A Unified Self-Regulating Training Framework for Federated Deep Reinforcement Learning

Meng Xu[1], Xinhong Chen[*1], Zhongying Chen[2], Guanyi Zhao[1], Yang Jin[3], Jianping Wang[*1]

[1]Department of Computer Science, City University of Hong Kong, Hong Kong, China
[2]School of Computer Science and Engineering, Southeast University, Nanjing, China
[3]Hangzhou Hikvision Digital Technology Co., Ltd., Hangzhou, China
xinhong.chen@cityu.edu.hk; jianwang@cityu.edu.hk

Abstract

Federated Deep Reinforcement Learning (FDRL) aims to enable distributed collaborative training of multiple DRL models while preserving privacy. Existing FDRL methods function in static client environments, but real-world scenarios often involve dynamic state transitions, such as noise, which render static model topologies inadequate and result in biased policy loss. This degrades client performance and leads to suboptimal global policies. To address this challenge, we develop a generic solution, referred to as the self-regulating training framework, which can be seamlessly integrated into existing FDRL approaches to address dynamic state transitions. Specifically, we propose a Sparse Training (ST) method that dynamically sparsifies and adjusts the topology of each model during training to maximize model performance and reduce model complexity. Additionally, we introduce an auxiliary model to adaptively regulate the policy loss of client models, mitigating loss bias and facilitating updates that yield improved returns. Experimental results demonstrate that our method enhances six state-of-the-art (SOTA) FDRL approaches across nine tasks in terms of return.

Introduction

FDRL enables the distributed training of DRL models across a network of devices or edge nodes without sharing raw data. This approach preserves data privacy while leveraging the local knowledge of client devices in a distributed environment, thus holding the potential to train a more performant global model. Consequently, FDRL has gained significant attention and is widely applied in areas such as recommendation systems (Javeed et al. 2023), autonomous driving (Fu et al. 2022), and robotic control (Yuan, Xu, and Zhu 2024).

Existing FDRL methods are categorized into Horizontal FDRL (HFDRL) and Vertical FDRL (VFDRL). HFDRL involves clients' training in independent local environments (Cha et al. 2020)(Jiang et al. 2025)(Wang et al. 2023), while VFDRL requires clients to collaborate in a shared environment with diverse observations (Zhuo et al. 2019). HFDRL effectively protects client privacy while delivering superior model performance, making it the dominant approach. This work falls within this category. Current FDRL methods typically assume that each client's environment remains static, implying that state transition dynamics do not change. However, real-world applications face factors like system noise and environmental variability, which introduce randomness and alter state transition dynamics, leading to dynamic transitions. This variability compromises the performance of existing FDRL methods, resulting in suboptimal policies. For example, client vehicles using current FDRL methods may learn suboptimal policies due to dynamic road environments, leading to frequent collisions and traffic accidents.

Despite efforts within the DRL community to address dynamic environments, these methods fall short for FDRL for two main reasons. First, federated learning exacerbates the negative effects of dynamic environments on DRL. In FDRL, the global model is highly sensitive to the quality of client models (Jin et al. 2022)(Mai et al. 2023). Dynamic environments can lead to suboptimal client models, which, when aggregated, significantly degrade the global model's performance. This degraded model is then downloaded by clients for further training, creating a negative feedback loop. Second, FDRL imposes strict limitations on client model size and training due to the limited hardware capabilities of devices like smartphones. Current solutions often rely on larger models, such as Generative Adversarial Networks (GANs) (Luo et al. 2024)(Khayatian, Nagy, and Bollinger 2021), which demand significant computational, storage, and communication resources to mitigate the effects of dynamic transitions. Thus, these approaches are impractical for resource-constrained clients and limited communication bandwidth.

Motivated by these observations, this work proposes a generic approach that can be seamlessly integrated into existing FDRL methods to address dynamic task environments. To achieve this, three challenges must be addressed. First, existing research (Grooten et al. 2023) shows that static model topologies are insufficient for dynamic environments, requiring model topology adjustments during training to enhance performance. Second, clients cannot handle larger models, so the proposed method must minimize computational and storage costs. Finally, dynamic state transitions result in inaccurate reward signals, leading to biased loss calculations during model updates (DiMarco, Shipp, and Kishida 2024). This loss bias hinders the model's ability to train toward optimal returns, ultimately resulting in suboptimal policies.

To address the first and second challenges, we propose an ST method that sparsifies the FDRL model and dynamically

*Corresponding authors: Xinhong Chen, Jianping Wang.

adjusts its sparse topology during training. This approach optimizes the model's topology to improve performance and reduce its size. Specifically, we adjust each local model's topology by removing a fixed number of unimportant connections after each update and introducing an equal number of new connections in previously unconnected areas. During the global aggregation process, each local model retains a distinct topology. When local models aggregate into a global model, they maintain their individual connections (Cha et al. 2020)(Jiang et al. 2025)(Wang et al. 2023), resulting in the global model having a different sparsity than the local models. To resolve this, we apply a pruning step on the global model after aggregation to align its sparsity with that of the local models. To address the third challenge, we introduce an auxiliary model to adjust the policy loss of each local model, guiding the training process toward higher returns in dynamic environments. The auxiliary model computes an auxiliary loss, complementing the existing policy loss to reduce bias from dynamic state transitions. It is trained to maximize the Q-value by incorporating this auxiliary loss, ensuring the policy achieves greater returns.

The key contributions are: 1) We present a generic solution that integrates seamlessly with existing FDRL approaches, dynamically adjusting model topology and loss to handle dynamic state transitions. 2) We provide a complexity analysis covering computational, storage, and communication costs, along with a theoretical analysis explaining why our method outperforms existing approaches, demonstrating its superiority over current FDRL methods. 3) Extensive experiments show that our method outperforms six SOTA FDRL methods across nine dynamic tasks, achieving superior returns.

Related Work

FDRL. FDRL techniques are categorized into HFDRL and VFDRL. In HFDRL, agents operate independently in local environments with no mutual influence (Cha et al. 2020)(Jiang et al. 2025)(Wang et al. 2023). Key works include Sherine et al.'s hybrid FDRL approach for personalized gaming (Bodas et al. 2018), Fan et al.'s FDRL framework addressing the Byzantine General Problem (Fan et al. 2021), and a decentralized FDRL model (Wang et al. 2023). VFDRL involves agents collaborating in a shared environment with limited observations, as demonstrated by Zhuo et al.'s integration of FDRL with Gaussian differentials (Zhuo et al. 2019). However, due to its inadequate privacy protection, HFDRL is more commonly used than VFDRL. Recently, FDRL research has intersected with transfer learning, multitask learning, and policy distillation. Liang et al. propose an online FDRL method for car steering control (Liang et al. 2022), Liu et al. focus on meta-model training for lifelong learning (Liu, Wang, and Liu 2019), and Mai et al. introduce policy distillation for federated model aggregation (Mai et al. 2023). Other efforts, including (Zhou et al. 2024), (Wen et al. 2023), (Yue et al. 2024), and (Rengarajan et al. 2025), explore offline FDRL. However, existing FDRL methods are constrained by static environments, with real-world noise and dynamic factors significantly affecting performance.

DRL in dynamic environments. Addressing dynamic environments in DRL is a major challenge. Recent solutions focus on learning latent variables to infer environmental dynamics (Wei and Luo 2021)(Lee et al. 2020), stable state distributions amid dynamic shifts (Xue et al. 2024), and dynamic models for domain generalization (Cang et al. 2021). Additionally, Luo et al. (Luo et al. 2024) use a local replay buffer to capture stable dynamics, improving policy learning, while Bing et al. (Bing et al. 2023) introduce a meta-learning approach to enhance DRL. However, these solutions are not directly applicable to FDRL in dynamic environments due to two main reasons: 1) Federated learning exacerbates the impact of dynamic environments, with poor local models degrading the global model, creating a harmful feedback loop; 2) Current methods require extensive model components, demanding significant resources that resource-limited client devices and bandwidth cannot support.

ST. Sparsification in neural networks has been a key focus in the DRL community, with past studies exploring pruning techniques. Livne and Cohen (Livne and Cohen 2020) used a dense teacher network for iterative pruning and retraining via knowledge distillation. Zhang et al. (Zhang, He, and Li 2019) paired a smaller network with a larger target network to improve policy and reduce sampling time. Lee et al. (Lee et al. 2021) combined weight grouping and pruning for efficient weight compression, while Vischer et al. (Vischer, Lange, and Sprekeler 2021) explored the lottery ticket hypothesis, showing sparse subnetworks outperform dense ones. However, iteratively pruning dense networks can be computationally expensive. Recently, ST methods have emerged for training sparse DRL from scratch, optimizing model topology, and reducing parameters (Sokar et al. 2021)(Xu, Chen, and Wang 2024)(Grooten et al. 2023)(Tan et al. 2023)(Xu et al. 2025). Yet, these ST methods apply only to regular DRL and cannot be used in FDRL, as aggregating local models increases global model connections, causing sparsity inconsistencies.

Problem Formulation

Each client in FDRL employs DRL, training models using samples obtained through online interactions with the environment. Training a convergent FDRL model is achieved by minimizing the weighted aggregation of the Temporal Difference (TD) error $F(\theta)$ across all clients, where $F_i(\theta_i)$ represents the TD error for the i-th client. Let γ be the discount factor, s the current state, a the action, r the reward, and s' the next state. Let ij denote the index of the j-th sample from the i-th client, θ^{Q_i} the critic for the i-th client, and $\theta^{Q_i}{}'$ the target critic. The target value for the i-th client is $r_t^{ij} + \gamma \min_{i=1,2} Q_{\theta^{Q_i}{}'}\left(s_{t+1}^{ij}, \tilde{a}\right)$, and the current Q-value is $Q_{\theta^{Q_i}}\left(s_t^{ij}, a_t^{ij}\right)$. The minimization process of FDRL's TD error is expressed as $\min_\theta F(\theta) = \sum_{i=1}^{N_a} p_i F_i(\theta_i)$, where

$$F_i(\theta_i) = \sum_j \left(Q_{\theta^{Q_i}}\left(s_t^{ij}, a_t^{ij}\right) - \left(r_t^{ij} + \gamma \min_{i=1,2} Q_{\theta^{Q_i}{}'}\left(s_{t+1}^{ij}, \tilde{a}\right)\right)\right)^2 \tag{1}$$

FDRL performance is evaluated based on the cumulative reward from the global model θ on the server, with a higher

cumulative reward indicating better performance. While existing FDRL methods have made significant progress in aggregating local models to enhance global model performance in static environments, real-world tasks are dynamic, influenced by factors such as environmental changes and system noise. These dynamics limit the effectiveness of current FDRL methods, resulting in suboptimal global models.

Methodology

This section begins with an outline of our method's framework, followed by a detailed overview of each module and a complexity analysis. The supplementary materials provide a full description of the pseudocode for our method and a theoretical analysis demonstrating its advantages.

The Framework For Our Method

The framework for our method is illustrated in Fig. 1, which introduces two modules to existing FDRL methods. The first module, the ST, aims to: 1) sparsify and adaptively adjust the topology of each local model throughout the training process to improve model performance and reduce size by dynamically removing unimportant connections and adding new ones at random locations, and 2) align the sparsity of the global model with that of each local model after aggregation on the server. The second module is the auxiliary model, integrated with each local model. The auxiliary model adjusts the actor's loss in each local model by calculating an auxiliary loss, guiding the policy toward higher returns. The auxiliary model is trained using auxiliary mini-batches of the same size as the regular ones, randomly sampled from the local replay buffer, which stores the samples collected by each client from its interactions with the environment.

The Training For The Local Model

We now illustrate the client-side training process using a DRL model with one actor and two critics as an example, outlined as follows.

(1) Initialization. Let θ^{μ} denote the actor, θ^{Q_1} and θ^{Q_2} represent the two critics, and $\hat{\theta}$ is the auxiliary model. Let L be the number of layers in each network. The target actor and target critics, denoted as $\theta^{\mu\prime}$, ${\theta^{Q_1}}'$, and ${\theta^{Q_2}}'$, share the same initial sparse topology as the current networks.

The sparsification of each model is implemented using a binary mask $\boldsymbol{M} = \left\{\boldsymbol{M}^i\right\}_{i=1}^{L}$, which indicates the locations of connections between layers. For $M^{ij} \in \{0,1\}$, $M^{ij} = 1$ indicates the presence of a connection j from layer $i-1$ to layer i, while $M^{ij} = 0$ signifies no connection. Let n^{i-1} and n^i represent the number of neurons in layers $i-1$ and i, respectively. Following the current ST approach (Sokar et al. 2021)(Xu, Chen, and Wang 2024)(Grooten et al. 2023)(Tan et al. 2023)(Xu et al. 2025), we initialize the sparse topology of each layer i using an Erdős-Rényi random graph, where the mask probability P_{M^i} for layer i is given by $P_{M^i} = \xi^i \frac{n^i + n^{i-1}}{n^i \times n^{i-1}}$. Here, ξ^i controls the sparsity level for layer i, and $1 - P_{M^i}$ represents the sparsity of the layer. Let $\boldsymbol{M}_{\theta^{\mu}}$ denote the binary mask for the actor, and $\boldsymbol{M}_{\theta^{Q_1}}, \boldsymbol{M}_{\theta^{Q_2}}, \boldsymbol{M}_{\hat{\theta}}$ represent the binary masks for each critic and auxiliary model, respectively. The sparse topology for the actor, critic, and auxiliary model is then defined as $\theta^{\mu} = \theta^{\mu} \odot \boldsymbol{M}_{\theta^{\mu}}; \hat{\theta} = \hat{\theta} \odot \boldsymbol{M}_{\hat{\theta}}; \theta^{Q_l} = \theta^{Q_l} \odot \boldsymbol{M}_{\theta^{Q_l}}, \quad \forall l \in \{1,2\}$. Here, $\odot$ is the element-wise multiplication operator.

(2) Training with the auxiliary model. We now introduce the training of the actor, the critics, and the auxiliary model separately. Before model updates, a mini-batch $\mathcal{D}_0$ of $|\mathcal{D}_0|$ samples $\{(s,a,r,s')\}$ is drawn from the replay buffer $\mathcal{D}$ to update both the actor and the critics. Additionally, a separate auxiliary mini-batch $\mathcal{D}_0^{\text{aux}}$ of $|\mathcal{D}_0^{\text{aux}}|$ transitions, denoted as $\left\{\left(\hat{s},\hat{a},\hat{r},\hat{s'}\right)\right\}$, is sampled from the replay buffer $\mathcal{D}$ to update the auxiliary model. Note that $|\mathcal{D}_0^{\text{aux}}| = |\mathcal{D}_0|$.

(2.1) The update of critics. The critic θ^{Q_i} is updated using the loss function $\mathcal{L}(\theta^{Q_i})$, defined as follows:

$$\mathcal{L}(\theta^{Q_i}) = \text{argmin}_{\theta^{Q_i}} |\mathcal{D}_0|^{-1} \sum_{(s,a)\in\mathcal{D}_0} \left(y - Q_{\theta^{Q_i}}(s,a)\right)^2 \tag{2}$$

Here, $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta^{Q_i\prime}}(s', \tilde{a})$, where $\tilde{a} \leftarrow \pi_{\theta^{\mu\prime}}(s') + \epsilon$, with $\epsilon \sim \mathcal{N}(0, 0.2)$ representing noise added to the target actor's actions, which are clipped to the range $(-0.5, 0.5)$.

(2.2) The update of the actor. The actor is updated using the mini-batch $\mathcal{D}_0$ in two steps: 1) the standard policy gradient update, and 2) the update using the auxiliary model. The first step involves the conventional update, $J(\theta^{\mu})$, as follows:

$$J(\theta^{\mu}) = |\mathcal{D}_0|^{-1} \sum_{(s,a)\in\mathcal{D}_0} \nabla_a Q_{\theta^{Q_1}}(s, \pi_{\theta^{\mu}(s)}) \nabla_{\theta^{\mu}} \pi_{\theta^{\mu}}(s) \tag{3}$$

After the standard update, we compute the actor's loss on the auxiliary mini-batch $\mathcal{D}_0^{\text{aux}}$, denoted as $J^{\text{aux1}}(\theta^{\mu})$, as follows:

$$J^{aux1}(\theta^{\mu}) = -|\mathcal{D}_0^{\text{aux}}|^{-1} \sum_{(\hat{s},\hat{a})\in\mathcal{D}_0^{\text{aux}}} Q_{\theta^{Q_1}}(\hat{s}, \pi_{\theta^{\mu}(\hat{s})}) \tag{4}$$

Next, we update the actor using the auxiliary model, a process referred to as the auxiliary update, as follows. Typically, the actor $\pi_{\theta^{\mu}}(s) = \bar{\pi}(\tilde{\pi}(s))$ is viewed as both a feature extraction module $\tilde{\pi}_{\theta^{\mu}}$ and a decision-making module $\bar{\pi}_{\theta^{\mu}}$, with the decision-making module implemented as the final layer of the actor network. The features extracted by $\tilde{\pi}_{\theta^{\mu}}$, along with the state s and action a, are then input into an auxiliary model $\hat{\theta}$, consisting of a three-layer sparse Multi-Layer Perceptron (MLP). The output of this model is used to compute the loss $\mathcal{L}_a(\theta^{\mu})$ as follows:

$$\mathcal{L}_a(\theta^{\mu}) = |\mathcal{D}_0|^{-1} \sum_{(s,a)\in\mathcal{D}_0} \hat{\theta}\left(\tilde{\pi}_{\theta^{\mu}}, s, a\right) \tag{5}$$

where $\hat{\theta}\left(\tilde{\pi}_{\theta^{\mu}}, s, a\right)$ is the output of the auxiliary model when input $\tilde{\pi}_{\theta^{\mu}}$, s and a. The actor is then updated using the auxiliary loss $\mathcal{L}_a(\theta^{\mu})$ as follows: $\theta^{\mu} \leftarrow \theta^{\mu} + \alpha \nabla_a \mathcal{L}_a(\theta^{\mu})$ where α is the learning rate. After updating the actor with the auxiliary model and letting $\bar{\theta^{\mu}}$ represent the updated actor, we compute the actor's loss $J^{aux2}(\bar{\theta^{\mu}})$ using the auxiliary mini-batch $\mathcal{D}_0^{\text{aux}}$, as follows:

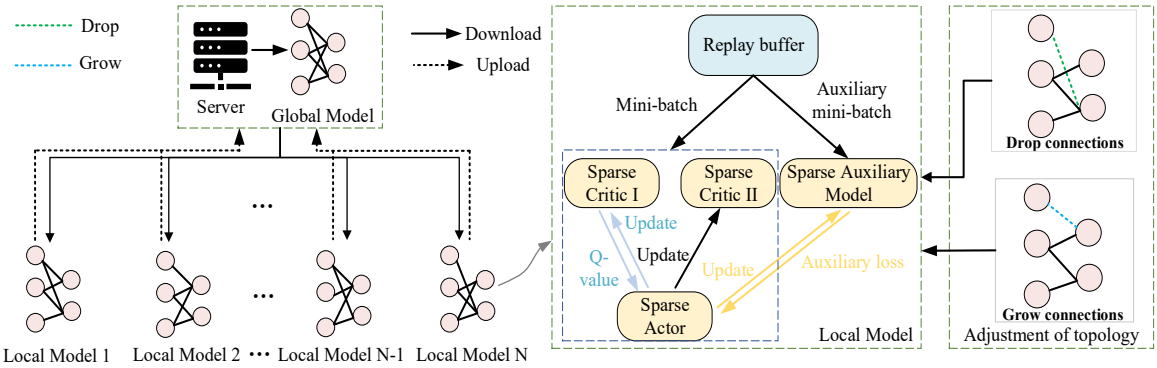

Figure 1: The framework for our method. Here, we illustrate our approach using a DRL model with two critics and one actor.

$$J^{aux2}(\bar{\theta^{\mu}}) = -|\mathcal{D}_0^{\text{aux}}|^{-1} \sum_{(\hat{s},\hat{a})\in\mathcal{D}_0^{\text{aux}}} Q_{\theta^{Q_1}}(\hat{s}, \pi_{\bar{\theta^{\mu}}}(\hat{s})) \quad (6)$$

(2.3) The update of the auxiliary model. Once the actor and critic have been updated, we proceed to update the auxiliary model. The auxiliary model improves actor updates by reducing actor loss on the mini-batch $\mathcal{D}_0^{\text{aux}}$ after the auxiliary update, compared to before. This reduction indicates the actor's benefit from the auxiliary model, resulting in better updates. Additionally, lower actor loss correlates with higher Q-values, reflecting greater long-term returns. Consequently, the loss for the auxiliary model consists of two parts. The first part, $\mathcal{L}_1$, represents the TD loss on the auxiliary mini-batch $\mathcal{D}_0^{\text{aux}}$, and is given by:

$$\mathcal{L}_1 = \sum_{i=1}^{2} \left\{ |\mathcal{D}_0^{\text{aux}}|^{-1} \sum_{(\hat{s},\hat{a})\in\mathcal{D}_0^{\text{aux}}} (\hat{y} - Q_{\theta^{Q_i}}(\hat{s},\hat{a}))^2 \right\} \quad (7)$$

Here, $\hat{y} \leftarrow \hat{r} + \gamma \min_{i=1,2} Q_{\theta^{Q_i\prime}}(\hat{s}', \tilde{a})$, where $\tilde{a} \leftarrow \pi_{\theta^{\mu\prime}}(\hat{s}') + \epsilon$, with $\epsilon \sim \mathcal{N}(0, 0.2)$ representing noise, which is clipped to the range $(-0.5, 0.5)$. The first part focuses on minimizing the TD loss to ensure the convergence of the auxiliary model. Regarding the second part of the loss, $-J^{aux2}(\bar{\theta^{\mu}})$ represents the Q-value (long-term value) of the actor's actions after the auxiliary update, while $-J^{aux1}(\theta^{\mu})$ represents the Q-value before the update. The goal is to maximize the difference $-J^{aux2}(\bar{\theta^{\mu}}) - (-J^{aux1}(\theta^{\mu}))$. Therefore, the second part, $\mathcal{L}_2$, aims to minimize $J^{aux2}(\bar{\theta^{\mu}}) - J^{aux1}(\theta^{\mu})$, as given by:

$$\mathcal{L}_2 = tanh(J^{aux2}(\bar{\theta^{\mu}}) - J^{aux1}(\theta^{\mu})) \quad (8)$$

where the $\tanh$ function smoothly normalizes the difference $J^{aux2}(\bar{\theta^{\mu}}) - J^{aux1}(\theta^{\mu})$, ensuring the output remains within the $[-1, 1]$ range, thereby preventing excessive values. Based on the above, the auxiliary model $\hat{\theta}$ is updated as follows:

$$\hat{\theta} \leftarrow \hat{\theta} + \alpha \nabla_{\hat{\theta}} (\mathcal{L}_1 + \mathcal{L}_2) \quad (9)$$

(3) Dynamic adjustment of sparse topology. After updating the networks, we dynamically adjust their sparse topology using a two-step "drop and grow" procedure. Importance values identify connections to remove, while new connections are grown randomly. Following SOTA ST methods in DRL (Sokar et al. 2021)(Xu, Chen, and Wang 2024), we use absolute values to represent connection importance, where lower absolute values indicate lower importance.

(3.1) Drop connections. We first identify a subgroup of connections within each layer with the lowest importance based on their absolute values, referred to as low $\mathcal{I}_{\boldsymbol{\theta}_i}$. The number of these low-importance connections is denoted by N_d. To remove these connections, we use a mask $\boldsymbol{M_\theta}$, which sets their values to 0. The mask is updated as follows: $\boldsymbol{M_\theta} = \boldsymbol{M_\theta} - \mathbf{1}\left(-\mathcal{I}_{\boldsymbol{\theta}_i}, N_d\right), \quad \text{for all } i \in \{1, \ldots, \|\boldsymbol{\theta}\|_0\}$. Here, the norm $\|\cdot\|_0$ represents the standard L_0 norm, and $\boldsymbol{\theta}_i$ denotes the connection at index i. The function $\mathbf{1}$ is an indicator that converts the value to 1 based on the condition $-\mathcal{I}_{\boldsymbol{\theta}_i}$ (lower importance) and the count N_d.

(3.2) Grow connections. In the second step, we randomly add N_g connections to positions where no connections currently exist. Here, N_g is equal to N_d. Similarly, the connection growth is achieved using $\boldsymbol{M_\theta}$. The update process for $\boldsymbol{M_\theta}$ is as follows: $\boldsymbol{M_\theta} = \boldsymbol{M_\theta} + \mathbf{1}\left((\boldsymbol{\theta}_i == 0) \wedge \text{Random}, N_g\right), \quad \text{for all } i \in \{1, \ldots, \|\boldsymbol{\theta}\|_0\}$. Here, $\wedge$ denotes the logical AND operator, and $\boldsymbol{\theta}_i == 0$ indicates that no connection exists at the given position. Any newly added connections are initialized with zero values. After computing $\boldsymbol{M_\theta}$, we update the model $\boldsymbol{\theta}$ using the element-wise multiplication: $\boldsymbol{\theta} = \boldsymbol{\theta} \odot \boldsymbol{M_\theta}$.

(4) Updating the target networks. The target networks consist of an actor and two critics, and are updated using a soft update method with a delay. The update process is as follows: $\theta^{\mu\prime} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu\prime}; \theta^{Q_i\prime} \leftarrow \tau\theta^{Q_i} + (1-\tau)\theta^{Q_i\prime}, i = 1, 2$. Here, τ is the update coefficient.

Global Aggregation

After each local model is trained for a specified duration, defined by the communication frequency N_c according to the configuration of the integrated baseline FDRL method, the local models are uploaded to the server. These local models are then aggregated into a global model, as follows: $\boldsymbol{\theta}^{agg} = F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{N_a})$. Here, N_a denotes the number

of local models, $F(\cdot)$ is the aggregation function, and $\boldsymbol{\theta}_j$ represents the parameters of the j-th local model. Different aggregation functions result in different global models. When applying our method to various baseline FDRL approaches, the aggregation function aligns with the specific method. For example, the average aggregation function is given by: $F(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{N_a}) = \sum_{j=1}^{N_a} \frac{1}{N_a} \boldsymbol{\theta}_j$.

After aggregating local models into a global model, each client downloads it to continue local training. Since local models have distinct sparse topologies, current aggregation methods like average aggregation retain existing connections but recalculate their values (Cha et al. 2020)(Jiang et al. 2025)(Wang et al. 2023). This results in a global model with reduced sparsity, prompting pruning post-aggregation to match the sparsity of the local models. First, we compute the absolute value $\tilde{\theta}_j^{agg}$ of each connection in the global model, where $j = 1, 2, \ldots, \|\tilde{\theta}_j^{agg}\|_0$, and $\|\tilde{\theta}_j^{agg}\|_0$ represents the total number of connections. The values are sorted in descending order, and the k largest connections to retain are identified, with k determined based on the sparsity of the global model. Next, we remove connections whose absolute value is smaller than the k-th largest connection, $\tilde{\theta}_k^{agg}$ (i.e., $\tilde{\theta}_j^{agg} < \tilde{\theta}_k^{agg}$). The pruned connections are set to zero using a binary mask $\boldsymbol{M}_{\boldsymbol{\theta}^{agg}}$, defined as: $\boldsymbol{\theta}^{agg} = \boldsymbol{\theta}^{agg} \odot \boldsymbol{M}_{\boldsymbol{\theta}^{agg}}$.

Complexity Analysis

This section evaluates the complexity of our method, including floating-point operations (FLOPs), storage consumption, and communication cost. The FLOPs reflect the computational cost of our approach. Let $\boldsymbol{\theta}_{baseline}$ denote the dense model of the baseline FDRL, $\hat{\boldsymbol{\theta}}$ the dense auxiliary model, I the input dimension of a fully connected layer (FCL), O the output dimension of the FCL, and S the sparsity.

(1) FLOPs: To evaluate the FLOPs of our approach, we use an MLP as a representative example. Previous studies (Sokar et al. 2021)(Xu, Chen, and Wang 2024)(Grooten et al. 2023) show that the total FLOPs of a network are the sum of the FLOPs of each FCL. The FLOPs for an individual FCL are expressed as $(1 - S) \times (2 \times I - 1) \times O$. Additionally, the computational cost of adjusting the network topology, such as adding or removing connections, is relatively small compared to the model's computation and training and is often overlooked in current research (Sokar et al. 2021)(Tan et al. 2023)(Xu et al. 2025). Based on these factors, the FLOPs of our method with 50% sparsity are linearly related to the network's parameter size, i.e., $O(0.5 * \|\boldsymbol{\theta}_{baseline}\|_0 + 0.5\|\hat{\boldsymbol{\theta}}\|_0)$, while the FLOPs of baseline FDRL methods are linearly related to $O(\|\boldsymbol{\theta}_{baseline}\|_0)$. Furthermore, since $\hat{\boldsymbol{\theta}}$ is equivalent in size to a critic in existing FDRL methods, we have $O(0.5*\|\boldsymbol{\theta}_{baseline}\|_0+0.5\|\hat{\boldsymbol{\theta}}\|_0) < O(\|\boldsymbol{\theta}_{baseline}\|_0)$, demonstrating that the FLOPs of our method are lower than those of baseline FDRL methods.

(2) Storage Consumption: The storage consumption of our method is $O(0.5 * \|\boldsymbol{\theta}_{baseline}\|_0 + |\mathcal{D}| + 0.5 * \|\hat{\boldsymbol{\theta}}\|_0)$ when the model sparsity is 50%. The storage consumption of existing FDRL methods is $O(\|\boldsymbol{\theta}_{baseline}\|_0 + |\mathcal{D}|)$. Since $\hat{\boldsymbol{\theta}}$ in our approach is equivalent in size to a single critic in DRL, we have $O(0.5 * \|\boldsymbol{\theta}_{baseline}\|_0 + |\mathcal{D}| + 0.5 * \|\hat{\boldsymbol{\theta}}\|_0) < O(\|\boldsymbol{\theta}_{baseline}\|_0 + |\mathcal{D}|)$. As a result, the storage consumption of our method is lower than that of existing FDRL methods.

(3) Communication cost: In FDRL, each client uploads its model to the server and downloads the global model, so communication costs directly depend on the network size. The network size, determined by the number of parameters in the network, is given by $\sum_{i=1}^{N-1}(1 - S_i)n_i \times n_{i+1}$, as illustrated using the example of an N-layer MLP. With 50% sparsity applied to each component $\boldsymbol{\theta}_{baseline}$ and $\hat{\boldsymbol{\theta}}$ in our method, the network size is smaller compared to baseline FDRL methods that use dense models, as shown in the analysis of FLOPs and storage overhead above. Therefore, our method results in lower communication costs due to its reduced network size.

Experiments

We use Torch 1.2.0, Gym 0.16.0, and mujoco-py 1.50.0.1 for software, while the hardware consists of an Intel Core i7-9700 processor, 64GB RAM, and an NVIDIA RTX 2080 GPU. Each method is tested with five different random seeds. The supplementary materials provide implementation details of our method and the evolution curves of episode returns for each test unit.

Experiment Settings

Baselines. Six SOTA FDRL methods are used as baselines: Server-Client Collaborative Distillation (SCCD) (Mai et al. 2023), Federated Reinforcement Distillation (FRD) (Jiang et al. 2025), FedKL (Xie and Song 2023), QAvg (Jin et al. 2022), Federated Reinforcement Learning (FRL) (Chen 2024), and Asynchronous Federated Reinforcement Learning with Policy Gradient Updates (AFedPG) (Lan et al. 2024).

Benchmarks. We use nine learning tasks, consisting of five MuJoCo tasks (Grooten et al. 2023): HalfCheetah-v2, Hopper-v2, Walker2d-v2, Ant-v2, and Humanoid-v2, and four Gym tasks (Mai et al. 2023): CartPole, Pendulum, BipedalWalkerHardcore, and LunarLander. Five clients are deployed for each task.

For the five MuJoCo tasks, we implement dynamic, heterogeneous client environments as described in (Grooten et al. 2023), introducing random noise from a normal distribution, $\psi\mathcal{N}(0, 1)$, to each client's observations. Here, $\mathcal{N}(0, 1)$ denotes the standard normal distribution, and ψ represents the disturbance level, which can be set to 0, 1, or 3, with $\psi = 3$ representing maximum disturbance and making the task most challenging. The communication frequency for these MuJoCo tasks is set to every 10,000 steps, with a total of 100 communication rounds. For the four Gym tasks, we configure different environment parameters for each client, following the approach in (Mai et al. 2023), creating heterogeneous environments. For example, CartPole and Pendulum involve variations in pole length, gravity, and mass across clients, BipedalWalkerHardcore introduces diverse obstacles with varying appearance probabilities, and LunarLander features irregular lunar surface conditions and varying landing pad heights. To ensure dynamic environments, physical parameters for each client vary within a 50% range of the original values, introducing significant disturbances. For instance, in

CartPole, the pole length fluctuates within $\pm 50\%$ of its original value. Communication frequencies follow the settings from (Mai et al. 2023): 20 steps for CartPole, 100 steps for Pendulum, 400 steps for BipedalWalkerHardcore, and 300 steps for LunarLander.

Metrics. Given that we have shown our method outperforms existing FDRL approaches in complexity analysis, we evaluate the global model's performance using two key metrics: the final return and the episode return. The main text focuses on the final return, the average reward from the global model over the last 10 communication rounds. The supplementary material discusses the episode return, which calculates the average reward per communication round and is typically presented as a progression curve. Higher values in both metrics indicate better model performance.

Hyperparameters. We set the sparsity level to 50% ($\xi^1 = \xi^2 = 64$), which was identified as the optimal value through parameter sensitivity testing. Additionally, N_d is set equal to N_g, using the optimal value verified in existing literature (Sokar et al. 2021)(Xu, Chen, and Wang 2024), which is 10% of the current model parameters.

Experimental Result

Comparison under MuJoCo tasks. This section evaluates our method's improvements over three SOTA FDRL approaches: QAvg, FRL, and AFedPG. We use noisy MuJoCo tasks for validation, focusing on the final return. The noise level ψ is set to 0, 1, and 3, respectively.

Table 1 presents the experimental results, highlighting the final return. It shows the mean return, calculated as the average return of the global model over the last 10 communication rounds from five runs with different random seeds, along with the standard deviation. All tables in the experimental section follow this format. From the results, two conclusions can be drawn. First, baseline FDRL methods show a progressively severe decline in returns as the noise level increases from 0 to 3, with the most significant drop at level 3. This suggests that dynamic environments, with more stochastic state transition dynamics, degrade FDRL performance. Second, when our method is integrated, including Ours+QAvg, Ours+FRL, and Ours+AFedPG, returns exceed those of the baseline methods in most cases, demonstrating that our approach enhances FDRL performance in high-dimensional control tasks across various levels of perturbation. Dynamic transitions render static model topologies unsuitable, while also causing biased rewards and losses in each local model, which leads to suboptimal policies. When aggregated, these suboptimal local models degrade the global model's performance, creating a negative feedback loop that further trains local models on the degraded global model. As disturbances increase, the performance of existing FDRL methods continues to deteriorate. The performance gains of our method stem from ST and the auxiliary model. ST dynamically adjusts the topology of each local model, boosting performance, while the auxiliary loss regulates policy loss, steering the policy toward higher returns. By integrating ST and the auxiliary model, each local model dynamically adapts its topology and loss, leading to improved performance in current FDRL methods.

Comparison under Gym tasks. This section evaluates the improvements of our method over three SOTA FDRL approaches: SCCD, FRD, and FedKL, using Gym tasks for validation, with a focus on final returns. Perturbation levels for each client are set at 50%. Table 2 presents the experimental results, highlighting that incorporating our method, including Ours+SCCD, Ours+FRD, and Ours+FedKL, boosts the performance of baseline FDRL methods, leading to higher returns. These results validate the effectiveness of our approach in improving FDRL methods across various dynamic Gym tasks. Our method improves upon existing approaches by using ST to dynamically adjust client model topologies in different environments, and the auxiliary model adapts policy loss to guide the model toward higher returns.

Ablation verification. This section validates the effectiveness of the two modules in our method: ST and the Auxiliary Model (AM). We evaluate various methods using SCCD and FedKL as baseline FDRL algorithms across two standard Gym tasks, CartPole and Pendulum, with 5 clients and a 50% disturbance level. Performance is assessed based on the final return, with results presented in Table 3. The findings show that using ST or AM individually, such as ST+SCCD or AM+SCCD, does not yield optimal performance; both configurations are outperformed by the full version of our method, emphasizing the need for both modules. The ST module adjusts model topology, while the Auxiliary Model regulates policy loss. Together, they enable self-regulation during training, leading to optimal performance.

Hyperparameter sensitivity testing. This section examines the impact of model sparsity, a hyperparameter, on the performance of our method. We evaluate various methods using SCCD and FedKL as baseline FDRL algorithms on two Gym tasks, CartPole and Pendulum, with 5 clients and a 50% disturbance level. Model sparsity is set at 50%, 70%, and 90%, with 50% chosen for this study. Performance is assessed based on the final return, and results are presented in Table 4. The findings show that when sparsity exceeds 50%, increasing sparsity reduces model parameters, lowering both training and communication burdens. However, this also leads to a decline in performance. At 50% sparsity, our method achieves optimal performance while reducing training and communication costs compared to baseline methods. Therefore, we adopt a sparsity level of 50%.

Comparison with varying participant numbers. This section compares our method's performance with baseline methods as the number of clients in federated learning increases. We evaluate Ours+SCCD and Ours+FedKL on two standard Gym tasks, CartPole and Pendulum, using 5, 10, and 15 clients, with a 50% disturbance level for clients. Table 5 shows the results in terms of final return. The findings indicate that the same FDRL method performs differently with varying client numbers. However, integrating our method with these approaches consistently improves model performance, demonstrating its effectiveness. Despite variations in client numbers, the learning process remains consistent. Our method dynamically adjusts model topology using ST and regulates policy loss through the auxiliary model, enhancing

Static environment (A disturbance level of 0)					
Methods	HalfCheetah	Hopper	Walker2d	Ant	Humanoid
QAvg	10470.72 ± 571.24	1587.7 ± 812.27	739.68 ± 508.72	2133.83 ± 1284.37	2858.84 ± 804.25
Ours+QAvg	$\mathbf{10649.88} \pm 1032.78$	$\mathbf{2283.57} \pm 1197.35$	$\mathbf{3370.63} \pm 1279.05$	$\mathbf{5098.16} \pm 446.74$	$\mathbf{3012.95} \pm 872.47$
FRL	10413.22 ± 657.27	2397.48 ± 1026.72	4336.6 ± 986.66	2307.97 ± 893.52	2759.49 ± 1852.01
Ours+FRL	10365.02 ± 1721.2	$\mathbf{2913.37} \pm 1063.85$	$\mathbf{4500.55} \pm 649.08$	$\mathbf{4566.21} \pm 819.91$	$\mathbf{4330.87} \pm 935.47$
AFedPG	10253.8 ± 1489.44	3147.12 ± 812.26	4073.8 ± 640.73	3534.45 ± 828.12	3832.78 ± 712.39
Ours+AFedPG	$\mathbf{12062.33} \pm 1751.76$	$\mathbf{3454.37} \pm 402.17$	$\mathbf{4143.91} \pm 1251.4$	$\mathbf{4205.8} \pm 602.48$	$\mathbf{4316.23} \pm 828.68$
A disturbance level of 1					
QAvg	8063.06 ± 953.99	757.49 ± 511.41	424.88 ± 289.63	1939.87 ± 1170.18	2815.56 ± 749.1
Ours+QAvg	$\mathbf{8651.54} \pm 669.54$	$\mathbf{831.13} \pm 469.43$	$\mathbf{1195.06} \pm 572.41$	$\mathbf{3987.09} \pm 867.08$	$\mathbf{4443.03} \pm 4089.87$
FRL	7356.14 ± 1174.69	481.98 ± 308.37	1526.48 ± 1176.06	2472.42 ± 1150.22	1928.98 ± 1268.21
Ours+FRL	$\mathbf{9341.62} \pm 621.23$	$\mathbf{1301.02} \pm 969.22$	$\mathbf{2138.87} \pm 906.19$	$\mathbf{4224.7} \pm 1018.19$	$\mathbf{4308.19} \pm 703.64$
AFedPG	7927.79 ± 858.08	741.85 ± 346.6	463.5 ± 218.6	3393.69 ± 1437.34	3845.11 ± 1033.81
Ours+AFedPG	$\mathbf{9347.46} \pm 479.42$	$\mathbf{1563.17} \pm 873.19$	$\mathbf{2508.24} \pm 1059.08$	$\mathbf{3987.09} \pm 867.08$	$\mathbf{4321.83} \pm 957.36$
A disturbance level of 3					
QAvg	5765.02 ± 630.17	291.74 ± 54.15	308.05 ± 53.35	-178.61 ± 176.28	1262.3 ± 942.77
Ours+QAvg	$\mathbf{7234.65} \pm 602.27$	$\mathbf{440.57} \pm 210.76$	284.08 ± 169.47	$\mathbf{779.42} \pm 285.49$	$\mathbf{2858.84} \pm 804.25$
FRL	5915.04 ± 419.58	257.98 ± 77.87	371.06 ± 83.02	2304.22 ± 1328.07	921.34 ± 225.16
Ours+FRL	$\mathbf{7828.79} \pm 395.78$	$\mathbf{1472.49} \pm 840.67$	$\mathbf{3009.24} \pm 1099.04$	$\mathbf{3951.14} \pm 803.0$	$\mathbf{4606.58} \pm 722.08$
AFedPG	5396.47 ± 682.48	286.17 ± 98.28	365.43 ± 48.51	2405.04 ± 890.72	3895.28 ± 917.21
Ours+AFedPG	$\mathbf{8036.75} \pm 778.22$	$\mathbf{1670.02} \pm 986.72$	$\mathbf{1573.29} \pm 629.63$	$\mathbf{2981.54} \pm 1230.07$	$\mathbf{4217.13} \pm 638.15$

Table 1: Performance comparison under MuJoCo tasks (Mean ± Standard Deviation). Here, Ours+QAvg refers to the QAvg method incorporating our approach. The bolded results indicate that our method achieves a higher average return (Mean) compared to the baseline.

Methods	CartPole	Pendulum
SCCD	148.15 ± 18.0	-456.25 ± 182.33
Ours+SCCD	$\mathbf{159.77} \pm 10.69$	$\mathbf{-290.94} \pm 75.92$
FRD	83.46 ± 7.98	-540.4 ± 214.91
Ours+FRD	$\mathbf{174.22} \pm 26.81$	$\mathbf{-283.53} \pm 150.59$
FedKL	86.12 ± 25.56	-395.8 ± 181.63
Ours+FedKL	$\mathbf{151.98} \pm 56.86$	$\mathbf{-302.61} \pm 171.84$
Methods	BipedalWalkerHardcore	Lunar Lander
SCCD	169.6 ± 30.28	99.11 ± 98.07
Ours+SCCD	$\mathbf{195.68} \pm 32.35$	$\mathbf{240.66} \pm 32.85$
FRD	135.01 ± 72.22	236.24 ± 54.26
Ours+FRD	$\mathbf{165.4} \pm 58.38$	$\mathbf{254.03} \pm 27.44$
FedKL	130.39 ± 47.35	265.69 ± 10.94
Ours+FedKL	$\mathbf{218.44} \pm 47.67$	$\mathbf{275.08} \pm 13.33$

Table 2: Performance comparison of our method under Gym tasks (Mean ± Standard Deviation).

Methods	CartPole	Pendulum
ST+SCCD	126.63 ± 17.44	-419.76 ± 106.97
AM+SCCD	135.91 ± 25.59	-445.86 ± 66.22
Ours+SCCD	$\mathbf{159.77} \pm 10.69$	$\mathbf{-290.94} \pm 75.92$
ST+FedKL	120.27 ± 41.39	-354.07 ± 24.76
AM+FedKL	105.35 ± 29.02	-355.93 ± 41.96
Ours+FedKL	$\mathbf{151.98} \pm 56.86$	$\mathbf{-302.61} \pm 171.84$

Table 3: Ablation verification (Mean ± Standard deviation).

Methods	CartPole	Pendulum
Ours+SCCD (50%)	$\mathbf{159.77} \pm 10.69$	$\mathbf{-290.94} \pm 75.92$
Ours+FedKL (50%)	$\mathbf{151.98} \pm 56.86$	$\mathbf{-302.61} \pm 171.84$
Ours+SCCD (70%)	140.22 ± 21.73	-512.9 ± 209.59
Ours+FedKL (70%)	142.19 ± 63.23	-311.28 ± 180.6
Ours+SCCD (90%)	123.28 ± 24.8	-679.47 ± 309.34
Ours+FedKL (90%)	132.64 ± 51.13	-358.58 ± 182.84

Table 4: Hyperparameter sensitivity (Mean ± Standard deviation).

Methods	CartPole	Pendulum
SCCD (5)	148.15 ± 18.0	-456.25 ± 182.33
Ours+SCCD (5)	$\mathbf{159.77} \pm 10.69$	$\mathbf{-290.94} \pm 75.92$
FedKL (5)	86.12 ± 25.56	-395.8 ± 181.63
Ours+FedKL (5)	$\mathbf{151.98} \pm 56.86$	$\mathbf{-302.61} \pm 171.84$
SCCD (10)	160.11 ± 11.42	-504.84 ± 73.71
Ours+SCCD (10)	$\mathbf{167.38} \pm 13.57$	$\mathbf{-469.26} \pm 58.42$
FedKL (10)	69.47 ± 20.2	-455.49 ± 127.03
Ours+FedKL (10)	$\mathbf{151.19} \pm 7.3$	$\mathbf{-437.92} \pm 132.43$
SCCD (15)	159.74 ± 10.94	-543.35 ± 59.61
Ours+SCCD (15)	$\mathbf{172.07} \pm 10.82$	$\mathbf{-464.07} \pm 28.22$
FedKL (15)	60.13 ± 23.23	-552.42 ± 36.51
Ours+FedKL (15)	$\mathbf{154.17} \pm 9.06$	$\mathbf{-397.81} \pm 141.0$

Table 5: Performance comparison under varying client quantities (Mean ± Standard deviation). Here, Ours+SCCD (5) represents the performance of the Ours+SCCD method with five clients.

the performance of existing FDRL methods across different client quantities.

Conclusion

This work presents a generic method that can be seamlessly integrated into existing FDRL approaches to address dynamic state transitions. We introduce ST to adaptively adjust the model topology, enhancing performance during training while reducing model parameters. Additionally, we propose an auxiliary model to regulate the policy loss of local models, facilitating updates toward higher returns. We validate our method across nine dynamic environments, showing significant improvements in six SOTA FDRL methods in terms of return. Future work will explore using higher model sparsity to further reduce the FDRL model complexity without compromising performance in dynamic environments.

Acknowledgements

This work is supported in part by the Hong Kong Research Grant Council under General Research Fund (GRF) 11216323 and Research Impact Fund (RIF) project R5060-19. This work acknowledges the Hon Hai-CityU Joint Research Center and Hon Hai Research Institute for their financial and technical support.

References

Bing, Z.; Lerch, D.; Huang, K.; and Knoll, A. 2023. Meta-Reinforcement Learning in Non-Stationary and Dynamic Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3476–3491.

Bodas, A.; Upadhyay, B.; Nadiger, C.; and Abdelhak, S. 2018. Reinforcement learning for game personalization on edge devices. In *2018 International Conference on Information and Computer Technologies (ICICT)*, 119–122. IEEE.

Cang, C.; Rajeswaran, A.; Abbeel, P.; and Laskin, M. 2021. Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl. *arXiv preprint arXiv:2106.09119*.

Cha, H.; Park, J.; Kim, H.; Bennis, M.; and Kim, S.-L. 2020. Proxy experience replay: Federated distillation for distributed reinforcement learning. *IEEE Intelligent Systems*, 35(4): 94–101.

Chen, X. 2024. Federated Reinforcement Learning-Driven Offloading Strategy for Edge Computing. In *2024 6th International Conference on Electronics and Communication, Network and Computer Technology (ECNCT)*, 467–470. IEEE.

DiMarco, E. K.; Shipp, A. R.; and Kishida, K. T. 2024. Expected reward value and reward prediction errors reinforce but also interfere with human time perception. *bioRxiv*, 2024–04.

Fan, X.; Ma, Y.; Dai, Z.; Jing, W.; Tan, C.; and Low, B. K. H. 2021. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems*, 34: 1007–1021.

Fu, Y.; Li, C.; Yu, F. R.; Luan, T. H.; and Zhang, Y. 2022. A selective federated reinforcement learning strategy for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(2): 1655–1668.

Grooten, B.; Sokar, G.; Dohare, S.; Mocanu, E.; Taylor, M. E.; Pechenizkiy, M.; and Mocanu, D. C. 2023. Automatic noise filtering with dynamic sparse training in deep reinforcement learning. *arXiv preprint arXiv:2302.06548*.

Javeed, D.; Saeed, M. S.; Kumar, P.; Jolfaei, A.; Islam, S.; and Islam, A. N. 2023. Federated learning-based personalized recommendation systems: An overview on security and privacy challenges. *IEEE Transactions on Consumer Electronics*, 70(1): 2618–2627.

Jiang, W.; Wang, J.; Zhang, X.; Bao, W.; Tan, C.; and Fan, F. X. 2025. FedHPD: Heterogeneous Federated Reinforcement Learning via Policy Distillation. *arXiv preprint arXiv:2502.00870*.

Jin, H.; Peng, Y.; Yang, W.; Wang, S.; and Zhang, Z. 2022. Federated reinforcement learning with environment heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, 18–37. PMLR.

Khayatian, F.; Nagy, Z.; and Bollinger, A. 2021. Using generative adversarial networks to evaluate robustness of reinforcement learning agents against uncertainties. *Energy and Buildings*, 251: 111334.

Lan, G.; Han, D.-J.; Hashemi, A.; Aggarwal, V.; and Brinton, C. G. 2024. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. *arXiv preprint arXiv:2404.08003*.

Lee, J.; Kim, S.; Kim, S.; Jo, W.; and Yoo, H.-J. 2021. Gst: Group-sparse training for accelerating deep reinforcement learning. *arXiv preprint arXiv:2101.09650*.

Lee, K.; Seo, Y.; Lee, S.; Lee, H.; and Shin, J. 2020. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, 5757–5766. PMLR.

Liang, X.; Liu, Y.; Chen, T.; Liu, M.; and Yang, Q. 2022. Federated transfer reinforcement learning for autonomous driving. In *Federated and Transfer Learning*, 357–371. Springer.

Liu, B.; Wang, L.; and Liu, M. 2019. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4): 4555–4562.

Livne, D.; and Cohen, K. 2020. Pops: Policy pruning and shrinking for deep reinforcement learning. *IEEE Journal of Selected Topics in Signal Processing*, 14(4): 789–801.

Luo, Y.; Ji, T.; Sun, F.; Zhang, J.; Xu, H.; and Zhan, X. 2024. OMPO: A Unified Framework for RL under Policy and Dynamics Shifts. *arXiv preprint arXiv:2405.19080*.

Mai, W.; Yao, J.; Chen, G.; Zhang, Y.; Cheung, Y.-M.; and Han, B. 2023. Server-Client Collaborative Distillation for Federated Reinforcement Learning. *ACM Transactions on Knowledge Discovery from Data*, 18(1): 1–22.

Rengarajan, D.; Ragothaman, N.; Kalathil, D.; and Shakkottai, S. 2025. Federated ensemble-directed offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 6154–6179.

Sokar, G.; Mocanu, E.; Mocanu, D. C.; Pechenizkiy, M.; and Stone, P. 2021. Dynamic sparse training for deep reinforcement learning. *arXiv preprint arXiv:2106.04217*.

Tan, Y.; Hu, P.; Pan, L.; Huang, J.; and Huang, L. 2023. Rlx2: Training a sparse deep reinforcement learning model from scratch. *International Conference on Learning Representations*.

Vischer, M. A.; Lange, R. T.; and Sprekeler, H. 2021. On lottery tickets and minimal task representations in deep reinforcement learning. *arXiv preprint arXiv:2105.01648*.

Wang, J.; Hu, J.; Mills, J.; Min, G.; Xia, M.; and Georgalas, N. 2023. Federated Ensemble Model-Based Reinforcement Learning in Edge Computing. *IEEE Transactions on Parallel and Distributed Systems*, 34(6): 1848–1859.

Wei, C.-Y.; and Luo, H. 2021. Non-stationary reinforcement learning without prior knowledge: An optimal black-box

approach. In *Conference on learning theory*, 4300–4354. PMLR.

Wen, J.; Dai, H.; He, J.; Xi, M.; Xiao, S.; and Yang, J. 2023. Federated offline reinforcement learning with multimodal data. *IEEE transactions on consumer electronics*, 70(1): 4266–4276.

Xie, Z.; and Song, S. 2023. Fedkl: Tackling data heterogeneity in federated reinforcement learning by penalizing kl divergence. *IEEE Journal on Selected Areas in Communications*, 41(4): 1227–1242.

Xu, M.; Chen, X.; Fu, W.; Lin, Y.-R.; Li, Y.-H.; and Wang, J. 2025. A Unified Sparse Training Framework for Lightweight Lifelong Deep Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–15.

Xu, M.; Chen, X.; and Wang, J. 2024. A Novel Topology Adaptation Strategy for Dynamic Sparse Training in Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–13.

Xue, Z.; Cai, Q.; Liu, S.; Zheng, D.; Jiang, P.; Gai, K.; and An, B. 2024. State regularized policy optimization on data with dynamics shift. *Advances in neural information processing systems*, 36.

Yuan, Z.; Xu, S.; and Zhu, M. 2024. Federated reinforcement learning for robot motion planning with zero-shot generalization. *Automatica*, 166: 111709.

Yue, S.; Deng, Y.; Wang, G.; Ren, J.; and Zhang, Y. 2024. Federated offline reinforcement learning with proximal policy evaluation. *Chinese Journal of Electronics*, 33(6): 1360–1372.

Zhang, H.; He, Z.; and Li, J. 2019. Accelerating the deep reinforcement learning with neural network compression. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Zhou, D.; Zhang, Y.; Sonabend-W, A.; Wang, Z.; Lu, J.; and Cai, T. 2024. Federated offline reinforcement learning. *Journal of the American Statistical Association*, 119(548): 3152–3163.

Zhuo, H. H.; Feng, W.; Lin, Y.; Xu, Q.; and Yang, Q. 2019. Federated deep reinforcement learning. *arXiv preprint arXiv:1901.08277*.