Huang K, Yang W, Wang S, Zhang Z. On the convergence of FedAvg on non-IID data; 2020. Available from: https://arxiv.org/abs/1907.02189?context=stat.ML.
120. Bonawitz KA, Eichner H, Grieskamp W, et al. Towards federated learning at scale: system design. CoRR 2019;abs/1902.01046. Available from: http://arxiv.org/abs/1902.01046.
121. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015;518:529–33. Available from: https://doi.org/10.1038/nature14236.
122. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning; 2019. Available from: https://arxiv.org/abs/1509.02971.
123. Lyu L, Yu H, Yang Q. Threats to federated learning: a survey. CoRR 2020;abs/2003.02133. Available from: https://arxiv.org/abs/2003.02133.
124. Fung C, Yoon CJM, Beschastnikh I. Mitigating sybils in federated learning poisoning. CoRR 2018;abs/1808.04866. Available from: http://arxiv.org/abs/1808.04866.
125. Anwar A, Raychowdhury A. Multi-task federated reinforcement learning with adversaries; 2021.
126. Zhu L, Liu Z, Han S. Deep leakage from gradients. CoRR 2019;abs/1906.08935. Available from: http://arxiv.org/abs/1906.08935.
127. Nishio T, Yonetani R. Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC); 2019. pp. 1–7.
128. Yang T, Andrew G, Eichner H, et al. Applied federated learning: improving Google Keyboard query suggestions. CoRR 2018;abs/1812.02903. Available from: http://arxiv.org/abs/1812.02903.
129. Yu H, Liu Z, Liu Y, et al. A fairness-aware incentive scheme for federated learning. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. AIES '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 393–399. Available from: https://doi.org/10.1145/3375627.3375840.